

Get started here

ESTIMATING PRODUCT BACKLOG PRODUCT OWNER SPRINTS USER STORIES

Live
Online
Certified
Scrum
Training
Courses
Now
Available
No
travel
required.
Train
online
at
home
and
become
a
Certified
ScrumMaster®
or
Certified
Scrum
Product
Owner®.
[View
here](#)

[Why I Don't Emphasize Sprint Goals](#)

Sprint goals are considered a mandatory part of Scrum. Here's why I disagree.

Mar 21, 2023

[User Stories: How to Create Story Maps](#)

Story maps help to create a shared understanding of the product, visualize user needs, and elicit ...

Mar 14, 2023

[Read](#)

[How to Run a Successful User Story Writing Workshop](#)

What you need to know to conduct a story-writing workshop with your team.

Feb 28, 2023 • [3 Comments](#)

[Read](#)

[Get Weekly Email Tips](#)

[What Is Cross-Functional Collaboration in Agile?](#)

Perhaps the most prevalent and persistent myth in agile is that a cross-functional team is one on ...

Feb 14, 2023 • [49 Comments](#)

[Read](#)

[Be a Great Product Owner: Six Things Teams and Scrum Masters Need](#)

Learn six ways effective product owners ensure their teams' success.

Jan 31, 2023 • [16 Comments](#)

[Read](#)

[Six Attributes of a Great ScrumMaster](#)

Scrum Masters exhibit a shared set of traits. Discover 6 in-demand qualities, plus a bonus ...

Jan 17, 2023 • 32 Comments

[Read](#)

[What Happens When During a Sprint](#)

Succeeding with Scrum is easier when you know when and why to conduct each of the Scrum events ...

Jan 03, 2023 • 34 Comments

[Read](#)

[What Are Agile Story Points?](#)

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments

[Read](#)

[Don't Equate Story Points to Hours](#)

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • [39 Comments](#)

[Read](#)

[Epic, Features and User Stories](#)

I've been getting more and more emails lately from people confused about the difference between ...

Nov 08, 2022 • [93 Comments](#)

[Read](#)

[Nine Agile Halloween Costumes for Scrum Teams](#)

Oct 25, 2022

[Read](#)

8 Reasons Scrum Is Hard to Learn (but Worth It)

Transitioning to Scrum is worth it, but some aspects are challenging.

Oct 11, 2022

[Read](#)

OLDER POSTS

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

Hi! I'm Mike Cohn and I've been blogging here since 2005. That means there's a lot of content here.

I've prepared this handy guide to help you find what you want on the most important topics I've written about. In addition to blog posts, you'll find links to presentations, videos and more.

To get started, scroll down to find the topic you're interested in then click on one of the personally selected items on that topic.

From there, you'll be able to navigate to more items on the same topic.

Estimating

An estimate of story points reflects the effort to do the work as influenced by four factors and not just complexity.

[Agile Estimating](#)

Maybe you've heard about agile software development projects but aren't sure if they allow for the ...

[Presentation](#)

[Read](#)

[Story Points Estimate Effort Not Just Complexity](#)

This agile estimation blog post from Mike Cohn looks at story points. Learn why user story points ...

Jun 21, 2010 • [219 Comments](#) [Read](#)

Product Backlog

A product backlog is a prioritized list of the functionality that remains to be added to the product. A well groomed backlog ensures that teams are always working on the most valuable features. For details, start with one of these resources.

[What Is a Product?](#)

In this agile principles post from Mike Cohn, discover how Scrum organizations define products. ...

Sep 06, 2016 • 41 Comments [Read](#)

[Product Backlog Refinement \(Grooming\)](#)

Learn about product backlog grooming: how, when, and why the agile team and product owner refine ...

May 26, 2015 • 58 Comments [Read](#)

[Writing the Product Backlog](#)

[Just in Time and Just Enough](#)

This article addresses the issue of how much detail should be included in product backlog items and ...

Feb 10, 2008 • 10 Comments [Read](#)

[Prioritizing Your Product Backlog](#)

The biggest risk to most projects is building the wrong product. Regardless of how fast your agile ...

Presentation [Read](#)

Product Owner

In Scrum, the product owner represents the users or customers of a product or system. Product owners ensure that the right product is being built in the right order. To learn more about the product owner role, select one of these items.

[Presentation for Product Owners: Storytelling with the Prioritized Product Backlog](#)

The vast majority of what has been written about agile processes is intended for programmers and ...

Presentation [Read](#)

[Incorporating Learning and Expected Cost of Change](#)

Product owners are typically asked to prioritize based on the nebulous term "business value." But ...

Presentation [Read](#)

[Can a Product Owner Dictate the Architecture?](#)

In general, a Scrum product owner's job is to specify what to build, not how to build it. Read this ...

Jun 16, 2015 • 21 Comments [Read](#)

[The Product Owner in a Sprint Retrospective](#)

Is your agile team excluding the Scrum product owner from your sprint retrospectives? Check out ...

Feb 06, 2013 • 9 Comments [Read](#)

Sprints

Sprints, or iterations, are timeboxed periods where a team plans, builds, delivers, and reviews a set of small, valuable, potentially shippable features. Sprints last no more than one month, creating opportunities for frequent feedback on both the product and the process.

A Regular Heartbeat

We all crave regularity. We want a steady rhythm and a strong downbeat so we know the steps we need ...

Jan 26, 2005

[Read](#)

Don't Take Partial Credit for

Semi-Finished Stories

Accurate measures of a Scrum team's velocity don't include partial credit for user stories that ...

Dec 17, 2013 • 38 Comments [Read](#)

Selecting the Right Iteration Length

A key consideration in adopting an iterative process is selecting how long your iterations will be. ...

Mar 03, 2006 • 4 Comments [Read](#)

Six Times Two Plus One Equals a Good Project Cadence

In last month's newsletter I wrote about the idea that everything happens within a sprint. There is ...

Feb 20, 2014 • 13 Comments [Read](#)



User Stories

User stories are simple, short descriptions of desired functionality. Their power comes from the conversations they enable, allowing people to shift from writing about requirements to talking about them.

Introduction to User Stories

The technique of expressing requirements as user stories is one of the most broadly applicable ...

Presentation

[Read](#)

Non-functional Requirements as User Stories

When writing user stories, how should agile teams handle non-functional requirements—desired ...

Nov 21, 2008 • 93 Comments [Read](#)

Epics, Features and User Stories

I've been getting more and more emails lately from people confused about the difference between ...

Nov 08, 2022 • 93 Comments [Read](#)

User stories are simple, short descriptions of desired functionality. Their power comes from the conversations they enable, allowing people to shift from writing about requirements to talking about them.

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

FEATURED POSTS

[Agile Estimating](#)

[Story Points Estimate Effort Not Just Complexity](#)

Getting started with...
[Planning Poker](#)

[What Are Agile Story Points?](#)

[How Can We Get the Best Estimates of Story Size?](#)

[3 Roles That Need to be Involved in Agile Estimating with Planning Poker](#)

[When Should We Estimate the Product Backlog](#)

[Get my chapters now!](#)

MORE ESTIMATING POSTS:

[What Are Agile Story Points?](#)

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

[Don't Equate Story Points to Hours](#)

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

[For Better Agile Planning, Be Collaborative](#)

The best plans are created by developers and stakeholders working together.

[Four Reasons Agile Teams Estimate Product Backlog Items](#)

Estimating product backlog items provides benefits beyond predicting when a project will be finished.

[Automatically Triangulating Estimates in Planning Poker](#)

Comparing estimates during Planning Poker just got easier.

[Estimating with Story Points Course Available for One Week](#)

Registrations are now open to Estimating with Story Points.

[Are People Problems Creating Estimating Problems?](#)

To create great estimates, you need to know that estimating is a human task, and humans can be complicated.

[Does the Perfect Estimate Exist? \(Free Video Training\)](#)

For a limited time, you can access free video training for creating estimates with story points.

[The Five Possible Estimates and Which One Your Team Should Use](#)

Unless team members have discussed it, they are almost certainly providing different types of estimates.

[Is It Dangerous to Calculate the Cost per Point?](#)

Calculating a cost per story point comes with risks. Here's what you need to know.

[Should You Re-Estimate Unfinished Stories?](#)

We avoid having unfinished work, but it sometimes happens. Here's what to do.

[Doors Now Open to Estimating With Story Points \(Plus 4 Bonuses\)](#)

Register for Estimating With Story Points before the deadline and get 4 FREE bonuses

[Watch Now: How to Stop People Problems from Hurting Your Estimates](#)

This is the final video in a short training series about story points. Available for a limited time only.

[Watch Now: What to Do When Teams and Stakeholders Want Perfect Estimates](#)

Now available: The second video in a series to help you coach your team on story points

[Watch Now: Training on Story Points for a Limited Time](#)

More than 5,900 agile professionals took this training! I'm making it available for free until ...

[Should a Team Assign Work During Sprint Planning?](#)

Some teams assign all tasks upfront. Others don't. Here's what works best.

Brand New Estimating with Story Points Video Course Now Available

Doors are now open (for a limited time only) to my new course: Estimating with Story Points.

VIDEO: Overcoming People Problems to Get Good Estimates

In this final video on creating estimates with story points, I help you overcome one of the biggest ...

VIDEO: Coaching Tips to Stop Teams Equating Points to Hours

Just released: Free videos to help teams solve problems when creating estimates with story points.

[Are We Really Bad at Estimating?](#)

It seems like most projects finish late. Here's why.

[Are Estimates Ever Helpful to Developers?](#)

Team members often think estimates are only useful for stakeholders and managers. Here's how they ...

[Time Pressure Improves Productivity & Quality...Up to a Point](#)

Time pressure improves productivity & quality...up to a point. Learn how short iterations and ...

[How to Estimate Story Points With Multiple Teams](#)

Establishing a common baseline allows multiple teams to estimate consistently with story points.

[Why the Fibonacci Sequence Works Well for Estimating](#)

If you've estimated with Planning Poker, you may very well have used cards with either the ...

[Story Point Estimates Are Best Thought of as Ranges](#)

When estimating in story points, teams should think in terms of ranges and rounding up. Here's why.

[Three Approaches to Estimating the Impact of Holidays and Time Off on Velocity](#)

Velocity can be great for predicting how much a team can deliver in a given period. But it needs to ...

[Get Your Free Personalized Guide to Agile](#)

Succeeding with agile isn't just about knowing where to start, it's about knowing where to go ...

[Why Agile Teams Should Estimate at Two Different Levels](#)

It's important for most agile teams to estimate both their product and sprint backlogs. But why?

[Nine Questions Scrum Masters and Product Owners Should Be Asking](#)

Scrum Masters and product owners can develop a habit of making statements rather than asking ...

[Why the Whole Team Should Participate When Estimating](#)

Even though not everyone may work on a product backlog item, it's still worth having the full team ...

[The Best Way to Establish a Baseline When Playing Planning Poker](#)

Planning Poker relies on relative estimating, in which the item being estimated is compared to one ...

[Don't Estimate the Sprint Backlog Using Task Points](#)

Some teams like story points so much, they invent task points and use those for sprint planning. ...

[How to Prevent Estimate Inflation](#)

A rose by any other name may smell as sweet, but a five-point story better not go by any other ...

[Should You Use Zero-Point Estimates on Your Product Backlog?](#)

I'm occasionally asked about the merits (and whether there are any) of putting an estimate of zero ...

[Change Isn't Free](#)

Agile teams are told to "embrace change," which is the subtitle to Kent Beck's wonderful Extreme ...

[Budget When You Can't Estimate](#)

I've written before that we should only estimate if having the estimate will change someone's ...

[Estimates on Split Stories Do Not Need to Equal the Original](#)

It is good practice to first write large user stories (commonly known as epics) and then to split ...

[Capacity-Driven Sprint Planning](#)

There are two primary ways for planning a sprint: velocity-driven sprint planning and ...

[The Main Benefit of Story Points](#)

If story points are an estimate of the time (effort) involved in doing something, why not just ...

[Story Points Are Still About Effort](#)

Story points are about time. There, I've said it, and can't be more clear than that. I've written ...

[Paying the Cost for More Precise Estimates](#)

I have no problem with a boss who asks a team to provide an estimate for how long a huge set of ...

[2 Times to Play Planning Poker and 1 Time Not To](#)

I recommend using Planning Poker on product backlog items rather than on the tasks that make up a ...

[3 Roles That Need to be Involved in Agile Estimating with Planning Poker](#)

All of a Scrum team's members should be present when playing Planning Poker. You may be tempted to ...

[Using Vertical Slicing and Estimation to Make Business Decisions at Adobe](#)

I recently helped to facilitate a two-day planning session for an important initiative at Adobe ...

[Assigning Story Points at the Right Time—Or Not at All](#)

As much as I value estimating the product backlog, not every team needs to do it. And those who do ...

[Scrum Shortcuts without Cutting Corners: Agile Tactics, Tools, & Tips](#)

Rather than a conventional review, here is the foreword I was asked to write for this ...

[Agile Planning and Project Management](#)

In this session we will shatter the myth that agile teams can't plan. We'll start by looking at the ...

[How to Estimate Velocity As an Agile Consultant](#)

Many of you work in a dedicated in-house team, but some of you contract with companies for Scrum ...

[When You Miss the Point of Sprint Planning Meetings](#)

In a recent interview for an upcoming agile book by Sondra Ashmore and Kristin Runyan, they asked ...

[How Can We Get the Best Estimates of Story Size?](#)

I was recently interviewed for an upcoming agile textbook written by Sondra Ashmore and Kristin ...

[Sprint Burndown Sums All Work Remaining](#)

A sprint burndown chart shows only one thing: How much work remains.

[Estimating with Tee Shirt Sizes](#)

I occasionally encounter the use of t-shirt sizes (Small, Medium, Large, or so on) in use as ...

[Overheard During a Customer Conversation About Estimates](#)

Estimating is a way of buying knowledge. If having the additional knowledge will lead to different ...

[Experiencing Agility From Requirements to Planning](#)

You know that before the development team writes their first line of code, they need a funded ...

[Succeeding with Agile](#)

Too many teams view planning as something to be avoided, and too many organizations view plans as ...

[Planning for Contract Agile Projects](#)

Maybe you work for a vendor who must bid for work in response to an RFP. Or perhaps your company ...

[Agile Estimating and Planning](#)

You've probably heard some people say, "Agile teams don't plan." Nothing could be further from the ...

[Agile Estimating](#)

Maybe you've heard about agile software development projects but aren't sure if they allow for the ...

[Points Are About Relative Effort Not Ranking](#)

Story points on an agile product backlog represent the effort to implement the backlog item.

[Estimating and Planning Are Necessary for Maximizing Delivered Value](#)

Planning is the act of thinking about the future. Sometimes that future holds risk and uncertainty. ...

[Recommendations, Not Rules](#)

I recommend that the team pick a day other than Mondays for starting their iterations. I recommend ...

[In Defense of Large Numbers](#)

Removing the large values from a deck of Planning Poker cards is like deciding to strike millions ...

[Seeing How Well a Team's Story Points Align from One to Eight](#)

The topic of how well a team estimates two point stories relative to one point stories (and so on) ...

[Estimating a Full Backlog Based on a Sample of It](#)

How do we estimate how many hours it will take to deliver a given product backlog if we have no ...

[Estimating Non-Functional Requirements](#)

Doing performance testing creates some amount of overhead on the team (the tax). This overhead or ...

[Should Story Points Be Assigned to a Bug Fixing Story?](#)

My usual recommendation is to assign points to bug fixing the agile defects. This really achieves ...

[The Problems with Estimating Business Value](#)

When we get features that are too small (as good user stories are), it is very difficult to put a ...

[Estimating Work Shared Between Two Backlog Items](#)

It would be nearly impossible to remove all dependencies between product backlog items and so our ...

[Story Points Estimate Effort Not Just Complexity](#)

This agile estimation blog post from Mike Cohn looks at story points. Learn why user story points ...

[Separate Estimating from Committing](#)

Remember the difference between an estimate and a commitment and keep the two activities separate, ...

[The Benefits of Feature Teams](#)

Moving away from component teams is a difficult but necessary step for those who want to adopt an ...

[How Do Story Points Relate to Hours?](#)

I'm often asked about the relationship between story points and hours. People who ask are usually ...

[Clarifying the Purpose of Iteration Planning](#)

Until I learn how to receive divine inspiration at the start of my iteration planning meetings, I'm ...

Interviewed by Software Process and Measurement Cast

[Is It a Good Idea to Establish a Common Baseline for Story Points?](#)

Some teams may respond to the pressure for their abstract measure of velocity to increase by ...

[Establishing a Common Baseline for Story Points](#)

A common criticism of story points is that the meaning of a story point will differ among teams.

[When Should We Estimate the Product Backlog](#)

Sprint planning meetings typically go into deeper detail than is appropriate for product backlog ...

[Why I Don't Use Story Points for Sprint Planning](#)

I don't use story points for sprint planning because story points are a useful long-term measure. ...

[Don't Average During Planning Poker](#)

While I want teams to come to agreement, I don't care how heartfelt the agreement is.

To Re-estimate or not; that is the question

When we estimate it is important that we not mix knowledge-before-the-fact with knowledge-after-the-fact.

Sprint and release planning should be in different units

In sprint planning the team should always talk of tasks and hours.

Writing Contracts for Agile Development

User stories are a great way to get people talking about requirements. However, there's a reason ...

[Estimating With Use Case Points](#)

Too much work goes into use cases to not employ them to their full potential. By assigning points ...

[Learn About Agile](#) [Agile & Scrum Training](#) [More...](#)

[New to Agile and Scrum?](#) [Online Certifications](#) [About Us](#)

[Scrum](#) [Video Courses](#) [Contact Us](#)

[User Stories](#) [In-Person Certifications](#) [Our Students](#)

[Planning Poker](#) [On-Site Courses](#) [Blog](#)

[Agile Software Development](#) [Training Schedule](#) [Books by Mike Cohn](#)

[Development](#) [Compare Courses](#) [Agile Tools](#)

[Agile Project Management](#) [PDUs and SEUs](#)

[Agile Presentations](#)

[Mountain Goat on YouTube](#)

[Agile Mentors Podcast](#)

[Elements of Agile](#)

[Assessment](#)

FEATURED POSTS

[Scrum Product Backlog](#)

[What Is a Product?](#)

[Product Backlog Refinement \(Grooming\)](#)

[Writing the Product Backlog Just in Time and Just Enough](#)

[Prioritizing Your Product Backlog](#)

[Why There Should Not Be a “Release Backlog”](#)

Get Your Free Scrum Cheat Sheet

Sign up below to receive this FREE reference.

First Name

Email Address

[Privacy Policy](#)

[Download my PDF](#)

MORE PRODUCT BACKLOG POSTS:

[What Are Agile Story Points?](#)

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

[Epics, Features and User Stories](#)

I've been getting more and more emails lately from people confused about the difference between ...

Relationship between Definition of Done and Conditions of Satisfaction

I'd like to clarify the relationship between two important concepts: a team's Definition of Done ...

Four Reasons Agile Teams Estimate Product Backlog Items

Estimating product backlog items provides benefits beyond predicting when a project will be finished.

Needs, Wants, and Wishes on Your Product Backlog

To prioritize a backlog, think about what users need, what they want, and what they merely wish for.

[How Detailed Should a User Story Be?](#)

Capturing too much or too little detail in a user story causes problems. Here's how to get it ...

[Agile Requirements Gathering: Three Types of Requirements](#)

Deadlines are often missed because teams fail to consider emergent requirements. Learn about the 3 ...

[Job Stories Offer a Viable Alternative to User Stories](#)

As useful as user stories can be, they've never been right for every team. An exciting alternative ...

[Story Point Estimates Are Best Thought of as Ranges](#)

When estimating in story points, teams should think in terms of ranges and rounding up. Here's why.

[Why the Three-Part User Story Template Works So Well](#)

The standard "As a...I...so that..." user story template has stood the test of time. Here's why each ...

[Four Steps to Keep Your Product Backlog Small and Manageable](#)

When a product backlog becomes too big, it hinders agility. Discover four steps your team can take ...

[Get Your Free Personalized Guide to Agile](#)

Succeeding with agile isn't just about knowing where to start, it's about knowing where to go ...

[Why Your Product Backlog Should Look Like an Iceberg](#)

An agile product backlog should evolve over time, with product backlog items and user stories ...

[My Most Popular Posts of 2016](#)

A summary of the most popular and engaging blog posts of 2016.

[What Is a Product?](#)

In this agile principles post from Mike Cohn, discover how Scrum organizations define products. ...

[The Dangers of a Definition of Ready](#)

This post explains how to use a Definition of Ready successfully and avoid it becoming a first step ...

[Can a Traditional SRS Be Converted into User Stories?](#)

A lot of traditionally managed projects begin with a Software Requirements Specification (SRS). ...

Should You Use Zero-Point Estimates on Your Product Backlog?

I'm occasionally asked about the merits (and whether there are any) of putting an estimate of zero ...

Who Can Add Items to the Product Backlog?

My wife, daughters and I use Wunderlist to manage our shared grocery list. Any time one of us ...

Handling Work Left at the End of a Sprint

It's quite common for a team to have a bit of unfinished work at the end of an agile sprint or ...

[Budget When You Can't Estimate](#)

I've written before that we should only estimate if having the estimate will change someone's ...

[Quickly Help Users Identify Their Needs](#)

Many projects bog down during requirements gathering.

[Not Everything Needs to Be a User Story: Using FDD Features](#)

User stories are great. When you've got users, that is. Sometimes, though, the users of a system or ...

[Product Backlog Refinement \(Grooming\)](#)

Learn about product backlog grooming: how, when, and why the agile team and product owner refine ...

[The Difference Between a Story and a Task](#)

What's the difference between a user story and a task? Well that's an easy question, I thought, the ...

[Focus on Benefits Rather than Features](#)

Suppose your boss has not bought into trying an agile...

[If it needs to happen: Schedule it!](#)

In the following guest post, Lisa Crispin argues the benefits of scheduling anything that's important.

[Why I Prefer Capacity-Driven Sprint Planning](#)

The problem with velocity-driven sprint planning is that velocity is simply too variable to be ...

[Velocity-Driven Sprint Planning](#)

There are two general approaches to planning sprints: velocity-driven planning and capacity-driven ...

[Handling Requests for Unnecessary Artifacts](#)

"Working software over comprehensive documentation." You've certainly seen...

Critiquing One of My Own Real User Stories

In case you haven't noticed, a few months ago we launched Front Row Agile, a site dedicated to ...

The Main Benefit of Story Points

If story points are an estimate of the time (effort) involved in doing something, why not just ...

Story Points Are Still About Effort

Story points are about time. There, I've said it, and can't be more clear than that. I've written ...

Now vs. Not-Now Prioritization Along with Medium-Term Goals

In last month's newsletter I wrote about how we make personal financial decisions in a now vs. ...

[Simplify Prioritization into "Now" and "Not Now"](#)

I think I'd like to buy a big-screen plasma television. And maybe after that, a new amplifier for ...

[Choose Backlog Items That Serve Two Purposes](#)

I've been playing a fair amount of Go lately. If you're not familiar with Go, it's a strategy game ...

[Adding Decorated User Roles to Your User Stories](#)

When writing user stories there are, of course, two parts: user and story. The user is embedded ...

[4 Tips for Spring Cleaning Your Product Backlog](#)

It's May, and we're well into spring now. If you're like me, you haven't yet done your annual ...

[Paying the Cost for More Precise Estimates](#)

I have no problem with a boss who asks a team to provide an estimate for how long a huge set of ...

[2 Times to Play Planning Poker and 1 Time Not To](#)

I recommend using Planning Poker on product backlog items rather than on the tasks that make up a ...

[Building a Product Users Want: From Idea to Backlog with the Vision Board](#)

Vision and Backlog

Scrum is a great framework for building a product with the right features. It ...

[3 Roles That Need to be Involved in Agile Estimating with Planning Poker](#)

All of a Scrum team's members should be present when playing Planning Poker. You may be tempted to ...

[Know Exactly What Velocity Means to Your Scrum Team](#)

To see how this applies to an agile project, consider the issue of whether a team should earn ...

[Using Vertical Slicing and Estimation to Make Business Decisions at Adobe](#)

I recently helped to facilitate a two-day planning session for an important initiative at Adobe ...

[New Year's Resolutions for ScrumMasters and Product Owners](#)

Happy New Year! It's resolution time. A ScrumMaster may want to resolve to praise the team more often.

[How to Be Sure You've Thought of Everything](#)

A common question I get is how can a product owner (or team) be sure they've thought...

[Assigning Story Points at the Right Time—Or Not at All](#)

As much as I value estimating the product backlog, not every team needs to do it. And those who do ...

[Product Backlog Bankruptcy!](#)

Humans are natural hoarders and that's why we often find ourselves in Backlog bedlam!

When You Miss the Point of Sprint Planning Meetings

In a recent interview for an upcoming agile book by Sondra Ashmore and Kristin Runyan, they asked ...

How Can We Get the Best Estimates of Story Size?

I was recently interviewed for an upcoming agile textbook written by Sondra Ashmore and Kristin ...

Backlog Refinement: Who Should Attend and How to Maximize Value

I was recently interviewed for an upcoming agile textbook written by Sondra Ashmore and Kristin ...

[Names Should Not Be Needed for User Stories](#)

I've never been a fan of naming user stories. Stories should generally be short enough that naming ...

[Three Tips for New ScrumMasters](#)

They say an ounce of prevention is worth a pound of cure—this adage definitely applies to new ...

[Using Scrum on an Analysis Project](#)

Just the right amount of urgency can generate greater creativity.

[Sprint Zero: A Good Idea or Not?](#)

One of the biggest problems with having a sprint zero is that it establishes a precedent that there ...

[The Product Owner in a Sprint Retrospective](#)

Is your agile team excluding the Scrum product owner from your sprint retrospectives? Check out ...

[Two Examples of Splitting Epics](#)

A Scrum trainer recently asked for a couple of good, real examples of large user stories (epics) ...

[Overheard During a Customer Conversation About Estimates](#)

Estimating is a way of buying knowledge. If having the additional knowledge will lead to different ...

[The PMI-ACP Exam: How to Pass On Your First Try](#)

I was looking forward to reading this book. I haven't paid much attention to the PMI-ACP initiative ...

[Handling Requirements from Architects Outside the Team](#)

An organization's Wise Architects often provide requirements to a team in the form of ...

[The Rules vs. The Generally Accepted Practices of Scrum](#)

Does a team have to work with user stories to do Scrum? Of course not.

[Experiencing Agility From Requirements to Planning](#)

You know that before the development team writes their first line of code, they need a funded ...

Prioritizing Your Product Backlog

The biggest risk to most projects is building the wrong product. Regardless of how fast your agile ...

GASPing About the Product Backlog

I've been wondering lately if Scrum is on the verge of getting a new standard meeting--the Backlog ...

[Points Are About Relative Effort Not Ranking](#)

Story points on an agile product backlog represent the effort to implement the backlog item.

[In Defense of Large Numbers](#)

Removing the large values from a deck of Planning Poker cards is like deciding to strike millions ...

[Estimating a Full Backlog Based on a Sample of It](#)

How do we estimate how many hours it will take to deliver a given product backlog if we have no ...

[A Sample Format for a Spreadsheet-Based Product Backlog](#)

I want to show a real easy way to put user stories in a spreadsheet-based product backlog. I wrote ...

[Estimating Non-Functional Requirements](#)

Doing performance testing creates some amount of overhead on the team (the tax). This overhead or ...

[A New Artifact - The Long-Term Product Backlog](#)

Left unattended, a product backlog can become large and hard to work with. If your backlog has ...

[Deciding What Kind of Projects are Most Suited for Agile](#)

Agile is most appropriate on any urgent project with significant complexity and novelty--and that ...

[Estimating Work Shared Between Two Backlog Items](#)

It would be nearly impossible to remove all dependencies between product backlog items and so our ...

[Story Points Estimate Effort Not Just Complexity](#)

This agile estimation blog post from Mike Cohn looks at story points. Learn why user story points ...

[Agile Teamwork](#)

High-performing Scrum teams have learned to do a little bit of everything during a sprint, thereby ...

[Agile Product Management with Scrum](#)

As a project management framework, Scrum introduces many changes. One of the biggest is the role of ...

[Distributed Teams: Build Trust through Early Progress](#)

Teams with subgroups formed around compatible skills, attitudes and approaches to work are less ...

[Remove Finish-to-Start Activities on Agile Projects](#)

With a little experience, most teams are able to see how to overlap some types of work and create ...

[Mix the Sizes of the Product Backlog Items You Commit To](#)

Scrum teams learn to work by doing a little of everything all the time.

[Make the Product Backlog DEEP](#)

A DEEP product backlog is detailed appropriately, estimated, emergent and prioritized.

[Agile Design: Intentional Yet Emergent](#)

The difference on a Scrum project is not that intentional design is thrown out, but that it is done ...

[How Do You Get from Here to Agile? Iterate.](#)

The effort of adopting Scrum is best managed using Scrum itself.

[Bugs on the Product Backlog](#)

The ideal situation is to put the bugs right onto the product backlog.

[Using a Task Board with One Remote Team Member](#)

What should we do in the case when the team really likes having a task board, but when one team ...

[The Ideal Agile Workspace](#)

Each person on the team should ideally be able to see each other person on the team.

[Why There Should Not Be a “Release Backlog”](#)

We've already overloaded the word *backlog* with product backlog and sprint backlog. Why confuse ...

[Clarifying the Purpose of Iteration Planning](#)

Until I learn how to receive divine inspiration at the start of my iteration planning meetings, I'm ...

[Should a Team Swarm on to One Backlog Item at a Time?](#)

Discover whether agile teams should work on one product backlog item at a time or if it's OK for ...

[Should the Daily Standup Be Person-by-Person or Story-by-Story?](#)

Most teams do the daily standup per-person, but some encounter the problem described here and ...

[Establishing a Common Baseline for Story Points](#)

A common criticism of story points is that the meaning of a story point will differ among teams.

[Rolling Lookahead Planning](#)

Rolling lookahead planning is a useful technique for large projects or any project with external ...

[Visualizing a Large Product Backlog With a Treemap](#)

Treemaps are an excellent way of visualizing large product backlogs.

[How To Fail With Agile](#)

Not everyone involved in an agile transition wants the change to be successful. This ...

[Improving On Traditional Release Burndown Charts](#)

By producing a single chart that shows both a team's rate of progress and the product backlog, we ...

[Working with "Storyless Tasks"](#)

A question I get frequently is what to do with tasks that do not belong to a particular user story ...

[Advantages of the "As a user, I want" user story template.](#)

In my user stories book and in all my training and conference sessions on user stories I advocate ...

Prioritizing Tasks Within a Sprint

When a team plans a sprint they make a commitment to complete the user stories they select from the ...

When Should We Estimate the Product Backlog

Sprint planning meetings typically go into deeper detail than is appropriate for product backlog ...

Writing the Product Backlog Just in Time and Just Enough

This article addresses the issue of how much detail should be included in product backlog items and ...

[Should Companies Measure Productivity in Story Points / Ideal Days?](#)

One measure we may want to include in our suite of metrics could be the responsiveness of the ...

[Why I Don't Use Story Points for Sprint Planning](#)

I don't use story points for sprint planning because story points are a useful long-term measure. ...

[OLDER POSTS](#)

Learn About Agile

New to Agile and Scrum?

Agile & Scrum Training

Online Certifications

More...

About Us

Scrum

Video Courses

Contact Us

User Stories

In-Person Certifications

Our Students

Planning Poker

On-Site Courses

Blog

Agile Software Development

Training Schedule

Books by Mike Cohn

Agile Project Management

Compare Courses

Agile Tools

Agile Presentations

PDUs and SEUs

Mountain Goat on YouTube

Agile Mentors Podcast

Elements of Agile

Assessment

FEATURED POSTS

[Product Owner Role and Responsibilities](#)

[Presentation for Product Owners: Storytelling with the Prioritized Product Backlog](#)

[Incorporating Learning and Expected Cost of Change](#)

[Can a Product Owner Dictate the Architecture?](#)

[The Product Owner in a Sprint Retrospective](#)

[Want Better Software? Just Ask](#)

Get Your Free Scrum Cheat Sheet

Sign up below to receive this FREE reference.

First Name

Email Address

[Privacy Policy](#)

MORE PRODUCT OWNER POSTS:

[Be a Great Product Owner: Six Things Teams and Scrum Masters Need](#)

Learn six ways effective product owners ensure their teams' success.

[The Product Owner's Second Team](#)

Successful product owners will see their stakeholders as a team, not merely a set of individuals.

[Top 5 changes in the 2020 version of the Scrum Guide](#)

Recently the 2020 version of the Scrum Guide was released. What changes were made that you need to ...

[\[VIDEO\] What does a virtual certified scrum course look like?](#)

Our virtual Certified Scrum courses aren't tedious 16-hour Zoom snoozefests. This video shows what ...

[4 Steps to Persuade a Product Owner to Prioritize Refactoring](#)

Teams often struggle to persuade their product owners to prioritize refactoring. This simple ...

[What I Want for an Agile Christmas](#)

It's coming on Christmas. That time of year when I drop hints to my family about gifts I'd like.

[Is It Time to Do Away with Scrum's Product Owner Role](#)

Scrum has only three official roles: Scrum Master, product owner, and team member. No distinction ...

[Get Your Free Personalized Guide to Agile](#)

Succeeding with agile isn't just about knowing where to start, it's about knowing where to go ...

[Why Your Product Backlog Should Look Like an Iceberg](#)

An agile product backlog should evolve over time, with product backlog items and user stories ...

[Six Guidelines for Saying No to a Stakeholder](#)

Telling a stakeholder you can't work on their feature is difficult. Here are ways to make that ...

[How to Ensure You're Working on the Most Important Items Each Iteration](#)

Product owners often sacrifice progress toward important goals to put out short-term fires. There's ...

[Nine Questions Scrum Masters and Product Owners Should Be Asking](#)

Scrum Masters and product owners can develop a habit of making statements rather than asking ...

[The Chief Product Owner on Large Agile Projects](#)

Advice on scaling agile projects by adding a chief product owner and sharing responsibility across ...

[Strategize](#)

Most agile processes are empty of any advice on forming a company or product strategy. Product ...

[Large-Scale Scrum](#)

"Large-Scale Scrum" by Craig Larman and Bas Vodde is great for anyone looking to scale Scrum up to ...

[The Sprint Review as a Sign-Off Meeting](#)

Some teams use the sprint review as a time for product owners or key stakeholders to formally ...

[What Is a Product?](#)

In this agile principles post from Mike Cohn, discover how Scrum organizations define products. ...

Who Can Add Items to the Product Backlog?

My wife, daughters and I use Wunderlist to manage our shared grocery list. Any time one of us ...

Budget When You Can't Estimate

I've written before that we should only estimate if having the estimate will change someone's ...

An Iterative Waterfall Isn't Agile

I've noticed something disturbing over the past two years. And it's occurred uniformly with teams ...

Prioritize and Optimize Over a Slightly Longer Horizon

A lot of agile literature stresses that product owners must prioritize the delivery of value. I'm ...

Can a Product Owner Dictate the Architecture?

In general, a Scrum product owner's job is to specify what to build, not how to build it. Read this ...

[Product Backlog Refinement \(Grooming\)](#)

Learn about product backlog grooming: how, when, and why the agile team and product owner refine ...

[5 Reasons Product Owners Should Let Teams Work Out of Order](#)

A product owner hands 10 story cards to the team. The team reads them and hands the fifth and sixth ...

[Who Picks the Sprint Length on a Scrum Team?](#)

Of course, the answer is the whole team – that collective of ScrumMaster plus product owner plus ...

[ScrumMasters Should Not Also Be Product Owners](#)

Hey, ScrumMaster: Step away from the index cards! You should not be a team's product owner if you ...

[Capacity-Driven Sprint Planning](#)

There are two primary ways for planning a sprint: velocity-driven sprint planning and ...

[Handling Requests for Unnecessary Artifacts](#)

"Working software over comprehensive documentation." You've certainly seen...

[The Agile Household: How Scrum Made Us a Better Family](#)

Martin Lapointe shares how he and his family used Scrum to manage their recent relocation from ...

[Now vs. Not-Now Prioritization Along with Medium-Term Goals](#)

In last month's newsletter I wrote about how we make personal financial decisions in a now vs. ...

[Ray Bradbury on the Benefits of Short Releases](#)

In 2001, author Ray Bradbury gave a talk at the annual Writer's Symposium by the Sea in San Diego. ...

[Simplify Prioritization into "Now" and "Not Now"](#)

I think I'd like to buy a big-screen plasma television. And maybe after that, a new amplifier for ...

[Choose Backlog Items That Serve Two Purposes](#)

I've been playing a fair amount of Go lately. If you're not familiar with Go, it's a strategy game ...

[Teams Should Go So Fast They Almost Spin Out of Control](#)

Yes, I really did refer to guitarist Alvin Lee in a Certified Scrum Product Owner class last week. ...

[4 Tips for Spring Cleaning Your Product Backlog](#)

It's May, and we're well into spring now. If you're like me, you haven't yet done your annual ...

[Introduction to Scrum PPT](#)

You may have heard Scrum is one of the leading agile software development processes. With more than ...

[Schedule vs. Cost: The Tradeoff in Agile](#)

Just because a shorter schedule is more important most of the time, does not mean it is more ...

[4 Reasons to Include Developers in Story Writing](#)

Participants in my Certified ScrumMaster courses are often surprised when I recommend that ...

[2 Times to Play Planning Poker and 1 Time Not To](#)

I recommend using Planning Poker on product backlog items rather than on the tasks that make up a ...

[Building a Product Users Want: From Idea to Backlog with the Vision Board](#)

Vision and Backlog

Scrum is a great framework for building a product with the right features. It ...

[3 Roles That Need to be Involved in Agile Estimating with Planning Poker](#)

All of a Scrum team's members should be present when playing Planning Poker. You may be tempted to ...

[Know Exactly What Velocity Means to Your Scrum Team](#)

To see how this applies to an agile project, consider the issue of whether a team should earn ...

[Making the Decision to Abnormally Terminate a Sprint](#)

It's always good to have a Plan B. And all Scrum teams do--it's called an abnormal termination.

An ...

[Six Times Two Plus One Equals a Good Project Cadence](#)

In last month's newsletter I wrote about the idea that everything happens within a sprint. There is ...

[Using Vertical Slicing and Estimation to Make Business Decisions at Adobe](#)

I recently helped to facilitate a two-day planning session for an important initiative at Adobe ...

New Year's Resolutions for ScrumMasters and Product Owners

Happy New Year! It's resolution time. A ScrumMaster may want to resolve to praise the team more often.

[How to Be Sure You've Thought of Everything](#)

A common question I get is how can a product owner (or team) be sure they've thought...

[Assigning Story Points at the Right Time—Or Not at All](#)

As much as I value estimating the product backlog, not every team needs to do it. And those who do ...

[Product Backlog Bankruptcy!](#)

Humans are natural hoarders and that's why we often find ourselves in Backlog bedlam!

[One of a Kind Beats Three of a Kind](#)

I worked with a team last week that had three business people who each established the team's ...

[Providing Feedback to Team Members](#)

Even if you've taken the often stated agile stance of getting rid of periodic performance reviews ...

[Three Tips for New ScrumMasters](#)

They say an ounce of prevention is worth a pound of cure—this adage definitely applies to new ...

[Sprint Zero: A Good Idea or Not?](#)

One of the biggest problems with having a sprint zero is that it establishes a precedent that there ...

[The Product Owner in a Sprint Retrospective](#)

Is your agile team excluding the Scrum product owner from your sprint retrospectives? Check out ...

[Selecting the Right User Role](#)

It is sometimes beneficial to write stories for someone other than the user.

[Just Start: Take Action, Embrace Uncertainty, Create the Future](#)

I've never been a big fan of the Shewhart or Deming cycle of Plan-Do-Check-Act. Sure, it works fine ...

[Essential Scrum: A Practical Guide to the Most Popular Agile Process](#)

The book is a comprehensive overview of Scrum. It goes from the principles of agile through the ...

[Handling Requirements from Architects Outside the Team](#)

An organization's Wise Architects often provide requirements to a team in the form of ...

[The Rules vs. The Generally Accepted Practices of Scrum](#)

Does a team have to work with user stories to do Scrum? Of course not.

[Agile Product Management](#)

The vast majority of what has been written about agile processes is intended for programmers and ...

[Project Economics](#)

Building the right product at the right time is just as important as building a quality product. ...

[Incorporating Learning and Expected Cost of Change](#)

Product owners are typically asked to prioritize based on the nebulous term "business value." But ...

[Presentation for Product Owners: Storytelling with the Prioritized Product Backlog](#)

The vast majority of what has been written about agile processes is intended for programmers and ...

[Estimating and Planning Are Necessary for Maximizing Delivered Value](#)

Planning is the act of thinking about the future. Sometimes that future holds risk and uncertainty. ...

[Please Help Me List the Problems with Using Agile or Scrum](#)

I'm trying to create a list of the biggest, most common, or hardest to overcome agile problems that ...

[Protecting the Team Cuts Both Ways](#)

Scrum Masters need to protect agile teams from more than just overly aggressive product owners. ...

[Estimating Non-Functional Requirements](#)

Doing performance testing creates some amount of overhead on the team (the tax). This overhead or ...

[A New Artifact - The Long-Term Product Backlog](#)

Left unattended, a product backlog can become large and hard to work with. If your backlog has ...

[Should Story Points Be Assigned to a Bug Fixing Story?](#)

My usual recommendation is to assign points to bug fixing the agile defects. This really achieves ...

[Avast Combining the ScrumMaster and Product Owner, Matey!](#)

A common question is whether it's acceptable to combine the role of product and ScrumMaster and ...

[The Roles of the Project Management Office in Scrum](#)

A project management office (PMO) that is engaged in and supportive of transitioning to Scrum can ...

[Estimating Work Shared Between Two Backlog Items](#)

It would be nearly impossible to remove all dependencies between product backlog items and so our ...

[Agile Teamwork](#)

High-performing Scrum teams have learned to do a little bit of everything during a sprint, thereby ...

[Agile Product Management with Scrum](#)

As a project management framework, Scrum introduces many changes. One of the biggest is the role of ...

[Distributed Teams: Build Trust through Early Progress](#)

Teams with subgroups formed around compatible skills, attitudes and approaches to work are less ...

[Removing Team Members](#)

People often ask me whether teams should have the right to vote members off. To help answer that ...

[Make the Product Backlog DEEP](#)

A DEEP product backlog is detailed appropriately, estimated, emergent and prioritized.

[Agile Design: Intentional Yet Emergent](#)

The difference on a Scrum project is not that intentional design is thrown out, but that it is done ...

[Bugs on the Product Backlog](#)

The ideal situation is to put the bugs right onto the product backlog.

[The Ideal Agile Workspace](#)

Each person on the team should ideally be able to see each other person on the team.

[Why There Should Not Be a “Release Backlog”](#)

We've already overloaded the word *backlog* with product backlog and sprint backlog. Why confuse ...

[Non-functional Requirements as User Stories](#)

When writing user stories, how should agile teams handle non-functional requirements—desired ...

[Rolling Lookahead Planning](#)

Rolling lookahead planning is a useful technique for large projects or any project with external ...

[Visualizing a Large Product Backlog With a Treemap](#)

Treemaps are an excellent way of visualizing large product backlogs.

[How To Fail With Agile](#)

Not everyone involved in an agile transition wants the change to be successful. This ...

[Improving On Traditional Release Burndown Charts](#)

By producing a single chart that shows both a team's rate of progress and the product backlog, we ...

[Advantages of the “As a user, I want” user story template.](#)

In my user stories book and in all my training and conference sessions on user stories I advocate ...

[Prioritizing Tasks Within a Sprint](#)

When a team plans a sprint they make a commitment to complete the user stories they select from the ...

[When Should We Estimate the Product Backlog](#)

Sprint planning meetings typically go into deeper detail than is appropriate for product backlog ...

[Do Products Owners Evolve As a Species?](#)

What my friend had found is that his product owners had evolved in adaptation to their environment, ...

[Don't Average During Planning Poker](#)

While I want teams to come to agreement, I don't care how heartfelt the agreement is.

[Advice on Conducting the Scrum of Scrums Meeting](#)

The scrum of scrums meeting is an important technique in scaling Scrum to large project teams. ...

[ScrumMaster: Appointed or Team-Selected?](#)

The selection of a new Scrum team's ScrumMaster can impact the success or failure of the team's ...

[Innovation Games](#)

One of the challenges in new product innovation is that the process cannot be broken down into a ...

[Sprint Planning](#)

Many teams try to divide and conquer when it comes to sprint planning, often with disjointed and ...

[The Role of Learning and Expected Cost of Change](#)

An academic paper that describes the importance of using more than just the vaguely defined ...

[Incorporating Learning and Expected Cost of Change](#)

An experience report presented at XP2006 covering why it is not as simple as telling product owners ...

[Change Is Good...Or is It?](#)

Change may be a constant, but it doesn't have to be constant. By following some simple guidelines, ...

[Want Better Software? Just Ask](#)

This article presents very specific advice on seven things a product owner or customer can do to ...

[Toward a Catalog of Scrum Smells](#)

This article was written for the Scrum Alliance soapbox. It presents an initial collection of Scrum ...

OLDER POSTS

[Learn About Agile](#)

- [New to Agile and Scrum?](#)
- [Scrum](#)
- [User Stories](#)
- [Planning Poker](#)
- [Agile Software Development](#)
- [Agile Project Management](#)
- [Agile Presentations](#)
- [Mountain Goat on YouTube](#)
- [Agile Mentors Podcast](#)
- [Elements of Agile Assessment](#)

[Agile & Scrum Training](#)

- [Online Certifications](#)
- [Video Courses](#)
- [In-Person Certifications](#)
- [On-Site Courses](#)
- [Training Schedule](#)
- [Compare Courses](#)
- [PDUs and SEUs](#)

[More...](#)

- [About Us](#)
- [Contact Us](#)
- [Our Students](#)
- [Blog](#)
- [Books by Mike Cohn](#)
- [Agile Tools](#)

FEATURED POSTS

Getting started with...

[The Scrum Framework](#)

[A Regular Heartbeat](#)

[Don't Take Partial Credit for Semi-Finished Stories](#)

[Selecting the Right Iteration Length](#)

[Six Times Two Plus One Equals a Good Project Cadence](#)

[Synchronize Rather Than Overlap Sprints](#)

[Agile Needs to Be Both Iterative and Incremental](#)

[Take the Assessment](#)

MORE SPRINTS POSTS:

[How Implementation Intentions Help My Sprints](#)

Planning an exact time to do something within a sprint helps make sure you get the important stuff ...

[The Goal of Sprint Planning](#)

Sprint planning may look like it's about tasks and estimates but those are not the goal.

[Learn About Agile](#)

[New to Agile and Scrum?](#)

[Agile & Scrum Training](#)

[Online Certifications](#)

[More...](#)

[About Us](#)

[Scrum](#)

[Video Courses](#)

[Contact Us](#)

[User Stories](#)

[In-Person Certifications](#)

[Our Students](#)

[Planning Poker](#)

[On-Site Courses](#)

[Blog](#)

[Agile Software](#)

[Training Schedule](#)

[Books by Mike Cohn](#)

[Development](#)

[Compare Courses](#)

[Agile Tools](#)

[Agile Project Management](#)

[PDUs and SEUs](#)

[Agile Presentations](#)

[Mountain Goat on YouTube](#)

[Agile Mentors Podcast](#)

[Elements of Agile](#)

[Assessment](#)

FEATURED POSTS

Getting started with...

User Stories

[Introduction to User Stories](#)

[Non-functional Requirements as User Stories](#)

[Epics, Features and User Stories](#)

[Get 200 Real Life User Stories](#)

[Examples Written by Mike Cohn](#)

[Get Your Free Scrum Cheat Sheet](#)

See user stories Mike Cohn wrote as part of several real product backlogs.

First Name

Sign up below to receive this FREE reference.

First Name

Email Address

Email Address

[Privacy Policy](#)

[Privacy Policy](#)

[Download my PDF](#)

MORE USER STORIES POSTS:

[User Stories: How to Create Story Maps](#)

Story maps help to create a shared understanding of the product, visualize user needs, and elicit ...

[How to Run a Successful User Story Writing Workshop](#)

What you need to know to conduct a story-writing workshop with your team.

I've been getting more and more emails lately from people confused about the difference between ...

[Advantages of User Stories over Requirements and Use Cases](#)

At the surface, user stories appear to have much in common with use cases and traditional ...

[Relationship between Definition of Done and Conditions of Satisfaction](#)

I'd like to clarify the relationship between two important concepts: a team's Definition of Done ...

[How Detailed Should a User Story Be?](#)

Capturing too much or too little detail in a user story causes problems. Here's how to get it ...

[Job Stories Offer a Viable Alternative to User Stories](#)

As useful as user stories can be, they've never been right for every team. An exciting alternative ...

[Why the Three-Part User Story Template Works So Well](#)

The standard "As a....I...so that..." user story template has stood the test of time. Here's why each ...

[Get Your Free Personalized Guide to Agile](#)

Succeeding with agile isn't just about knowing where to start, it's about knowing where to go ...

[How to Work with Complex User Stories That Cannot Be Split](#)

Here's what to do when facing a complex user story that cannot be split but is too large for one ...

[Video Training: How to Split User Stories](#)

End your agile team's struggle with story splitting. Discover this free video training and learn 5 ...

[Five Story-Splitting Mistakes and How to Stop Making Them](#)

There are plenty of mistakes a team can make when splitting user stories. Here are five of the most ...

[The Two Ways to Add Detail to User Stories](#)

User stories can be deliberately vague at first but detail needs to be added eventually. There are ...

[How Programmers and Testers \(and Others\) Should Collaborate on User Stories](#)

What do the testers do at the start of a sprint when there's nothing to test? That problem is ...

[Five Simple but Powerful Ways to Split User Stories](#)

Splitting user stories is hard. Here are the only five techniques you need to be able to split ...

[Advice on How to Split Reporting User Stories](#)

Splitting stories has long been one of the biggest challenges facing agile teams. Here are some ...

[Can a Traditional SRS Be Converted into User Stories?](#)

A lot of traditionally managed projects begin with a Software Requirements Specification (SRS). ...

[Who Can Add Items to the Product Backlog?](#)

My wife, daughters and I use Wunderlist to manage our shared grocery list. Any time one of us ...

[An Iterative Waterfall Isn't Agile](#)

I've noticed something disturbing over the past two years. And it's occurred uniformly with teams ...

[Quickly Help Users Identify Their Needs](#)

Many projects bog down during requirements gathering.

[Estimates on Split Stories Do Not Need to Equal the Original](#)

It is good practice to first write large user stories (commonly known as epics) and then to split ...

[Not Everything Needs to Be a User Story: Using FDD Features](#)

User stories are great. When you've got users, that is. Sometimes, though, the users of a system or ...

[The Difference Between a Story and a Task](#)

What's the difference between a user story and a task? Well that's an easy question, I thought, the ...

[Critiquing One of My Own Real User Stories](#)

In case you haven't noticed, a few months ago we launched Front Row Agile, a site dedicated to ...

[Now vs. Not-Now Prioritization Along with Medium-Term Goals](#)

In last month's newsletter I wrote about how we make personal financial decisions in a now vs. ...

[Ray Bradbury on the Benefits of Short Releases](#)

In 2001, author Ray Bradbury gave a talk at the annual Writer's Symposium by the Sea in San Diego. ...

[Choose Backlog Items That Serve Two Purposes](#)

I've been playing a fair amount of Go lately. If you're not familiar with Go, it's a strategy game ...

[Adding Decorated User Roles to Your User Stories](#)

When writing user stories there are, of course, two parts: user and story. The user is embedded ...

[4 Tips for Spring Cleaning Your Product Backlog](#)

It's May, and we're well into spring now. If you're like me, you haven't yet done your annual ...

4 Reasons to Include Developers in Story Writing

Participants in my Certified ScrumMaster courses are often surprised when I recommend that ...

2 Times to Play Planning Poker and 1 Time Not To

I recommend using Planning Poker on product backlog items rather than on the tasks that make up a ...

[Building a Product Users Want: From Idea to Backlog with the Vision Board](#)

Vision and Backlog

Scrum is a great framework for building a product with the right features. It ...

[Agile Planning and Project Management](#)

In this session we will shatter the myth that agile teams can't plan. We'll start by looking at the ...

[How to Estimate Velocity As an Agile Consultant](#)

Many of you work in a dedicated in-house team, but some of you contract with companies for Scrum ...

[Backlog Refinement: Who Should Attend and How to Maximize Value](#)

I was recently interviewed for an upcoming agile textbook written by Sondra Ashmore and Kristin ...

[Names Should Not Be Needed for User Stories](#)

I've never been a fan of naming user stories. Stories should generally be short enough that naming ...

[Hard to Read Handwriting Is Best for User Stories](#)

I've always been envious of those with nice handwriting or even distinctive printing. I've always ...

Selecting the Right User Role

It is sometimes beneficial to write stories for someone other than the user.

Two Examples of Splitting Epics

A Scrum trainer recently asked for a couple of good, real examples of large user stories (epics) ...

[Scrum and Project Governance](#)

Unfortunately, in many companies, we see an inconsistency between project governance and project management.

[The Rules vs. The Generally Accepted Practices of Scrum](#)

Does a team have to work with user stories to do Scrum? Of course not.

[Agile Product Management](#)

The vast majority of what has been written about agile processes is intended for programmers and ...

[Experiencing Agility From Requirements to Planning](#)

You know that before the development team writes their first line of code, they need a funded ...

[Planning for Contract Agile Projects](#)

Maybe you work for a vendor who must bid for work in response to an RFP. Or perhaps your company ...

[Introduction to User Stories](#)

The technique of expressing requirements as user stories is one of the most broadly applicable ...

[Presentation for Product Owners: Storytelling with the Prioritized Product Backlog](#)

The vast majority of what has been written about agile processes is intended for programmers and ...

[Estimating and Planning Are Necessary for Maximizing Delivered Value](#)

Planning is the act of thinking about the future. Sometimes that future holds risk and uncertainty. ...

[Recommendations, Not Rules](#)

I recommend that the team pick a day other than Mondays for starting their iterations. I recommend ...

[In Defense of Large Numbers](#)

Removing the large values from a deck of Planning Poker cards is like deciding to strike millions ...

[Seeing How Well a Team's Story Points Align from One to Eight](#)

The topic of how well a team estimates two point stories relative to one point stories (and so on) ...

[Estimating a Full Backlog Based on a Sample of It](#)

How do we estimate how many hours it will take to deliver a given product backlog if we have no ...

[A Sample Format for a Spreadsheet-Based Product Backlog](#)

I want to show a real easy way to put user stories in a spreadsheet-based product backlog. I wrote ...

[Estimating Non-Functional Requirements](#)

Doing performance testing creates some amount of overhead on the team (the tax). This overhead or ...

A New Artifact - The Long-Term Product Backlog

Left unattended, a product backlog can become large and hard to work with. If your backlog has ...

The Problems with Estimating Business Value

When we get features that are too small (as good user stories are), it is very difficult to put a ...

The Art of Compromise

The concepts of agility and project governance are not fundamentally opposed. Each is an attempt to ...

[Make the Product Backlog DEEP](#)

A DEEP product backlog is detailed appropriately, estimated, emergent and prioritized.

[Bugs on the Product Backlog](#)

The ideal situation is to put the bugs right onto the product backlog.

[Why There Should Not Be a “Release Backlog”](#)

We've already overloaded the word *backlog* with product backlog and sprint backlog. Why confuse ...

[How Do Story Points Relate to Hours?](#)

I'm often asked about the relationship between story points and hours. People who ask are usually ...

[Non-functional Requirements as User Stories](#)

When writing user stories, how should agile teams handle non-functional requirements—desired ...

[Should the Daily Standup Be Person-by-Person or Story-by-Story?](#)

Most teams do the daily standup per-person, but some encounter the problem described here and ...

[Improving On Traditional Release Burndown Charts](#)

By producing a single chart that shows both a team's rate of progress and the product backlog, we ...

[Advantages of the “As a user, I want” user story template.](#)

In my user stories book and in all my training and conference sessions on user stories I advocate ...

[Prioritizing Tasks Within a Sprint](#)

When a team plans a sprint they make a commitment to complete the user stories they select from the ...

[When Should We Estimate the Product Backlog](#)

Sprint planning meetings typically go into deeper detail than is appropriate for product backlog ...

[Writing User Stories for Back-end Systems](#)

I was asked recently how to go about writing user stories for a back-end financial system. This is ...

[Writing the Product Backlog Just in Time and Just Enough](#)

This article addresses the issue of how much detail should be included in product backlog items and ...

[Don't Average During Planning Poker](#)

While I want teams to come to agreement, I don't care how heartfelt the agreement is.

To Re-estimate or not; that is the question

When we estimate it is important that we not mix knowledge-before-the-fact with knowledge-after-the-fact.

Programmers and Testers Can Work on Things Smaller Than User Stories

A common misperception is that testers cannot do any work during an iteration until the programmers ...

Sprint and release planning should be in different units

In sprint planning the team should always talk of tasks and hours.

[Writing Contracts for Agile Development](#)

User stories are a great way to get people talking about requirements. However, there's a reason ...

[The Role of Learning and Expected Cost of Change](#)

An academic paper that describes the importance of using more than just the vaguely defined ...

[Incorporating Learning and Expected Cost of Change](#)

An experience report presented at XP2006 covering why it is not as simple as telling product owners ...

I Didn't Know I Needed That!

Products that do everything they're supposed to do *and* offer consumers something they like, but ...

Estimating With Use Case Points

Too much work goes into use cases to not employ them to their full potential. By assigning points ...

[What's Holding You Back?](#)

I was honored to be the guest editor of a special issue of *Better Software* magazine that was ...

[Writing User Stories - Questioning Your Users](#)

Let's face it, most people don't know what they want. Most of us just

[Want Better Software? Just Ask](#)

This article presents very specific advice on seven things a product owner or customer can do to ...

[Introducing An Agile Process to an Organization](#)

The transition from a plan-driven to an agile process affects not only the development team ...

[The Upside of Downsizing](#)

This article describes how a project was successfully downsized from 100 to 12 developers. To make ...

New to Agile and Scrum? [Online Certifications](#) [About Us](#)

[Scrum](#) [Video Courses](#) [Contact Us](#)

[User Stories](#) [In-Person Certifications](#) [Our Students](#)

[Planning Poker](#) [On-Site Courses](#) [Blog](#)

[Agile Software Development](#) [Training Schedule](#) [Books by Mike Cohn](#)

[Agile Project Management](#) [Compare Courses](#) [Agile Tools](#)

[Agile Presentations](#) [PDUs and SEUs](#)

[Mountain Goat on YouTube](#)

[Agile Mentors Podcast](#)

[Elements of Agile](#)

[Assessment](#)

Agile and Scrum Training

We've helped **29,036** people succeed with agile. Let us help find the training right for you.

Certified ScrumMaster®	Certified Scrum Owner®	Advanced Product Owner®	Advanced Scrum Master®	Estimating Story Points	Better Stories	Agile Planning	Scrum Guide	Scrum Repair	Let Go of Agile	Succeeding with Agile
Our Certified Scrum Master training is a two-day in-person course.	If you're a Certified ScrumMaster® with a Product Owner® experience, take this two-day course.	This course is stand-out from others, writing ideal user stories for anyone who wants to join the ranks.	Stand out from the crowd by writing ideal user stories for anyone who wants to join the ranks.	Overcome the challenge of writing user stories to gain a solid understanding of access and how to deliver.	This agile training shows you how to go about solving problems.	The perfect introduction to Scrum. You'll learn how to overcome the most common challenges with it's online troubleshooting course.	ScrumMasters and their teams will learn how to overcome the key roles, ceremonies, and vexing artifacts, Scrum and challenges with this online troubleshooting course.	Watch Scrum. You'll get an overview of the key learnings, common ceremonies, and vexing artifacts, Scrum and challenges with this online troubleshooting course.	Watch Mike Cohn's popular conference keynote session and learn how to help an organization move beyond initial success to sustained agility.	
Our Certified Scrum Master training is a two-day in-person course.	If you're a Certified ScrumMaster® with a Product Owner® experience, take this two-day course.	This course is stand-out from others, writing ideal user stories for anyone who wants to join the ranks.	Stand out from the crowd by writing ideal user stories for anyone who wants to join the ranks.	Overcome the challenge of writing user stories to gain a solid understanding of access and how to deliver.	This agile training shows you how to go about solving problems.	The perfect introduction to Scrum. You'll learn how to overcome the key roles, ceremonies, and vexing artifacts, Scrum and challenges with this online troubleshooting course.	ScrumMasters and their teams will learn how to overcome the key roles, ceremonies, and vexing artifacts, Scrum and challenges with this online troubleshooting course.	Watch Scrum. You'll get an overview of the key learnings, common ceremonies, and vexing artifacts, Scrum and challenges with this online troubleshooting course.	Watch Mike Cohn's popular conference keynote session and learn how to help an organization move beyond initial success to sustained agility.	
Our Certified Scrum Master training is a two-day in-person course.	If you're a Certified ScrumMaster® with a Product Owner® experience, take this two-day course.	This course is stand-out from others, writing ideal user stories for anyone who wants to join the ranks.	Stand out from the crowd by writing ideal user stories for anyone who wants to join the ranks.	Overcome the challenge of writing user stories to gain a solid understanding of access and how to deliver.	This agile training shows you how to go about solving problems.	The perfect introduction to Scrum. You'll learn how to overcome the key roles, ceremonies, and vexing artifacts, Scrum and challenges with this online troubleshooting course.	ScrumMasters and their teams will learn how to overcome the key roles, ceremonies, and vexing artifacts, Scrum and challenges with this online troubleshooting course.	Watch Scrum. You'll get an overview of the key learnings, common ceremonies, and vexing artifacts, Scrum and challenges with this online troubleshooting course.	Watch Mike Cohn's popular conference keynote session and learn how to help an organization move beyond initial success to sustained agility.	
Our Certified Scrum Master training is a two-day in-person course.	If you're a Certified ScrumMaster® with a Product Owner® experience, take this two-day course.	This course is stand-out from others, writing ideal user stories for anyone who wants to join the ranks.	Stand out from the crowd by writing ideal user stories for anyone who wants to join the ranks.	Overcome the challenge of writing user stories to gain a solid understanding of access and how to deliver.	This agile training shows you how to go about solving problems.	The perfect introduction to Scrum. You'll learn how to overcome the key roles, ceremonies, and vexing artifacts, Scrum and challenges with this online troubleshooting course.	ScrumMasters and their teams will learn how to overcome the key roles, ceremonies, and vexing artifacts, Scrum and challenges with this online troubleshooting course.	Watch Scrum. You'll get an overview of the key learnings, common ceremonies, and vexing artifacts, Scrum and challenges with this online troubleshooting course.	Watch Mike Cohn's popular conference keynote session and learn how to help an organization move beyond initial success to sustained agility.	

Scrum, the group fixing
but Scrum coaching teams
also product calls that
gives backlog included have
participants as after developed
hands- a the bad
on tool course. habits.
experience. for
project
success.

The Agile Mentors Community is a place for agile practitioners around the world to collaborate in a safe, experiential learning space. In this private community, you will find lively discussions on a variety of agile topics such as agile leadership, user stories and the product backlog, transitioning to agile, and more.

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		

by
Mike Cohn **Tagged:**
sprint planning, sprint goals
57 Comments

Not every team benefits from a sprint goal.

Blasphemy? Perhaps.

But stay with me for a moment. I want to talk about what sprint goals are and how Scrum teams benefit from them. Then, I can explain why I still don't insist on them for every Scrum team.

What Is a Sprint Goal?

What is a sprint goal in agile processes like Scrum? The sprint goal is defined as the single objective for the sprint. A sprint goal is created and finalized by the entire [Scrum team](#) ([Scrum Master](#), [product owner](#) and developers) during [sprint planning](#), and helps communicate why the sprint is valuable to [stakeholders](#).

Focusing on a single objective is a great idea. Development teams often benefit from the focus that a sprint goal provides.

Why Do Developers Need Sprint Goals?

The purpose of the sprint goal is to focus the team's attention on what most needs to be achieved. I remember how vital this was in the late 1990s, the early days of [Scrum](#).

Back then all teams did four-week sprints, no one did what we today call [backlog refinement](#), and Extreme Programming hadn't yet introduced the world to [user stories](#).

Back then, teams would lose sight of what they were working on. I can clearly remember a mid-sprint discussion with a developer who said, "Remind me: what is it we're trying to get done this sprint?"

In a long sprint with inadequately expressed product backlog items, this developer had quite literally forgotten whatever big goal the team was pursuing that sprint.

The sprint goal served to restore focus to that goal. I remember answering his question with, "increasing the average time spent on the search results screen," which was the sprint goal (and is a good sprint goal example).

I spend a lot of time advising organizations to narrow their focus regarding what they will achieve, whether it be for a year, a quarter, or even a single sprint. It's all too common for an organization or team to select a goal the way I load my plate at a buffet: a little of this, a little of that, some of the other.

Effective sprint goals encourage product owners to focus sprint plans on one specific goal for a product increment, so team members are always clear one what they are trying to accomplish in a time-bound iteration.

Why Are Sprint Goals Ineffective for Some Teams?

Yet with all the good sprint goals do, I admit I am a bit ambivalent toward Scrum's sprint goal. I coach teams to try creating sprint goals, but I also let them know that not every team benefits from a sprint goal.

Why don't I insist on a sprint goal? Here's one big reason. Sometimes a sprint cannot be focused on a single objective. Some teams simply have multiple things that need to get done.

For example, consider a digital agency. A team there could be simultaneously building a new website for one client, doing some mid-sized enhancements for a second client, and making small bug fixes or edits for five additional clients.

How should their sprint goal be written? Should the goal focus on the new website, which will consume the most work during the sprint? Or should it focus on the small edits if those are for the company's most important client?

Some teams merge all that into one goal with something like, "Finish the seven user stories we committed to."

That might be a sprint goal, but it doesn't benefit the team.

It doesn't provide the clarity a good goal gives, and it's too broad to focus the team on a single objective. Teams who write "finish the user stories we committed to" and consider that a sprint goal have effectively given up on sprint goals.

The Work of a Sprint Cannot Always Be Distilled into One Sentence

Here's the second reason sprint goals aren't always helpful. Sometimes, important things need to happen that are outside of a one- or two-sentence sprint goal.

In the early days of Scrum, sprint goals were usually used to evaluate the sprint—meet the sprint goal and the sprint was a success. Don't achieve the sprint goal and the sprint wasn't a success.

Looking at a sprint this way is incomplete.

Distilling an entire sprint to one goal has the same problems I have with to-do and personal productivity systems that tell me to make a list of the three most important things I need to accomplish each day.

Paying my mortgage is rarely the most important thing I need to do on a given day, but it does need to be done. Should I really put off paying my mortgage until the day I'm to be evicted for non-payment, at which point it truly is the most important thing that day?

A sprint is a complicated collection of work a team needs to accomplish; that work cannot always be encapsulated within a single goal.

Focus on Product Backlog Items When No Single Sprint Goal Is Possible

Imagine a golfer playing a typical hole somewhere right now. The golfer has the goal of getting the ball in the cup in four shots.

The golfer's first will be a drive from the tee. The second will be an approach shot, in which the player attempts to land the ball on the green. Next, the player will putt the ball, hoping to make it into the cup. But the player will likely need a second putt, which makes four shots overall.

Throughout this, the golfer is focused on a goal—get the ball in the cup in as few shots as possible. That goal is accomplished through a sequence of tasks—a long drive, a good approach shot, and a successful first or second putt.

I think a similar approach is appropriate for Scrum teams. The golfer can focus on successfully making each shot, knowing the shots will add up to the desired result of a good score on that hole. Scrum teams should be able to do the same: focus on the work, the sprint backlog items, that lead to a successful sprint.

This approach also supports a team that needs to work on things that cannot be contained within a single sprint goal, as was the case with the digital agency earlier.

In the same way that each of a golfer's shots can be viewed as leading indicators of a good score on the hole overall, a team can see completing individual sprint backlog items as leading indicators of success toward achieving some informal goal.

My Recommendation on Sprint Goals

If sprint goals work for your team, by all means you should continue using them. If you hear “What is it we’re trying to get done this sprint?” from the team, sprint goals can help. And if you’re [new to Scrum](#), you should absolutely try writing sprint goals for a few sprints.

But, if you’ve tried writing sprint goals and they just didn’t work for your team, then consider your effort a useful experiment, and proceed without them.

Let Me Know Your Thoughts

OK, let me hear it in the comments below. I know many people will strongly disagree. I’m quite open to changing my mind.

But let me know why I’m wrong—do us both a favor and make your argument something more than “because the Scrum Guide says so.”

Or if you agree, let me hear that I’m not alone in this opinion. And if you’ve tried writing sprint goals and found they weren’t for you, I’d love to hear more about that.

Please share your thoughts in the comments below.

[Take the Assessment](#)

Posted: March 21, 2023

Tagged: sprint planning, sprint goals

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of User Stories Applied for Agile Software Development, Agile Estimating and Planning, and Succeeding with Agile as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

[3 Ways to Help Agile Teams Plan Despite Uncertainty](#)

We might not like ambiguity, but it's a fact of life. Find out how to plan with uncertainty in mind.

Jun 21, 2022

[Read](#)

How Implementation Intentions Help My Sprints

Planning an exact time to do something within a sprint helps make sure you get the important stuff ...

Mar 22, 2022

[Read](#)

The Goal of Sprint Planning

Sprint planning may look like it's about tasks and estimates but those are not the goal.

Aug 24, 2021

[Read](#)

[Should a Team Assign Work During Sprint Planning?](#)

Some teams assign all tasks upfront. Others don't. Here's what works best.

Mar 30, 2021

[Read](#)

[Why the Fibonacci Sequence Works Well for Estimating](#)

If you've estimated with Planning Poker, you may very well have used cards with either the ...

Sep 10, 2019 • [31 Comments](#)

[Read](#)

Managing Sprint Interruptions by Tracking Buffer Use

In an ideal world, a Scrum team could perform the work of its sprints entirely uninterrupted. ...

Aug 13, 2019 • [26 Comments](#)

[Read](#)

Sorry, the browser you are using is not currently supported. Disqus actively supports the following browsers:

[Firefox](#)
[Chrome](#)
[Internet Explorer 11+](#)
[Safari](#)

[Derek Mahlitz](#) • 1 year ago

The way I teach teams is that not every PBI has to be a part of the sprint goal. If all PBI for the sprint goal is going well we can deliver all, if issues arise then we throw the non goal work out and concentrate just on sprint goal work. Some times 80% of PBI goes to Sprint Goal sometimes it's 30%, allow for changes as your team has other work each sprint. Start slow with goals and learn over time.

[Mike Cohn](#) • 1 year ago

I love it, Derek.

Duncan Bates • 1 year ago

Learn About Agile **Agile & Scrum Training** **More...**

New to Agile and Scrum? Online Certifications About Us

Scrum Video Courses Contact Us

User Stories In-Person Certifications Our Students

Planning Poker On-Site Courses Blog

Agile Software Development Training Schedule Books by Mike Cohn

Development Compare Courses Agile Tools

Agile Project Management PDUs and SEUs

Agile Presentations

Mountain Goat on YouTube

Agile Mentors Podcast

Elements of Agile

Assessment

Online Training Schedule

	All Courses	All Instructors	US Pacific Time	
Mar 27–28	Certified ScrumMaster (Live Online) <small>8:30am–3:30pm PDT</small>	with Mike Cohn	\$1400	More Info
Mar 29–30	Certified Scrum Product Owner (Live Online) <small>8:30am–3:30pm PDT</small>	with Mike Cohn	\$1400	More Info
Mar 29–30	Advanced Certified ScrumMaster (Live Online) <small>8:30am–3:30pm PDT</small>	with Lance Dacy	\$1300	More Info
Apr 13–14	Certified Scrum Product Owner (Live Online) <small>8:30am–3:30pm PDT</small>	with Brian Milner	\$1000	More Info
Apr 17–18	Certified ScrumMaster (Live Online) <small>8:30am–3:30pm PDT</small>	with Mike Cohn	\$1,300 \$1,200 <small>(Early Bird Price through Mar 27)</small>	More Info
Apr 17–18	Advanced Certified Scrum Product Owner (Live Online) <small>8:30am–3:30pm PDT</small>	with Brian Milner	\$1,300 \$1,200 <small>(Early Bird Price through Mar 27)</small>	More Info
Apr 19–20	Certified Scrum Product Owner (Live Online) <small>8:30am–3:30pm PDT</small>	with Mike Cohn	\$1,300 \$1,200 <small>(Early Bird Price through Mar 29)</small>	More Info
Apr 24–25	Certified ScrumMaster (Live Online) <small>1:00am–8:00am PDT</small>	with Lance Dacy	\$1,000 \$900 <small>(Early Bird Price through Apr 3)</small>	More Info

Apr 24–25	Certified ScrumMaster (Live Online) 8:30am–3:30pm PDT	with Brian Milner	\$1,000 \$900 <i>(Early Bird Price through Apr 3)</i>	More Info
May 1–2	Certified ScrumMaster (Live Online) 8:30am–3:30pm PDT	with Mitch Lacey	\$1,000 \$900 <i>(Early Bird Price through Apr 10)</i>	More Info
May 3–4	Certified Scrum Product Owner (Live Online) 8:30am–3:30pm PDT	with Scott Dunn	\$1,000 \$900 <i>(Early Bird Price through Apr 12)</i>	More Info
May 3	Better User Stories (Live Online) 8:30am–3:30pm PDT	with Mike Cohn	\$599 9 spots left!	More Info
May 10–11	Certified Scrum Product Owner (Live Online) 6:00am–1:00pm PDT	with Lance Dacy	\$1,000 \$900 <i>(Early Bird Price through Apr 19)</i>	More Info
Jun 7	Better User Stories (Live Online) 8:30am–3:30pm PDT	with Mike Cohn	\$599	More Info

[Get volume discounts](#) when you train your team or department

[Receive Our Upcoming Courses Email](#) for Next Available Dates

In-Person Schedule

There are no in-person courses currently scheduled.

[Sign Up Now!](#)

User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

by
Mike **Tagged:**
Cohn user stories, story maps, story writing workshop
Comments

A **story map** is a two-dimensional visual representation of the things a user wants to do. (The idea for story maps in software development—and the name—comes from [Jeff Patton](#).)

Story maps are a way to create a shared understanding of the product, to visualize user needs, and to elicit user story ideas during [story-writing workshops](#). I tend to create them using sticky notes and paper on a big wall or, if we're remote, on a virtual whiteboard.

Start with a Significant Objective

I recommend focusing story maps on a [single, significant objective or MVP \(minimum viable product\)](#). The MVP or objective should be chosen by the product owner in consultation with stakeholders.

A good significant objective is something that can be achieved in about three months. That period is long enough to make progress on big ideas yet short enough for progress to feel tangible.

Scrum teams can often use their product goal as their significant objective, as long as it was chosen to be about three months of work.

In some cases, a team may have a smaller significant objective and want to pursue more than one during a quarter. That's fine, but create a story map for the first objective before moving on to a second.

Who Participates when Story Mapping?

I recommend creating a story map with the entire [Scrum team](#) present, including the [product owner](#) (to answer questions and to lay out the focus of the map in terms of a significant objective or MVP) and the [Scrum Master](#) (to facilitate).

You will also generally want to include [stakeholders](#), customers, users, and others who can speak with authority about user needs.

Need help with story writing? [Join a live online Better User Stories course with Mike Cohn.](#)

The First Dimension: Across

The first dimension of a story map is a sequence of user tasks that is depicted horizontally across the map. The product owner leads participants through a discussion of the steps or actions a user will take to achieve a goal that is part or all of the significant objective.

A physical or virtual sticky note or index card is added to the map representing each step in the sequence. A meeting facilitator can write up all the cards if desired, but I find it more productive to let anyone write a new step on a card and then announce it as they add it to the map.

Figure 1: Cards are read horizontally by inserting the word *then* between each card.

Because the horizontal dimension of a story map represents a sequence, it can be read by inserting the word *then* between cards. So the map in Figure 1 is read as "a user does step one then step two then step three."

For example, suppose you're on a team that has been told to develop software that will allow company employees to submit expense reports and be reimbursed.

To submit an expense report, an employee first needs to log into the system. This becomes the first card in the map. Next the employee creates a new, blank expense report, which becomes the second card on the map—placed to the right of the log-in card to indicate that the two items occur in sequence.

This can be seen in Figure 2, which also shows that an employee next enters expenses and finally submits the expense report.

Figure 2: An initial story map for submitting expense reports.

To save time during the story-mapping process, the cards in a map are rarely written as complete [user stories](#) or [job stories](#). Writing short three- or four-word phrases is better while mapping: you want the story map to be quickly created and quickly read.

After the meeting, the product owner or an analyst will often move items into the team's [product backlog](#) management software (Jira, Trello, Monday.com, and so on) and create user stories using [user story templates](#) such as "As a ..., I ... so that...." or "When ..., I want ... so that..."

The Second Dimension: Down

Remember that earlier I told you a story map is two-dimensional. So what's the second dimension? Alternatives.

Still using that expense-report example, let's say meeting participants realize that some users could create a new expense by copying an old one. I do this quite often. Rather than starting with a blank expense report, I duplicate a recent one and edit the copy.

Duplicating an existing expense report can be added to the map under the option to create a blank expense report, as shown in Figure 3.

Figure 3. In addition to starting with a blank expense report, users may optionally duplicate an existing expense report.

Because the cards in a column are alternatives, the map is read by inserting the word or when reading down a column.

Now our map is read like this: A user logs in, then creates a blank expense report or duplicates an existing one, then enters expenses, and then submits the expense report.

Cards containing alternatives should be added to the map in descending order of priority—the most important at the top and the least important at the bottom. Priorities established like this in a story-mapping session aren't rigid, and the product owner can change them. But it can be useful to have short discussions about rough priorities within a column of alternatives.

Getting Fancy: Add a Header Row

Some story maps can get extremely wide. When that happens, it can be helpful to add a row of headings to the map. For example, suppose a simple word processor allows users to format, print, and save documents. Those categories of actions could be added to a map as headings as shown by the blue cards in Figure 4.

Figure 4. Blue heading cards have been added to indicate where portions of the map begin.

These heading cards can also be read with *then* between each: a user can format a document then print the document then save the document, in this case.

Heading cards are not themselves stories that will be implemented by the team directly, rather they are *epics*, *themes*, or whatever term your team uses to mean a group of related stories.

Getting Fancier: Use Submaps

I find wide maps difficult to follow even with heading rows. A better practice is to keep maps concise through the use of submaps. To do this, think of each card in a map as something you can metaphorically double click and expand into its own map.

Returning to the simple word processor that supported formatting, printing, and saving documents, we could create an extremely simple map with cards for only those three actions, as shown in Figure 5.

Figure 5. A high-level story map showing the three top-level actions in an extremely simple word processor.

Then imagine a submap contained within each of these high-level cards. If we mentally double-click on any of the top three cards, contained within that card is its own two-dimensional story map. This can be seen in Figure 6.

Figure 6. Each card in a map can potentially contain a submap.

There are a number of advantages to using submaps rather than creating an immensely large single map, including:

- Submaps are more readable
- Submaps add what are essentially third and fourth dimensions to the map
- Submaps, like functions or method in a programming language, can be referenced from more than one place in the parent map
- Submaps can include cards that contain their own submaps

Story Maps Help Uncover Missing Functionality

Story maps can help teams discover user activities or functionality they might have overlooked. To do this, walk through the story map from a user experience perspective and see if anything is missing.

In your walk-through, it can be very helpful to ask a few questions in each step, such as:

- What will a user most likely want to do next?
- What mistakes could a user make here?
- What could confuse a user at this point?
- What additional information could a user need?

If your product has multiple types of users, ask these questions for each type of user.

Wearing a user's shoes to walk through our sample story map about expense reports, I realized that users are often going to need to attach receipts for some expenses. So I'll need to add that as a new step.

And maybe the team suggests that the product needs to allow users to do this two ways. First, they suggest, it would be nice if the system itself could activate a device's camera and scan a receipt directly. Second, users will need an ability to attach an existing PDF or image to the receipt. Add each of these alternatives

below the Attach Receipts card on the story map.

Use a Story Map to Visualize Releases

Because story maps are arranged vertically in priority order, a product owner can use the story map to convey a product roadmap showing the intended sequence of features in one or more upcoming releases.

To draw a roadmap, add a horizontal line to the map and then drag above the line everything that must be included in the next release or version of the product.

Returning to the expense reporting system used earlier, a releasable product would give users the ability to log in, and make a new expense report. So these are placed above the line in Figure 7.

Figure 7. Lines can be added to a story map to indicate planned releases.

Let's assume that starting with a new blank report is good enough at first. The team can add the ability to duplicate an existing expense report later, so that is placed below the top line indicating it is not in the first release.

Users will need to attach receipts, but dragging and dropping existing image files is all that will be supported in the MVP release, as indicated by the label to the left of the top portion of the map. Activating the mobile device's camera to take photos of receipts is planned to be added in version two and is placed below the line.

Would You Like to Learn to Use Story Maps, Live and Online With Me?

Story maps are my favorite tool to help product owners, stakeholders, and developers understand user needs and the interconnectedness of those needs. I find them extremely efficient for brainstorming the product backlog items that will be needed to achieve a team's next significant objective.

On Wednesday May 3rd, I'll be running a [Better User Stories, one-day class live and online](#). I'll be helping you fix common user stories problems including adding detail and splitting stories, and we'll also tackle story mapping, including practical exercises for creating a valuable story map. You can [find out more and register for that class here](#).

Do you use story maps? Have you found them helpful? What challenges have you experienced with story maps? Please share your thoughts in the comments below.

[Read more](#)

Posted: March 14, 2023

Tagged: user stories, story maps, story writing workshop

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and

can be reached at hello@mountaingoatsoftware.com.
If you want to succeed with agile, you can also have
Mike email you a short tip each week.

You may also be interested in:

[How to Run a Successful User Story Writing Workshop](#)

What you need to know to conduct a story-writing workshop with your team.

Feb 28, 2023 • 3 Comments [Read](#)

[Epics, Features and User Stories](#)

I've been getting more and more emails lately from people confused about the difference between ...

Nov 08, 2022 • 93 Comments [Read](#)

[Advantages of User Stories over Requirements and Use Cases](#)

At the surface, user stories appear to have much in common with use cases and traditional ...

Oct 05, 2022 • 41 Comments [Read](#)

[Relationship between Definition of Done and Conditions of Satisfaction](#)

I'd like to clarify the relationship between two important concepts: a team's Definition of Done ...

Jun 28, 2022 • 38 Comments [Read](#)

[How Detailed Should a User Story Be?](#)

Capturing too much or too little detail in a user story causes problems. Here's how to get it ...

May 04, 2021 [Read](#)

[Job Stories Offer a Viable Alternative to User Stories](#)

As useful as user stories can be, they've never been right for every team. An exciting alternative ...

Oct 29, 2019 • 42 Comments [Read](#)

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile Assessment		

by
Mike Cohn Tagged:
6 user stories
Comments

[User stories](#) are simple, powerful tools but only if they're written in a way that encourages communication, helps you prioritize and plan a product, and focuses on end-user value.

A great way to ensure your stories do all of this is through a user story-writing workshop. Scheduling a workshop to write user stories can make it easier to generate a [product backlog](#).

Running a story-writing workshop opens up a whole new set of questions for people. Questions like:

How often do you hold a workshop?

Should it be once a year?

Every iteration?

Somewhere in between?

How do you know who to include?

If it's just a subset of the team are you missing valuable ideas?

If you include everyone how do you avoid conflicts and off-topic discussions?

These questions are very common. Although user story-writing workshops are an excellent way to generate a product backlog, it's important to do them the right way. Even experienced agile teams find themselves

going through a lot of trial and error to run a workshop that delivers great results.

If you are the kind of person who learns better by watching than by reading, you might prefer to learn about story-writing workshops and other user story tips in my [three free videos from the Better User Stories course](#). It goes beyond the information presented here to include how to have workshops with multiple teams and how to prevent stakeholder arguments from derailing the workshop.

Basics of Story-Writing Workshops

Let's start with a look at the basics of a story-writing workshop.

What is a story-writing workshop?

A story-writing workshop is a time for the product owner and development team members (developers) to write product backlog items in the form of user stories.

How often do teams hold workshops?

I recommend holding story writing workshops on a quarterly basis. Product backlogs are not meant to be a one-to-one replacement for requirements documents, where you replace reams of the system shall statements with an equally exhaustive list of "As a user, I want" stories. Product backlogs are meant to evolve over time. You cannot think of every possible feature a user might want up front. So don't try.

Instead, hold smaller, more frequent story-writing workshops. Target one-hour to a full day for each workshop (depending on whether you include story mapping or not). Be sure to include lunch and a few breaks if the workshop will last all day.

What is the focus of a story-writing workshop?

Each story-writing workshop should revolve around how to achieve a single significant objective (for example, launch a new [better user stories video course](#)) or a series of smaller objectives (for example, hold a [webinar about better user stories](#), update the BUS courseware, and begin to [sell group licenses](#)).

A **significant objective** is a goal that will almost always take more than one iteration or sprint to achieve. It can often be thought of as an MVP or MMF. The product owner will determine this with the stakeholders prior to the meeting.

A **Minimum Viable Product (MVP)** is a version of a product that allows a team to collect the maximum amount of information with the least effort. (For example, my free, three-video series on Better User Stories was an MVP for the larger Better User Stories course.)

The **Minimum Marketable Feature (MMF)** is a chunk of functionality that delivers a subset of the customer's requirements, and that is capable of returning value to the customer when released as an independent entity. (The [Better User Stories webinars](#) I hold throughout the year could be considered MMFs for the larger Better User Stories course.)

Who participates in user story writing workshops?

Mandatory participants include the product owner (or key stakeholder or whatever you call the visionary for the product) and the whole [team](#) of developers. You might be tempted to try this with just a subset of the team, but I strongly encourage you to include everyone in the beginning. People will be more invested and you'll get more creative solutions.

Although [product owners](#) can hold this meeting on their own, I always find it helpful to have a facilitator present. The team's [Scrum Master](#) or agile coach can help ensure the workshop runs smoothly, that everyone stays engaged, and that people take breaks.

You might also want to include [stakeholders](#), users, and customers if they are involved in the significant objective being discussed in the workshop.

Story-Writing Workshop Agenda and Prep

Prior to the workshop, be sure to have plenty of writing tools, paper, and sticky notes or index cards available. If you are doing this virtually, you'll need to prepare a workspace in [Miro](#), [Mural](#), or your favorite whiteboard tool that mimics an in-person environment.

Somewhere in your virtual tool, physical whiteboard or on a large sheet of paper, write the basics of the [user story template](#).

As a type of user

I some goal

so that some reason.

Start the workshop by asking the product owner (or key stakeholder) to explain the significant objective so everyone has a shared understanding of the goal. Then, as a group, discuss the user roles and personas that you'll need to consider.

Then, it's time to start capturing user activities as [agile user stories, epics, and themes](#).

I find it takes about 90 minutes of brainstorming (sometimes as much as three hours) to load up a few months worth of high-priority items in a product backlog. That's a guideline and will, of course, need to be adjusted based on the number of participants, previous agreement on the product vision, and other factors.

Next Steps

Many teams find that they feel a great sense of accomplishment after completing a workshop, but they are worried they've overlooked some requirements or just feel overwhelmed by the massive product backlog they just created.

That's why at my story-writing workshops, I also introduce a way to graphically visualize the relationships between the stories you write. There are a few ways to do this, such as a goal-story hierarchy or a mind map.

I recommend using a story map. You can read more about [story maps in this blog post](#).

[Read more](#)

Posted: February 28, 2023

Tagged: user stories

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

User Stories: How to Create Story Maps

Story maps help to create a shared understanding of the product, visualize user needs, and elicit ...

Mar 14, 2023

[Read](#)

[Epics, Features and User Stories](#)

I've been getting more and more emails lately from people confused about the difference between ...

Nov 08, 2022 • [93 Comments](#)

[Read](#)

[Advantages of User Stories over Requirements and Use Cases](#)

At the surface, user stories appear to have much in common with use cases and traditional ...

Oct 05, 2022 • [41 Comments](#)

[Read](#)

[Relationship between Definition of Done and Conditions of Satisfaction](#)

I'd like to clarify the relationship between two important concepts: a team's Definition of Done ...

Jun 28, 2022 • [38 Comments](#)

[Read](#)

[How Detailed Should a User Story Be?](#)

Capturing too much or too little detail in a user story causes problems. Here's how to get it ...

May 04, 2021

[Read](#)

Job Stories Offer a Viable Alternative to User Stories

As useful as user stories can be, they've never been right for every team. An exciting alternative ...

Oct 29, 2019 • [42 Comments](#)

[Read](#)

Learn About Agile

New to Agile and Scrum?

Scrum

User Stories

Planning Poker

Agile Software Development

Agile Project Management

Agile Presentations

Mountain Goat on YouTube

Agile Mentors Podcast

Elements of Agile

Assessment

Agile & Scrum Training

Online Certifications

Video Courses

In-Person Certifications

On-Site Courses

Training Schedule

Compare Courses

PDUs and SEUs

More...

About Us

Contact Us

Our Students

Blog

Books by Mike Cohn

Agile Tools

Concise Tips from Mike Cohn to Help You Succeed with Agile

First Name

Email Address

Also keep me updated about:

Updates from Mountain Goat Software about training, free videos and more

About Mike Cohn

Since 1995 Mike Cohn has been using agile and Scrum to build high-performing software development teams and organizations. Companies large and small and across all industries have attended his training courses. Mike is the author of User Stories Applied for Software Development, Agile Estimating and Planning, and Succeeding with Agile Software Development using Scrum.

A Look Inside “101+ Inspiring Quotes About Agile”:

"I love your emails. Whenever I see an email from you, I always pause whatever I'm doing and focus on them. Their nuggets of wisdom are short but deep, practical and meaningful, and touch upon the important aspects of Scrum life."

Gareth Gauci
CTO, ICON

"Getting these tips is one of the highlights of my agile experience as a scrum master and coach. I look forward to seeing your emails in my inbox. I gain really good insights that can be readily applied within my team and I've shared this with my peers so they too can sign up to get these tips."

Carol Fairbairn

ICP-ACC-Agile Coach / CSM-Agile Scrum Master / SA-Scaled Agile Framework

"I love the weekly tips you send out. I love that they are concise, immediately actionable (if we wanted to take action on them) and a quick read. Yours are one of the few emails I continue to subscribe to and I know when I see it in my inbox that it isn't going to take much time or brain power to reduce my inbox by 1, so the effort is well worth the value given."

Kim Kimmey

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile Assessment		

by
Mike Cohn Tagged:
59 teams, skills, cross functional, job roles, specialist
[Comments](#)

How do agile teams achieve cross-functionality? Hint: Contrary to what you might have heard, cross-functional does not mean everyone on the team must know how to do everything.

Every sandwich shop in the world has figured out how to work cross-functionally. So it's surprising to me that so many [agile and Scrum teams](#) still struggle with [what it means to be cross-functional](#), especially when it comes to balancing specialists on an agile team.

Think about the last time you watched a team fulfill your order for a sandwich. Likely, you noticed that there were one or two specialists—someone who only cooks, and someone who only works the counter.

But you probably also saw a few multi-skilled individuals. Maybe someone sliced the meat for your sandwich, but then took off their gloves to answer the phone and take an order. Or maybe the person who rang you up, also moved over to flip a sandwich over once the cheese had melted.

When I did my obligatory teenage stint at a fast food restaurant, I was a floater. I wasn't as quick at wrapping burritos and making tacos as Mark, one of the cooks. Nevertheless, I wrapped plenty of burritos during the lunch rush.

And whenever the cash register needed a new roll of paper, I had to yell for my manager, Nikki, because I could never remember how to do it. Nikki knew, though, that she could depend on me to ring up orders when

she got busy.

Although Mark and Nikki had clear specialties, we were consistently able to achieve our common goal of delivering products to our customers, because we also had people like me: floaters.

Specialists on Agile Teams Are OK

It's not that different when agile teams work collaboratively. Yet perhaps the most prevalent and persistent myth in agile is that to be cross-functional, every team member must possess every skill necessary to complete the work.

This is simply not true. It is perfectly acceptable to have specialists on Scrum teams.

A cross-functional team has members who together have the right mix of skills to deliver a [working product increment](#) to their customers. But it's rare to see a team where each member has all of those skills.

I suspect a lot of productivity has been lost by teams pursuing some false holy grail of having each team member learn how to do everything.

If my team includes the world's greatest database developer, I want that person doing amazing things with our database. I don't need the world's greatest database developer to learn JavaScript.

How Cross-Functional Collaboration Works

However, too much reliance on specialists can indeed cause problems for agile teams. Too many specialists make it hard to balance the types of work that can be brought into a sprint.

Let's look at a few cross-functional collaboration examples.

In Figure 1, we see a four-person team where each person is a specialist. Persons 1 and 2 are programmers and can only program. This is indicated by the red squares and the coding prompt icon within them.

Persons 3 and 4 are testers who do nothing but test. They are indicated by the green square and the pencil and ruler icons within those. You can imagine any skills you'd like, but for these examples I'll use programmers (red) and testers (green).

The four-person team in Figure 1 is capable of completing four red tasks in an iteration and four green tasks in an iteration. They cannot do five red tasks or five green tasks.

But if their work is distributed across two product backlog items as shown in Figure 2, this team will be able to finish that work in an iteration.

But, any allocation of work that is not evenly split between red and green work will be impossible for this team to complete. This means the specialist team of Figure 1 could not complete the work in any of the allocations shown in Figure 3.

The Impact of Multi-Skilled Team Members

Next, let's consider how the situation is changed if two of the specialist team members of Figure 1 are now each able to do both red and green work. I refer to such team members as *multi-skilled individuals*.

Sometimes people mix as *generalists and specialists*, but I find that misleading. We don't need someone to be able to do *everything*. It is often enough to have a team member or two who has a couple of the skills a team needs rather than *all* of the skills.

Figure 4 shows this team. Persons 1 and 2 remain specialists, only able to do one type of work each. But now, Persons 3 and 4 are multi-skilled and each can do either red or green work.

This team can complete many more allocations of work than could the specialist team of Figure 1. Figure 5 shows all the possible allocations that become possible when two multi-skilled members are added to the team.

By replacing just a couple of specialists with multi-skilled members, the team is able to complete any allocation of work except work that would require 0 or 1 unit of either skill.

In most cases, a team can avoid planning an iteration that is so heavily skewed simply through [carefully mixing the type and sizes of product backlog items](#). In this example, if the first product backlog item selected was heavily green, the team would not select a second item that was heavily green, even if that means [pulling some product backlog items out of order](#).

The Role of Specialists on an Agile Team

From this, we can see that specialists can exist on high-performing agile teams. But, it is the multi-skilled team members who allow that to be possible. There is nothing wrong with having a very talented specialist on a team—and there are actually many good reasons to value such experts.

But a good agile team will also include multi-skilled individuals. These individuals can smooth out the workload when a team needs to do more or less of a particular type of work in an iteration. Such individuals may also benefit a team in bringing more balanced perspectives to design discussions.

Evidence from My Local Grocery Store

As evidence that specialists are acceptable as long as they are balanced by multi-skilled team members, consider your local grocery store. A typical store will have cashiers who scan items and accept payment. The store will also have people who bag the groceries for you. If the bagger gets behind, the cashier shifts and helps bag items. The multi-skilled cashier/bagger allows the store to use fewer specialist baggers per shift.

What Role Do Specialists Play on Your Team?

What role do specialists play on your team? What techniques do you use to allow specialists to specialize? Please share your thoughts in the comments below.

[Take the Assessment](#)

Posted: February 14, 2023

Tagged: teams, skills, cross functional, job roles, specialist

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

[Don't Equate Story Points to Hours](#)

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • 39 Comments

[Read](#)

[From Project Manager to Scrum Master -3 Tips for Making the Transition](#)

What does a good project manager need to do to become a great Scrum Master?

Aug 23, 2022

[Read](#)

[Relationship between Definition of Done and Conditions of Satisfaction](#)

I'd like to clarify the relationship between two important concepts: a team's Definition of Done ...

Jun 28, 2022 • [38 Comments](#)

[Read](#)

[Agile Spikes Deliver Knowledge So Teams Can Deliver Products](#)

On agile projects, a spike is a time-boxed research activity that helps teams make better decisions ...

May 11, 2022 • [48 Comments](#)

[Read](#)

[What Is a High-Performing Agile Team?](#)

How to build and sustain a Scrum team that exceeds the sum of its parts.

Apr 19, 2022

[Read](#)

What Does Scrum Mean by Cross Functional Teams?

What exactly does cross functional mean and why does it matter?

Apr 05, 2022

[Read](#)

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile Assessment		

by
Mike Cohn **Tagged:** product owner, leadership
16 Comments

[Product owners](#): Want some advice from [Scrum Masters](#) and [teams](#) on how to be effective? You've come to the right place.

I've worked and talked with countless Scrum Masters and development teams over the years. They've told me about good product owners and bad product owners. They know that a good product owner can turn a good product into a great product—and a bad product owner can sink a project faster than any bad technology decision.

Successful product owners usually don't have better resumes than their less-effective counterparts. What they do have is a knack for knowing what people need (whether those people are their stakeholders, customers, or teams) and ensuring they have it (or can understand why not). What's behind that knack

I've found six things effective product owners must give to the various people they serve: Availability, Vision, Collaboration, High Expectations, Priorities & Flexibility, and Storytelling.

#1. Good Product Owners Are Available

Scrum teams are often under tremendous pressure to deliver more, faster. But that's impossible without time and support from the product owner to answer questions and clarify expectations.

Start with the mindset that, as much as any developer, you're part of the team. Sit with the team if you are in an office setting; if you are remote, participate in the team's Slack or Discord channels. Attend [daily scrums](#). Go to [sprint planning](#), [reviews](#), and [retrospectives](#).

Remember your role on the team is to be the authority on what users *want*, and more importantly, what they *value*. Demonstrate that you understand the product and how it will be used—not just by writing [user stories](#), but by knowing the product as it evolves and develops. That includes being able to demo new functionality at sprint reviews.

#2. Effective Product Owners Paint a Vision

The best product owners have a compelling vision for their product. This doesn't need to be a Steve Jobs-style vision of an entirely new industry. But teams do need to know why a product is being created.

Product owners like you are responsible for making sure the product has a vision that you can articulate. If you can't articulate it, don't start building it.

Team members become motivated when you provide them with what [Larson and LaFasto](#) call "a clear, elevating goal." According to these experts, the lack of a clear, elevating goal is the reason given most frequently for why teams fail.

The best example of a clear, elevating goal was President Kennedy's call to land a man on the moon before the end of the 1960s.

It was clear: Everyone would know if the goal had been met or not.

It was elevating: We can easily imagine the excitement of being on a team of such historic importance.

The product goal should be just as clear. Of course, the goal does not have to be quite as elevating as putting a man on the moon, but it should be something we want to accomplish—either because the goal itself is meaningful or because of the challenge it will be to achieve it.

Examples of clear, elevating product goals

The following are examples of clear, elevating product goals:

Win a "Product of the Year" award from the magazine covering our industry.

Reduce call duration in our call centers by three minutes per call.

Create a product so simple to use that training time is cut from three days to half a day.

The product vision can (and should) change over time. Even Jeff Bezos of Amazon could not have anticipated all that Amazon has become. If he had, he certainly wouldn't have given Amazon its initial slogan of "Earth's biggest bookstore."

#3. Successful Product Owners Collaborate with All Stakeholders

Product owners are great at understanding that [users and customers are stakeholders](#) in a product or project's success. They're also good at recognizing business stakeholders within an organization.

However, too many product owners don't recognize the importance of another group of stakeholders: development team members! The team has a vested interest in the success of the product and must be included along with business, user, and customer stakeholders.

That means considering developers' opinions on priorities.

And it means trusting their advice on product features and needs, too.

When team members ask to be able to do something, they want the product owner to trust that they are doing so for the good of the product.

For example, suppose team members want to clean up some old code. You, as product owner, might ask questions like:

What would happen if we don't ever clean up that code?

What would happen if we put it off for two sprints?

It's possible that answers to these questions lead to a decision not to perform that code clean-up. But it is more likely that the answers will help you assess whether it's better to do the code clean-up now or in a few weeks.

#4. The Best Product Owners Set High Expectations

Have high expectations of your team. Developers thrive on solving challenging problems—things like 'make this run ten times faster' or 'do that in one-fourth the memory'—when given the freedom and time to pursue creative solutions.

Having the time to do good work is key to success. I can't imagine standing over Van Gogh's shoulder saying, "Paint faster. It's good enough." Yet teams hear this nearly daily.

There will be times when products ship with known imperfections. There will be times a team needs to rush a feature doing a "[good enough](#)" version for now. And, believe me, the development team understands this.

But those times need to be balanced with times when team members can do quality work.

When people create anything [beyond a certain speed](#), they take shortcuts that come back to haunt them (and you). Very few products are worth doing at high speed, and all rushed products end up costing far more over the next few versions.

Since you want to deliver as quickly as possible, with high quality, it's wise to schedule time for learning and improvement. Technical skills go out of date quickly. New technologies are developed. Old technologies are improved, enhanced, or shown to be usable in new ways.

Team members know all this. It's why they want time to improve their existing skills and learn new ones. Many team members also enjoy the challenge and excitement that comes with learning.

It's a win-win situation: team members do the hard work of learning something and you, as product owner, benefit from what they've learned.

#5. What Makes a Good Product Owner in Agile? Priorities and Flexibility

A couple of years ago, I was talking to a colleague and I made the comment that a particular practice “would be good for time-constrained projects.” They challenged me: “Show me a project that is not time-constrained.” I laughed and said, “You’re right. There’s no such thing.”

Essentially all projects are time-constrained to some extent, so product owners must prioritize the functionality they want built. If you say, “Everything is top priority,” then when the project runs out of time, you can expect a random set of functionality, because the team won’t have known what was truly the most important piece to finish first.

That doesn’t mean priorities are set in stone. [Product owners can \(and likely should\) change their minds](#) as the market shifts and as they learn more about the product that’s being developed.

For example, suppose the team is halfway through a product backlog and an opportunity presents itself to sell the half-finished product to an initially small set of customers. That might be worth considering.

Alternatively, suppose a fierce competitor announces some new blockbuster feature. Adding that feature to your own product may become a higher priority than much of the other remaining work.

Indecisiveness and random changes are annoying, but changes based on new knowledge are an important interruption. In fact, if the team has established a development process to meet a product owners’ high expectations, the team’s ability to respond to change can become a competitive advantage for the organization.

#6. Product Owners Are Good Storytellers

Teams need to hear about features in chunks that are big enough to understand but small enough to estimate. User stories perfectly meet this need.

But user stories cannot stand alone. Every user story is a placeholder for a conversation: a reminder for the product owner and the developers to talk about a feature. The user story may be the most visible part of a story, but the most important part is the conversations that take place to refine the story and communicate the desired behavior of the software.

User stories should be tied to users’ goals and lead to achieving the clear, elevating goal of the product. A good product owner will not just rattle off a list of stories; they will be able to [relate those stories both to workflows and to knowledge of how users will interact with the system](#).

How to Be a Successful Product Owner

Great product ownership is hard. You have to spend time looking outward toward customers, users, competitors, trends in your industry and more. But you also have to spend time looking inward at the team,

working with them, and answering their questions.

None of the six things I have described require any special skills, training, or domain knowledge. Anyone can do them. But the importance of these attributes may be one of the most important things for a successful product owner to understand.

Be available, prioritize and re-prioritize the work, tell user stories, articulate a clear, elevating goal, set high expectations, and collaborate. You'll be rewarded with superior results.

What Do You Think?

What do you think is missing from my list? If you're a stakeholder, team member or Scrum Master, what more do you want from your product owner? If you're a product owner, what did I miss that your team wants from you? Please share your thoughts in the comments below.

Get Your Free Scrum Cheat Sheet

Sign up below to receive this FREE reference.

First Name

Email Address

[Privacy Policy](#)

Posted: January 31, 2023

Tagged: product owner, leadership

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the *Better User Stories* video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

From Project Manager to Scrum Master - 3 Tips for Making the Transition

What does a good project manager need to do to become a great Scrum Master?

How to Create Self-Sufficient Scrum Teams

Mar 01, 2022

[Read](#)

My Favorite Hard Questions to Ask When Making a Decision

Asking hard questions is a great way to confirm the right decision is being made.

Feb 08, 2022

[Read](#)

The Product Owner's Second Team

Successful product owners will see their stakeholders as a team, not merely a set of individuals.

7 Questions to Determine if Being a Scrum Master Is Right for You

Thinking of a career as a Scrum Master? 7 questions to help you decide if it's the right job for you.

Top 5 changes in the 2020 version of the Scrum Guide

Recently the 2020 version of the Scrum Guide was released. What changes were made that you need to ...

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile Assessment		

Become the Scrum Master Your Team Needs

by
Mike Cohn **Tagged:**
32 scrum master
[Comments](#)

Are you thinking about becoming a [Scrum Master](#) and wondering if you [have what it takes to do the job](#)? Are you already a Scrum Master and asking yourself, What are the characteristics of an effective Scrum Master? Or are you perhaps looking to hire someone and want to know what Scrum Master skills and traits to look for?

If you answered yes to any of those questions, this article is for you. I've identified six Scrum Master characteristics and behaviors that are essential for success. I've even included a seventh, bonus Scrum Master quality, for those who want to take their success to the next level.

Responsible

Being responsible is an essential characteristic of Scrum Masters. While a ScrumMaster is not responsible for the success of the project—that remains with the team—a ScrumMaster is responsible for the team's adoption and practice of the Scrum process.

A Scrum Master takes on this responsibility without assuming any of the power that might be useful in achieving it. They are instead a servant leader to the team.

A Scrum Master's role is similar to that of an orchestra conductor. Both must provide real-time guidance and

leadership to a talented collection of individuals who come together to create something that no one of them could create alone.

Boston Pops conductor [Keith Lockhart has said of his role](#), “People assume that when you become a conductor you’re into some sort of a Napoleonic thing—that you want to stand on that big box and wield your power. I’m not a power junkie, I’m a responsibility junkie.”

In an identical manner, ScrumMasters thrive on responsibility—that special type of responsibility that comes without power.

Humble

A humble Scrum Master is not in it to pump up their ego. They know that it’s not a job that comes with ego-boosting perks like a company car or parking spot near the building entrance.

Humble Scrum Masters take pride in their achievements but know that they look good when their [teams](#) look good. So they do whatever is necessary to help the team achieve its goal. And when the team achieves something extraordinary, they make sure everyone in the organization takes note and celebrates that success, [including the team members themselves!](#)

Humble Scrum Masters are quiet and wait for others to weigh in. They embrace the effectiveness of the all-important pause, the enabling power of silence. When they are tempted to speak, they wait first for others—serene in that awkward stillness that follows a complicated question or thought-provoking comment.

They are comfortable not being the one with all the answers. They know that to become a fully autonomous, [self-organizing team](#), the team needs silence more than it needs insightful comments from the Scrum Master.

Part of being quiet is listening. Humble Scrum Masters resist the temptation to begin forming a mental reply before a team member has finished speaking. Instead, when someone else is talking they give their full attention. They recognize the value in all team members and, by example, they lead others to the same opinion.

Being humble is one of the most effective Scrum Master traits.

Collaborative

Scrum Masters nurture a collaborative culture within the team. They ensure team members feel able to raise issues for open discussion and that they feel supported in doing so. They establish collaboration as the team norm and call out inappropriate behavior (if the calling-out isn’t already done by other team members). And they ensure the development team is working together effectively throughout each sprint.

They are aware of what’s happening with the team without crossing the line into micromanagement. In short, they check in without making it feel like they’re checking up.

While checking up and checking in may seem similar, they are actually quite different. Scrum Masters who check in don’t just ask about progress; they offer real help in removing any obstacles or distractions that are impeding that progress. They avoid blaming individuals at all costs. And, perhaps most importantly, they

ensure the team is given complete autonomy to solve whatever problem the team's been given.

Checking up is very different. When you check up on someone, you are looking for conformity to a plan—did the programmer finish the changes last night as she said she would? Did the tester automate the tests as promised or just run them manually “one more time” again?

Checking up is about getting status-type information. That's fine. But checking in goes beyond that to include offering to help remove any impediments to progress.

Collaborative Scrum Masters check in often, to ensure the team can move together toward their sprint goal.

Committed

While the Scrum Master role [does not always require a full-time, eight-hour-a-day commitment](#), it does require full commitment to the role for the duration of the project. The Scrum Master needs the same high level of commitment to the project and goals of the current sprint as team members.

Committed Scrum Masters have a daily goal to resolve all known impediments. That doesn't mean they can necessarily clear every impediment by the end of every day—some impediments take some time and maneuvering to remove. But they make some progress every single day.

They know that Scrum Master strengths are not about doing the work—but about making the work easier for the team to accomplish.

Influential

An essential trait of Scrum Masters is the ability to influence others, both on the team and outside it.

Initially, team members may need to be influenced to give Scrum a fair trial or to behave more collaboratively; later a Scrum Master may influence a team to try new technical practices such as test-driven development or pair programming. A Scrum Master should know how to exert influence without resorting to a command-and-control “because I say so” style.

Scrum Masters can also be called upon to influence those outside the team. A traditional team may need to be convinced to provide a partial implementation to the Scrum team, a QA director may need to be influenced to dedicate full-time testers to the project, or a vice president may need to be convinced to try Scrum at all.

While all Scrum Masters should know how to use their personal influence, the ideal ScrumMaster has a degree of corporate political skill. The term corporate politics is often used pejoratively, but it's an asset to a team to have a ScrumMaster aware of things like how decisions are made in the organization, who makes them, and which coalitions exist.

Knowledgeable

The best Scrum Masters have the technical, market, or specific knowledge to help the team in pursuit of its

goal.

[LaFasto and Larson](#) have studied successful teams and their leaders. They have concluded that “an intimate and detailed knowledge of how something works increases the chance of the leader helping the team surface the more subtle technical issues that must be addressed.”

LaFasto and Larson note that the knowledge may be broad rather than deep but that team leaders (such as Scrum Masters) “need to be conversant around the key technical issues.”

Knowledgeable Scrum Masters understand enough about key technical issues to be able to explain the problem to others in the organization when necessary. They also have a general understanding of the market and the opportunities and challenges surrounding the product.

Scrum Masters who don’t yet have all of these skills must work to acquire them. They don’t need to become expert developers or product marketers. But they do need to know enough about these areas to assist the team in resolving problems.

Bonus Trait: Know When to Break the Rules (And When to Hold Firm)

The [rules of Scrum](#) are minimal but they exist for good reasons. Each helps a team be agile and increases the likelihood of success.

Scrum Masters are unbending when it comes to certain rules and uncompromising in their confidence in the team’s ability to find a way forward. At the same time, they use common sense when a rule doesn’t work for their situation.

Here are some rules Scrum Masters should enforce almost all the time:

- Don’t let the team get away with taking partial credit for a story ([with one notable exception](#)).
- Don’t allow the team to honor only the agile principles they find the most enjoyable. (For example, many teams incorrectly interpret the Agile Manifesto to say that no documentation is needed.)
- Encourage participation from every team member in [daily scrums](#) and [retrospectives](#).

And an unbreakable rule for the Scrum Master is to refrain from solving problems for the team that they should solve themselves. Sometimes working together to solve a problem (such as too many bugs) helps a team identify ways to make sure the problem doesn’t recur (such as test-driven development, pair programming, or continuous delivery).

But, as the saying goes, rules are meant to be broken.

Successful Scrum Masters learn to recognize those situations in which it’s appropriate to break the rules.

Never extending a sprint is a great rule. Usually. Can it be broken? Yes—not often and always for a good reason (such as a holiday that makes a longer sprint sensible).

Working software at the end of each sprint is a great goal. Is it [reasonable to expect it every single sprint](#)? Not necessarily.

Three hours is the maximum duration of a [sprint retrospective](#). And 99% of my retrospectives are done

in less time, usually far less time. But have teams I've coached broken this three-hour rule? Yes, occasionally, when a really hot topic was being discussed and it seemed better to continue than to postpone a crucial conversation.

Qualities of a Good Scrum Master

To sum up, Scrum Masters share at least 6 traits: They are responsible, humble, collaborative, committed, influential, knowledgeable. And the best Scrum Masters also know when to abide by the rules and when to break the rules.

[Sign Up Now!](#)

Posted: January 17, 2023

Tagged: scrum master

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of User Stories Applied for Agile Software Development, Agile Estimating and Planning, and Succeeding with Agile as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

From Project Manager to Scrum Master - 3 Tips for Making the Transition

What does a good project manager need to do to become a great Scrum Master?

Aug 23, 2022

[Read](#)

Seven Tips for Running a Virtual Daily Scrum

How do you effectively facilitate a daily scrum with a remote team?

May 03, 2022

[Read](#)

What Does Scrum Mean by Cross Functional Teams?

What exactly does cross functional mean and why does it matter?

Apr 05, 2022

[Read](#)

How to Create Self-Sufficient Scrum Teams

Mar 01, 2022

[Read](#)

Retrospecting with a Quiet Team

How to run a Retrospective when your team won't talk.

Feb 01, 2022

[Read](#)

How To Coach Your Team to Run a Daily Scrum Meeting When You Cannot Attend

Coaching your team to run a daily scrum without you.

Dec 21, 2021

[Read](#)

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

by
Mike Cohn **Tagged:**
sprinting, meetings, ceremonies
55 [Comments](#)

Successful [Scrum](#) implementations involve a handful of important sprint events (also called sprint meetings or sprint ceremonies. This includes meetings such as sprint planning, sprint review, daily scrum, sprint retrospective, and more.

There's often confusion about who participates, when these events are conducted, how long each takes, the purpose of each event, and more.

To reduce the confusion, we've created [infographics](#) that answer each of these questions for sprints of 1-, 2-, 3- and 4-weeks.

Sprint Planning

The [sprint planning](#) event marks the official start of the sprint. Once this event starts, so has the sprint.

The [Scrum Master](#), [product owner](#), and [developers](#) (development team) all participate. Others may attend on rare occasions when the product owner and team both agree it's appropriate.

For example, if the coming sprint will include developing functionality best explained by a subject matter expert (who is not the product owner), it can be useful to have that person attend. Usually, however, that type of discussion is best conducted outside the actual planning meeting.

The length of the sprint planning ceremony is proportional to the length of the sprint. A four-week sprint should be planned in no more than 8 hours. A one-week sprint should be planned in no more than two hours.

These are time boxes (maximums). I recommend teams target completing sprint planning in about half the allowable time box.

As input into sprint planning, the Scrum Master will bring data on the team's average [velocity](#) and most recent velocity. The product owner will bring the product backlog, or at least the highest priority items on the product backlog. On many teams, the product owner will also supply a draft sprint goal, which may be collaboratively revised through the planning process.

The outputs of sprint planning include a team that is smarter about and better prepared for the upcoming work. Additional outputs include a [sprint backlog](#) and an agreed upon sprint goal.

Daily Scrum

The [daily scrum](#), also known as the daily standup, is a short 15-minute timebox during which team members synchronize effort each day. Daily scrums enable team members to ensure the right things are being worked on by the right people at the right time.

Each day, each participant addresses three topics:

1. What did I do yesterday to help achieve the sprint goal?
2. What will I do today to achieve the sprint goal?
3. What, if anything is impeding or blocking progress toward the sprint goal?

Questions can be phrased in any number of ways. For example, many teams find it beneficial for participants to describe what was *accomplished* rather than what they *did*.

Participants include the Scrum Master, development team, and, in my opinion, the product owner.

There is some debate within the Scrum community about whether the product owner should participate. Excusing the product owner from the daily scrum creates a separation within the overall team. Us-and-them feelings exist already in too many organizations. I don't know why a Scrum team or its product owner would want to do anything to further enhance that negative attitude.

Each daily scrum is limited to 15 minutes. The intent is for it to be a brief update and synchronization effort.

Unlike sprint planning, I don't recommend trying to complete a daily scrum in half the recommended timebox. For most teams, 5-7 minutes is simply not enough time to raise any real issues or understand the work being accomplished. When teams shorten the daily scrums too much, the ceremony devolves into a series of rote updates, such as "Yesterday I did such-and-such. Today I'll do this-and-that. Nothing is in my way."

There are no formal inputs to the daily scrum. The only output is increased coordination of the developers' work.

Sprint Review

The [sprint review](#) event happens on the last day of the sprint. It should be attended by the product owner, Scrum Master, the development team and any appropriate stakeholders. The stakeholder participants may vary from sprint to sprint based on what has been delivered.

The sprint review is time boxed to a maximum of four hours for a four-week sprint. It is proportionately shorter for shorter sprints, down to one hour for a one-week sprint.

As input to the sprint review, the team should show all of the product backlog items that meet the team's [definition of done](#). This means that the team does not show work that is still in process. Sometimes, however, it may be worth making an [exception to this rule](#).

The demo of finished functionality is the central activity of a typical sprint review. But most teams will also take time to discuss progress and problems. You can read about my [recommended agenda for the sprint review](#).

The goal of the review is to solicit feedback on what was built during the sprint. The product owner considers all feedback and can make changes to the [product backlog](#) as appropriate. The output of a sprint review is therefore a revised product backlog.

Sprint Retrospective

The sprint retrospective event is a time for team members to consider how to improve their way of working. This means they may change aspects of how they do Scrum, such as the length of their sprints.

But a retrospective can also cover general aspects of working together, such as whether to ban morning meetings or which topics are appropriate to discuss on Slack and which require a face-to-face conversation.

The [sprint retrospective should be attended by the whole team](#)—including the Scrum Master and the product owner. To do otherwise is to create a schism within the team. A good agile team should avoid any behavior that leads to an us/them mindset.

There are no formal inputs to a sprint retrospective other than a willingness to improve. The output is a list of changes the team will make to how it works. Some teams formalize this list as an [improvement backlog](#).

The sprint retrospective is formally timeboxed to 3 hours. A retrospective may occasionally take that long but most teams will conduct most retrospectives within an hour.

Product Backlog Refinement

Product backlog refinement refers to ensuring the items at the [top of the product backlog](#) are ready for the next sprint. This can include adding detail to existing items, estimating, deleting items, adjusting priorities, [splitting product backlog items](#) (or [user stories](#)) so as to better fit within a sprint, and creating new items.

While product backlog refinement itself is necessary, it is not mandatory that a team do refinement as a formal ceremony or that it be done every sprint. Most teams will, however, conduct regular product backlog refinement meetings, usually once per sprint or once per week.

Usual guidance is to spend no more than 10% of a team's total available time on product backlog refinement both in meetings and in discussions that may result from those meetings.

Most teams will have the entire team participate (including the product owner and Scrum Master). Unless a team will be estimating product backlog items during its refinement meetings, I find that perhaps half to two-thirds of the development is sufficient and reduces the overall meeting time burden on a team.

The only inputs to this ceremony are the items at the top of the product backlog. Outputs are product backlog items that are often split to be smaller and better fit within a sprint as well as greater understanding of some product backlog items.

Backlog Estimating

As noted above, many teams will [estimate during product backlog refinement meetings](#). That is the ideal approach, and is possible if the entire development team participates in backlog refinement.

If only a subset of the development team participates in backlog refinement, team members will meet once per sprint to estimate any new work for which the product owner may need an estimate.

For most teams, these estimating events should be very short. Most teams should not generate or receive a flood of new product backlog items each sprint. Work to be estimated should either be important, new product backlog items or existing items that have been split to better fit in the coming sprint.

I like to do product backlog estimating immediately following a daily scrum, a couple of days before the end of the sprint. That is late enough that most new items will have been identified but in time for the product owner to adjust priorities based on the new information conveyed by the estimates.

I do **not** recommend estimating during sprint planning. That is too late for the product owner to adjust priorities based on the estimates. It also leads to team members spending longer than they should on estimating. So, [don't estimate product backlog items during sprint planning](#).

Prioritization

Before a new sprint begins, the product owner ensures the top of the product backlog has been prioritized. According the [Oxford American Dictionary](#), prioritize means “to put tasks, problems, etc. in order of importance, so that you can deal with the most important first.”

This means it is not sufficient for a prioritization to merely say, “They’re all required.” Or, as one product owner told me, “They’re called requirements for a reason—they’re required.”

In most cases, there will not be an official prioritization meeting or ceremony. Rather, this is something the product owner does alone, often following conversations with stakeholders to understand their needs and desires.

Prioritization should happen as late as possible in the sprint, while making sure it’s done before the next sprint. This will often mean doing it on the last day or two of the sprint.

Usually prioritization is not time consuming. This is because the product owner is typically fine-tuning priorities based on progress and learning from the current sprint rather than performing an outright re-prioritization of an entire product backlog.

When Do You Conduct These Events?

When does your team conduct these events? Are there other events you’d recommend other teams do? Are your participants the same as I’ve described? Please share your thoughts in the comments below.

[Grab My Scrum Event Infographics](#)

Posted: January 3, 2023

Tagged: sprinting, meetings, ceremonies

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

Agile Mentors Podcast: Recap of Opening Series on Scrum

Recapping our agile podcast's initial launch series of topics

Sep 27, 2022

[Read](#)

Seven Tips for Running a Virtual Daily Scrum

How do you effectively facilitate a daily scrum with a remote team?

May 03, 2022

[Read](#)

How Implementation Intentions Help My Sprints

Planning an exact time to do something within a sprint helps make sure you get the important stuff ...

Mar 22, 2022

[Read](#)

Top 7 Ways to Get Stakeholders to Attend Sprint Reviews

Poorly attended sprint reviews cause real problems. Fortunately there are easy fixes.

Mar 08, 2022

[Read](#)

Should You Re-Estimate Unfinished Stories?

We avoid having unfinished work, but it sometimes happens. Here's what to do.

Jun 15, 2021

[Read](#)

Should Your Team Adopt No-Meeting Weeks?

It can be very tempting to schedule a week without any meetings. But there are better ideas.

Dec 08, 2020 • 12 Comments [Replies](#)

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

by
Mike Cohn **Tagged:** product backlog, estimating, story points, backlog
122 Comments

Story points are a unit of measure for expressing an estimate of the overall effort that will be required to fully implement a [product backlog](#) item or any other piece of work.

When we estimate with story points, we assign a point value to each item. The raw values we assign are unimportant: Some teams use a [modified fibonacci sequence](#) (1, 2, 3, 5, 8, 13); others use a doubling sequence (1, 2, 4, 8, 16).

What matters are the *relative values*. A [user story](#) that is assigned two story points should be twice as much effort as a one-point story. It should also be two-thirds the effort of a story that is estimated as three story points.

Instead of assigning 1, 2 and 3, that team could instead have assigned 100, 200 and 300. Or 1 million, 2 million and 3 million. It is the ratios that matter, not the actual numbers.

One of the [main reasons story points are so valuable](#) is that they allow team members with different skill levels to communicate about and agree on an estimate. Instead of arguing about how long it might take each team member personally to do something, teams instead can quickly say that this user story is about twice or three times as much effort as that user story. With story points, it's all relative.

How to Calculate Story Points in Agile

The best definition of story points is that they represent the **effort** to develop a user story or product backlog item.

Effort is a question of time: how long it will take to finish something. Many factors go into determining effort, including

- The amount of work to do
- The complexity of the work
- Any risk or uncertainty in doing the work

When estimating with story points, many things come into play: complexity, effort, risk, and volume. But ultimately, story points are an estimate of effort.

Let's see how each factor impacts the effort estimate given by story points. For each factor that goes into choosing story points, examples are provided to help increase understanding.

The Amount of Work to Do

Certainly, if there is more to do of something, the estimate of effort should be larger. Consider the case of developing two web pages. The first page has only one field and a label asking to enter a name. The second page has 100 fields to also simply be filled with a bit of text.

The second page is no more complex. There are no interactions among the fields and each is nothing more than a bit of text. There's no additional risk on the second page. The only difference between these two pages is that there is more to do on the second page.

The second page should be given more story points. It probably doesn't get 100 times more points even though there are 100 times as many fields. There are, after all, economies of scale and maybe making the

second page is only 2 or 3 or 10 times as much effort as the first page.

Risk and Uncertainty

The amount of risk and uncertainty in a product backlog item should affect the story point estimate given to the item.

If a team is asked to estimate a product backlog item and the stakeholder asking for it is unclear about what will be needed, that uncertainty should be reflected in the estimate.

If implementing a feature involves changing a particular piece of old, brittle code that has no automated tests in place, that risk should be reflected in the estimate.

Complexity

Complexity should also be considered when providing a story point estimate. Think back to the earlier example of developing a web page with 100 trivial text fields with no interactions between them.

Now think about another web page also with 100 fields. But some are date fields with calendar widgets that pop up. Some are formatted text fields like phone numbers or Social Security numbers. Other fields do checksum validations as with credit card numbers.

This screen also requires interactions between fields. If the user enters a Visa card, a three-digit CVV field is shown. But if the user enters an American Express card, a four-digit CVV field is shown.

Even though there are still 100 fields on this screen, these fields are harder to implement. They're more complex. They'll take more time. There's more chance the developer will make a mistake and be required to back up and correct it.

This additional complexity should be reflected in the estimate provided.

Consider All Factors: Amount of Work, Risk and Uncertainty, and Complexity

It may seem impossible to combine three factors into one number and provide that as an estimate to take to [sprint planning](#). It's possible, though, because effort is the unifying factor.

First, Scrum team members consider how much effort will be required to do the amount of work described by a product backlog item.

Next, these agile teams consider how much effort to include for dealing with the risk and uncertainty inherent in the product backlog item. Usually this is done by considering the risk of a problem occurring and the impact if the risk does occur. So, for example, more will be included in the estimate for a time-consuming risk that is likely to occur than for a minor and unlikely risk.

Finally, teams must also consider the complexity of the work to be done. Work that is complex will require more thinking, may require more trial-and-error experimentation, perhaps more back-and-forth with a customer, may take longer to validate and may need more time for mistake corrections.

During agile estimation, all three factors must be combined into one measure of effort.

Remember the Definition of Done

A story point estimate must include everything involved in getting a product backlog item all the way to done.

If a team's [definition of done](#) includes creating automated tests to validate the story (and that would be a good idea), the effort to create those tests should be included in the story point estimate.

Scrum, Story Points, and Conversations

Conversations are an essential component of agile estimating. Even with thought exercises like [story points as buckets](#), team members often don't agree at first on how much effort a story will be.

These varying estimates can spark illuminating conversations between team members and with product owners about acceptance criteria/conditions of satisfaction, approach, and other factors that can affect how much effort it will take to complete an item. Talking about a product backlog item increases the team's understanding of the work, and can reveal gaps and assumptions that the product owner can investigate.

The power of these conversations is one of the reasons I recommend [planning poker](#). Planning poker is a fun way to estimate, and it's also a way to keep each person's estimate private until the team members all reveal their cards. Individual estimates mean less bias in the numbers and, ultimately, estimates that are more accurate.

Once the team has agreed on an estimate, it assigns story points to the backlog item. That story point estimate is later used in calculating a team's [average sprint velocity, capacity, and more](#).

Story points can be a hard concept to grasp. But the effort to fully understand that points represent effort—as impacted by the amount of work, the complexity of the work and any risk or uncertainty in the work—will be worth it.

[Learn More](#)

Posted: December 6, 2022

Tagged: product backlog, estimating, story points, backlog

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of User Stories Applied for Agile Software Development, Agile Estimating and Planning, and Succeeding with Agile as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

Don't Equate Story Points to Hours

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • 39 Comments [Read](#)

Epics, Features and User Stories

I've been getting more and more emails lately from people confused about the difference between ...

Nov 08, 2022 • 93 Comments [Read](#)

For Better Agile Planning, Be Collaborative

The best plans are created by developers and stakeholders working together.

Jul 12, 2022 [Read](#)

Relationship between Definition of Done and Conditions of Satisfaction

I'd like to clarify the relationship between two important concepts: a team's Definition of Done ...

Jun 28, 2022 • 38 Comments [Read](#)

Four Reasons Agile Teams Estimate Product Backlog Items

Estimating product backlog items provides benefits beyond predicting when a project will be finished.

May 17, 2022 [Read](#)

Automatically Triangulating Estimates in Planning Poker

Comparing estimates during Planning Poker just got easier.

Nov 23, 2021 [Read](#)

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

by

Mike **Tagged:**

Cohn estimating, teams, story points, effort, collaboration, relative

39 estimating, hours

[Comments](#)

I'm a huge proponent of estimating in story points. (You can get a full overview of how to use story points in the video above or from reading [What Are Story Points](#).)

In all of my training and writing about story points, [user stories](#), [planning poker](#), and [agile estimating](#), I've been quite **adamant that story points are about effort**. I've also explained that we talk about that effort in terms of *how long it will take* to finish because that's 1) how we naturally think about the effort involved to do a task and 2) how we can answer questions about when a project can be delivered.

But when I say that agile story points are about effort and that effort is measured in time, it doesn't mean teams should say, "One story point equals eight hours." Nor ask, "One story point is how many hours?" Equating story points to a set number of hours is a bad idea. Don't do it.

Equating hours to story points obviates the primary reason to use story points in the first place: Story points are helpful because they allow team members who perform at different speeds to communicate and estimate the amount of work collaboratively.

Story points work because they are relative units of measure, whether you are estimating with a set of cards, T-shirt sizing, or the Fibonacci series. (For more on why relative estimates are essential, read [The Main Reason to Use Story Points](#).)

Agile Estimation Is Abstract On Purpose

By using story points, agile teams with developers who work at different speeds can agree on estimates. A senior developer might be able to knock out a certain product backlog item in 8 hours, and a more junior developer might take 16 hours to do the same work, but they can both agree that it's a 1-point story.

With that agreement in place, they can look at another story and agree that it will take twice as much effort, so it should be worth two points. Or it's five times as much effort, and should be five points.

Let's look at an example. For simplicity, let's assume the team has two members:

Superstar
Junior

Superstar is more experienced, skilled, and knowledgeable than Junior. This leads to Superstar being four times more productive than Junior. Any task that Junior can complete in four hours, Superstar can complete in one.

This team of 2 has an average velocity of 25 story points per sprint. This leads to them planning to complete the following product backlog items in the coming sprint.

A	10
B	5
C	5
D	5

Because Superstar is four times more productive than Junior, Superstar will be able to complete four times as many points in the sprint. That means Superstar will complete 20 and Junior 5 of the 25 points planned in the sprint.

Junior can work on any of the five-point items and successfully complete it during the sprint. Let's assume Junior chooses item D. That leaves Superstar with items A, B, and C as shown below.

A	10	X
B	5	X
C	5	X
D	5	X
Total	20	5

So what do we tell someone who asks, "How many hours does it take to complete one point?"

If we assume this example is a 1-week, 40-hour sprint, there are 3 possible answers.

Superstar worked 40 hours and delivered 20 points. Therefore, one point takes two hours of work.

Junior worked 40 hours and delivered 5 points. Therefore 1 point takes 8 hours. Note that Junior's number of hours per point is 4 times that of Superstar. This corresponds to the initial assumption that Superstar is 4 times as productive.

Together, they worked 80 hours and completed 25 points. Therefore, 1 point takes 3.2 hours (80/25).

You can see from this example that there is no equivalence between points and hours. You cannot say one point equals such-and-such number of hours. For Superstar, a point is 2 hours, for Junior it's 8 hours, and for the team it is 3.2 hours.

But if the team doesn't listen to me, and they define a point as being equal to 3.2 hours, Junior and Superstar will not be able to agree on estimates because they produce such dramatically different results in 3.2 hours.

With story points, on the other hand, everyone can talk about and estimate the work, and the estimate will be accurate no matter which developer works on the story. In this way, story points are still about effort, but the amount of time it takes to complete each point is not fixed at the same amount for all team members.

Equating Story Points to Hours Complicates Thinking

The second problem with equating story points to a set number of hours is that team members no longer think abstractly. If someone instructs team members that one point equals eight (or any number of) hours, the benefits of estimating in an abstract but relatively meaningful unit like story points are lost.

When you try instead to convert story points to hours, you suddenly initiate an hours-to-story-points calculator in every team member's head. When told to estimate the effort required for a story with a specific time per point in mind, the team member will mentally estimate first using the number of hours and then convert that estimate to points.

So in our first example, a senior developer who could complete a story in eight hours would call a product backlog item a one-point story ($8/8=1$ point). A junior developer who might take sixteen hours to do the work would call that same product backlog item a two-point story ($16/8=2$ points). Mathematically, they'd both be right, but they'd be miles away from each other in terms of agreeing on an estimate.

When story points are tied to a certain number of hours, story points are no longer relative. Story point estimation becomes entirely dependent on who is doing the work.

If someone in your company wants to start translating story points to hours, just stop calling the units points and use the label of hours or days instead. Calling them points when they're really just hours introduces needless complexity (and loses one of the main benefits of points: [team members with different skill levels have a common unit of measure](#)).

The Relationship Between Story Points and Hours

So is there a relationship of agile story points to hours? Yes. Suppose for some reason you have tracked how long every one-story-point story took to develop for a given team, and stored it in a story-points-to-hours table. If you graphed that data you would have something that would look like this:

This shows that some stories took more time than others and some stories took less time, but overall the amount of time spent on your one-point stories takes on the shape shown.

Now suppose you had also tracked the amount of time spent on two-point user stories. Graphing that data as well, we would see something like this:

Ideally the two-point stories would take twice as long as the one-point stories. That's unlikely to be exactly the case, of course. But a team that does a good job of estimating will be sufficiently close for reliable plans to be made from their estimates based on their average team [velocity](#).

What these two figures show us is that the relationship between points and hours is a *distribution*. One point equals a distribution with a mode of x , two points equals a distribution with a mode of $2x$, and so on.

By the way, notice that I've drawn the distributions of one- and two-point stories as having overlapping tails. It is very likely that some of the most time-consuming one-point backlog items take longer than some of the shortest two-point items. After all, no team can estimate with perfect insight, especially at the story point level.

So, while the tails of the one- and two-point distributions will overlap, it would be extraordinarily unlikely that the tails of, say, the one- and thirteen-point distributions will overlap (I'm assuming here that you are using a modified fibonacci sequence for your story points, but you could use any set of numbers).

Why This Matters

Some agile teams define the relationship between story points and hours as an equivalence. That is one point equals some number of hours. And by extension, two points is twice that number of hours and so on.

This is a mistake, and makes points irrelevant because they simply become a translation of hours. Mapping story points to hours makes it impossible for team members who produce their work at different rates to agree on estimates. Teams that convert Jira story points to hours through a fixed equivalence (such as one point equals eight hours) will end up with inaccurate plans.

These problems recede when teams understand that the relationship between story points and hours is a distribution. That is, one-point items take from x to y hours. And two-point backlog items take from about $2x$ to $2y$ hours.

So How Many Hours Is a Point?

When doing agile estimating, converting story points to hours through a simple one point equals x hours formula will result in misleading answers that are overprecise. When stakeholders tell us things like, "translate all those crazy agile fibonacci story points to hours so I know what it means" they want merely to know how to interpret the story points we tell them.

We can provide that understanding using velocity. Suppose stakeholders want to know how long a 5-point backlog item will take and that our team's average velocity is 20. We can tell the stakeholders that the five-point item is about one-fourth of the team's total capacity for the sprint.

Going Further

If you want to ensure you understand story points, I suggest this on-demand video course on [Estimating with Story Points](#).

Download Scrum Master Guide

Get a free copy of Situational Scrum Mastering: Leading an Agile Team

[Get my free guide now!](#)

Posted: November 22, 2022

Tagged: estimating, teams, story points, effort, collaboration, relative estimating, hours

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

What Is Cross-Functional Collaboration in Agile?

Perhaps the most prevalent and persistent myth in agile is that a cross-functional team is one on ...

Feb 14, 2023 • 49 Comments [Read](#)

What Are Agile Story Points?

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

For Better Agile Planning, Be Collaborative

The best plans are created by developers and stakeholders working together.

Jul 12, 2022 [Read](#)

Relationship between Definition of Done and Conditions of Satisfaction

I'd like to clarify the relationship between two important concepts: a team's Definition of Done ...

Jun 28, 2022 • 38 Comments [Read](#)

Scrum, Remote Teams, & Success: Five Ways to Have All Three

In a remote-first world, how does an agile team thrive working in a virtual, distributed way?

Jun 07, 2022 [Read](#)

Four Reasons Agile Teams Estimate Product Backlog Item

Estimating product backlog items provides benefits beyond predicting when a project will be finished.

May 17, 2022 [Read](#)

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

by

Mike Tagged:

Cohn product backlog, user stories, features, user story detail, epic,

93 theme

Comments

I've been getting more and more emails lately from people asking, "What is an epic, a feature and a user story in agile?" "Can you give me some epic, feature, and user story examples?" "Can you tell me how to write epics, features and user stories?". "Epics? User Stories? What's the Difference?"

So in this article, let's cover some basic—but very helpful—territory by explaining some user story-related language.

Stories, themes, epics, and features are merely terms agile teams use to help simplify some discussions. Some of these terms date back to the days of Extreme Programming (XP) teams, but the terms are used in newer ways now. I'll start with industry-standard definitions, and I'll comment on the changes as well.

Epic vs. User Story

When it comes to understanding the difference between epics and user stories it helps to understand what each is.

A *user story* is simply something a user wants. For our purposes here, we can think of a user story as a bit of text saying something like, "Paginate the monthly sales report" or, "Change tax calculations on invoices." There is more to [user](#)

[stories](#) than just text written on an index card or typed into a tool, but let's keep it simple here.

User stories are the most common form of product backlog item for agile teams.

During sprint or iteration planning, user stories are moved from the product backlog to the sprint backlog.

Many teams have learned the benefits of writing user stories in the form of: "As a [type of user] I [want this thing] so that [I can accomplish this goal]." Check out the [advantages of that user story format](#). Even within the format, though, we're free to customize user stories.

As defined by the XP teams that invented user stories, an *epic* is a large user story. There's no magic threshold at which we call a particular story an epic. For our purposes in agile and Scrum, epic just means *big user story*.

I like to think of this in relation to movies. If I tell you a particular movie was an "action-adventure movie" that tells you something about the movie. There's probably some car chases, probably some shooting, and so on. The term I used tells you this even though there is no universal definition that we've agreed to follow; no one claims an action-adventure movie must contain at least three car chases, at least 45 bullets must be shot, and so on.

While epic is just a label we apply to a large story, calling a story an epic can sometimes tell us how refined they are in the backlog.

Suppose you ask me if I had time yesterday to create user stories about the monthly reporting part of the system. "Yes," I reply, "but they are mostly epics." That tells you that while I did write them, most are still pretty big chunks of work, too big in fact to be brought directly into a sprint or iteration. Remember that the Scrum framework doesn't say anything about epics, stories, and themes. These terms come from XP.

Theme vs. User Story

The team that invented user stories used the word theme to mean a collection of user stories. We could put a rubber band around that group of stories about monthly reporting and we'd call that a theme.

Sometimes it's helpful to think about a group of stories, so we have a term for that. Sticking with the movie analogy above, in my DVD rack I have filed the James Bond movies together. They are a theme or grouping.

Feature vs. User Story

One more term worth defining is *feature*. This term was not used by the original user stories team, which has led to *feature* being applied to different things in different organizations and teams.

Most common is that a feature is a user story that is big enough to be released or perhaps big enough that users will notice and be happier. Many teams work with user stories that are very small. Some of those

teams find it useful to have a term they can apply to stories that are big enough to release on their own.

I'm not a big fan of the extra complexity one more term brings, and I find it a demotivating waste of time when teams get caught up in arguing whether a certain story is a feature or just a story, for example.

If you want to use *feature*, I recommend using it as a tag that can be applied to any item in your product backlog. A friend posts photos on Instagram of amazing meals he makes. He tags them with #yum. Do the same with #feature if you find it helps people work with your product backlog.

Confused? Your Software May Use These Terms Differently

If at this point you want to tell me I'm wrong, remember that I am describing these terms as defined by the team that invented user stories.

Some of the product backlog tool vendors use the terms differently. Jira, for example, uses epic to mean a group of user stories rather than a single, big user story. I do not know if this was a mistake on their part or not. But, a well-known vendor lock-in strategy is to manipulate the vocabulary so users think all other tools use the words incorrectly.

Here's the good news: While the ideas behind the terms can be useful, [the terms themselves are not very important](#). Theme, epic, feature, and user story are not terms with important specific meanings like pointer to a programmer or collateralized debt obligation to whomever deals with that.

Don't fight your tool—whatever term your software uses to mean small story (user story) group of stories (theme) or big story (epic)e is the term you should use.

Visualizing the Differences

The following illustration of a product backlog makes clear the difference between story, epic, and theme. Each box represents a product backlog item with the boxes drawn to show the size of the different items. Some items are small and thus quick to develop. Other items will take longer and are shown as bigger boxes.

The big items are epics. One item is maybe big enough that some on the team would call it epic, but not so big that everyone agrees. This reflects the fact that there is no precise threshold at which an item becomes an epic.

The illustration also shows a theme, which is depicted as a box grouping a set of stories together. Themes are not necessarily larger than epics. A theme can be larger than an epic if, perhaps, you have a theme comprising eight medium-sized stories. But a theme can be smaller than an epic, as would be the case if you have a theme of a dozen simple spelling errors to correct.

And since feature usually means an item that can be released on its own, those are shown in green.

An Example of How these Words Are Useful

Let's suppose we're building some sort of financial system that will include a set of reports. Since we're just starting development, we're not too concerned yet with exactly which reports those will be. But we don't want to risk forgetting we need to develop the reports, so we write a product backlog item as the user story:

As an authorized user, I can run reports so that I can see the financial state of the organization.

Not only is this a user story, it's an epic because it's too big to fit in one sprint or iteration. There are almost certainly going to be more reports needed than can be done in a single sprint or iteration. So we call this an epic.

Our reporting story/epic just stays as it is on the product backlog for awhile, perhaps a few months. One day the product owner decides that the team will work on these reports in the next iteration. That means the epic is split into a set of smaller stories, perhaps something like:

As an authorized user, I can run a profit and loss report...
As an authorized user, I can run a cash flow report...
As an authorized user, I can run a transaction detail report...

and more. Let's say we identify ten reports. A few days after splitting out these ten stories from their parent

epic, the product owner announces a shift in priorities. Instead of working on reports, the team will be working on something completely different.

The product backlog could now look cluttered because there are ten small reporting stories where there used to be a single epic. To reduce this clutter, consider grouping the ten reporting stories into a single theme labeled “Reporting.” If your software tool supports this, it can make it easier to look through a backlog because you’d see one Reporting theme rather than ten distinct reporting stories.

How Do You Use These Terms?

How do you use epic, theme, story, and feature? Does your tool support the definitions you use or have you had to shift your language to match the tool? Please share your thoughts in the Comments section below.

[Read more](#)

Posted: November 8, 2022

Tagged: product backlog, user stories, features, user story detail, epic, theme

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding

member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

User Stories: How to Create Story Maps

Story maps help to create a shared understanding of the product, visualize user needs, and elicit ...

Mar 14, 2023

[Read](#)

How to Run a Successful User Story Writing Workshop

What you need to know to conduct a story-writing workshop with your team.

Feb 28, 2023 • 3 Comments [Read](#)

What Are Agile Story Points?

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

Advantages of User Stories over Requirements and Use Cases

At the surface, user stories appear to have much in common with use cases and traditional ...

Oct 05, 2022 • 41 Comments [Read](#)

Relationship between Definition of Done and Conditions of Satisfaction

I'd like to clarify the relationship between two important concepts: a team's Definition of Done ...

Jun 28, 2022 • 38 Comments [Read](#)

Four Reasons Agile Teams Estimate Product Backlog Item

Estimating product backlog items provides benefits beyond predicting when a project will be finished.

May 17, 2022

[Read](#)

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile Assessment		

by

Mike Tagged:

Cohn humor, characteristics, agile roles, fun, halloween

Comments

It's time to start thinking about an appropriate costume you can wear to the many agile-themed Halloween parties you'll attend. In this post I'll share some last-minute costume ideas you can pull together whether you're a Scrum Master, product owner, or even a stakeholder on an agile project.

This run-down of Halloween costumes for agile teams started out just for fun. But then I got to thinking, it could also be a creative (and informative) way to run a Halloween retrospective: Agile team members could come dressed to represent the role they played in the past sprint.

Or you could try a Halloween agile retrospective for the project as a whole: and invite stakeholders, management, and team members to dress as the roles they have experienced, or wish they could experience, so far. Or you can just enjoy some light-hearted insights into the Scrum roles.

Auto Mechanic

Let's start with a Scrum Master costume. An auto mechanic costume would be great for a [Scrum Master](#). Like a mechanic, a Scrum Master troubleshoots and fixes problems. A skilled and experienced mechanic can often identify issues before they become problems ("your brake pads are getting thin"). A Scrum Master can be thought of as tuning up a team to achieve its best possible performance.

Costume: Grab some coveralls, maybe smear some grease on them, and either carry some tools or wear a toolbelt.

[Ghost](#)

Stakeholders can sometimes act like ghosts.

Some people say ghosts don't exist. My friend Jim Harold says otherwise on his [podcast](#). But the type of ghost I'm referring to is the [stakeholder who doesn't show up at sprint reviews](#).

Sometimes stakeholders tell us the project is important and get frustrated if the team misses a goal. But by never attending iteration reviews, effectively being as invisible as a ghost, these stakeholders' actions indicate the project must not be that important after all.

Costume: A simple white sheet with two eye holes.

[Tightrope Walker](#)

[Product owners](#) are often forced to prioritize competing needs from different stakeholders. Stakeholders may represent different geographic regions, departments within an organization, or they may just have different views of a product's future.

Regardless of the reason, stakeholders seek to influence a product owner into building what they need, which might be very different from what the customer or other stakeholders are looking for.

In these situations, the product owner is a tightrope walker balancing competing demands and striving not to fall by leaning too far to either side. One way to help maintain balance is for product owners to [treat stakeholders as a second team](#), rather than as individuals.

Costume: As long as you carry a long pole for balance, you can probably wear whatever you want, but some spandex might be appropriate.

[Golf Caddy](#)

A good Scrum Master is like a golf caddy. Everything a caddy does is to help the player—recommending clubs, carrying the clubs, observing other players—all these aids help a golfer focus on hitting the ball well. Scrum Masters emulate this by similarly removing all impediments to a team's progress.

After the final putt is made, cheers go up for the winning golfer as the caddy stands by, unnoticed by most. The same is true of Scrum Masters. When their teams succeed, praise is heaped on the team. But astute observers know how much the Scrum Master did behind the scenes to help the team succeed.

Costume: Channel Rodney Dangerfield from *Caddyshack*, or wear your loudest polyester pants and borrow your brother-in-law's old golf clubs.

[Bartender](#)

A good Scrum Master listens to detect when a team is having a problem, and then fixes that problem. A bartender listens to patrons' problems and...OK, offers solace at best, but doesn't fix the problem. So wear a bartender costume if you want to dress up like a bad Scrum Master.

Costume: A white apron perhaps with a holster belt that holds a bottle on each side will work nicely. If you want to go beyond the minimum, add a [vest or suspenders with flair](#).

Ted Lasso

Instead of dressing like a bad Scrum Master, though, consider dressing as a great agile coach: Ted Lasso. As a coach, Lasso exhibits many traits we'd all do well to emulate: empathy for his team members, great listening skills, willingness to experiment, lack of an outsized ego, seeking to make his team the best it can be, and more.

Costume: A blue tracksuit or pullover with sunglasses. Don't forget to either grow a mustache or buy a stick-on one. Closer to Halloween, you might need to buy the mustache.

Alfred Pennyworth

Batman would be lost without his trusted butler, Alfred. A retired actor and former special agent of the British government, Alfred does whatever Batman needs, usually without Batman even having to ask. Alfred may be the truest example possible of a [true servant leader](#).

Costume: This is a great DIY costume. Alfred is always impeccably dressed, so put on your best suit and tie a full Windsor knot—and quickly rehearse your English accent.

A Couple's Costume: A Chicken and a Pig

A story from the early days of Scrum involved a [chicken and a pig](#) who wanted to open a breakfast restaurant. The pig declined the opportunity, knowing that in producing a ham-and-egg breakfast he'd be committed, the chicken would be merely involved.

The story was told long ago to say that product owners were merely involved in sprints. They weren't committed like Scrum Masters and team members. I never agreed with that view, because in my experience the most successful teams have product owners who act every bit as committed as the developers.

But, probably because the joke is at least slightly amusing, chickens and pigs still occasionally come up in Scrum discussions. This makes them the perfect couple's or group Halloween costume.

Costume: Go all out and make (or buy) chicken and pig costumes. Or keep it simple and get a pig snout on an elastic band. Supplement it with a pink hoodie if you want. The chicken can wear a white or yellow hoodie and a plastic beak.

What Are You Wearing This Year?

I hope I've given you a few team costume ideas for work, and some group costume ideas for all of the Halloween parties, costume contests, and treat-or-treating outings you're sure to be invited to. And I hope that I've also given you some new things to think about with the various Scrum roles or planted the seeds for some new holiday-themed retrospective ideas.

What other ideas do you have? What costume would you wear to an agile Halloween party? Please share your thoughts in the comments below.

[Take the Assessment](#)

Posted: October 25, 2022

Tagged: humor, characteristics, agile roles, fun, halloween

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

[Announcing Scrum Prime](#)

If you really want to speed up your team, sign up now for Scrum Prime and have everything delivered ...

 Apr 01, 2020 • 17 Comments [Read](#)

[Announcing Online Dating Just for Scrum Masters and Product Owners](#)

Announcing the world's only dating website for Scrum Masters and product owners.

Apr 01, 2019 • 45 Comments [Read](#)

[Help Buy Scrum a Design Phase](#)

Scrum has never had a design phase. Perhaps it's time to fix that. Here's a humorous take on how to ...

Apr 02, 2018 [Read](#)

[Advice for Interviewing ScrumMasters](#)

Interviewing Scrum Masters can be difficult because the job is harder than most to turn into a ...

Nov 03, 2015 • 46 Comments [Read](#)

[Introducing the LAFABLE Process for Scaling Agile](#)

Over the last year or so, scaling agile and Scrum have become very popular topics. A variety of ...

Apr 01, 2014 • 68 Comments [Read](#)

[Top Ten Gifts For the ScrumMaster In Your Life](#)

If you're like me, you're late in starting your shopping this year, whether it's for Hanukkah, ...

Dec 09, 2013 • 10 Comments [Re:](#)

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

[Advantages of User Stories over Requirements and Use Cases](#)

At the surface, user stories appear to have much in common with use cases and traditional ...

Oct 05, 2022 • 41 Comments

[Read](#)

[Agile Mentors Podcast: Recap of Opening Series on Scrum](#)

Recapping our agile podcast's initial launch series of topics

Sep 27, 2022

[Read](#)

[Elements of Agile: An Agile Assessment Tool for Iterative Improvements](#)

New to agile, or needing an agile re-boot? We've got a new no-cost assessment and actionable ...

Sep 13, 2022

[Read](#)

[From Project Manager to Scrum Master -3 Tips for Making the Transition](#)

What does a good project manager need to do to become a great Scrum Master?

Aug 23, 2022

[Read](#)

[7 Ways to Get and Improve Fast Feedback](#)

Here's how to get the rapid feedback you need to ensure you build the right product.

Jul 26, 2022

[Read](#)

[For Better Agile Planning, Be Collaborative](#)

The best plans are created by developers and stakeholders working together.

Jul 12, 2022

[Read](#)

[Relationship between Definition of Done and Conditions of Satisfaction](#)

I'd like to clarify the relationship between two important concepts: a team's Definition of Done ...

Jun 28, 2022 • [38 Comments](#)

[Read](#)

[3 Ways to Help Agile Teams Plan Despite Uncertainty](#)

We might not like ambiguity, but it's a fact of life. Find out how to plan with uncertainty in mind.

Jun 21, 2022

[Read](#)

[Scrum, Remote Teams, & Success: Five Ways to Have All Three](#)

In a remote-first world, how does an agile team thrive working in a virtual, distributed way?

Jun 07, 2022

[Read](#)

[Announcing the New Agile Mentors Podcast](#)

Don't miss this new podcast from the Scrum and agile experts at Mountain Goat Software.

May 31, 2022

[Read](#)

[Four Reasons Agile Teams Estimate Product Backlog Items](#)

Estimating product backlog items provides benefits beyond predicting when a project will be finished.

May 17, 2022

[Read](#)

[Agile Spikes Deliver Knowledge So Teams Can Deliver Products](#)

On agile projects, a spike is a time-boxed research activity that helps teams make better decisions ...

May 11, 2022 • 48 Comments

[Read](#)

[NEWER POSTS](#) [OLDER POSTS](#)

Learn About Agile **Agile & Scrum Training** **More...**

- | | | |
|----------------------------|--------------------------|--------------------|
| New to Agile and Scrum? | Online Certifications | About Us |
| Scrum | Video Courses | Contact Us |
| User Stories | In-Person Certifications | Our Students |
| Planning Poker | On-Site Courses | Blog |
| Agile Software Development | Training Schedule | Books by Mike Cohn |
| Agile Project Management | Compare Courses | Agile Tools |
| Agile Presentations | PDUs and SEUs | |
- Mountain Goat on YouTube
Agile Mentors Podcast
Elements of Agile Assessment

Learn About Agile

[An Introduction to Agile and Scrum](#)

The Scrum Framework

User Stories

Planning Poker

Agile Software Development

Agile Project Management

[Scrum Foundations Video Series](#)

All the foundational knowledge of Scrum including: the framework, values, different roles, meetings, backlogs, and improving efficiency & quality.

[Intro to Agile and Scrum](#)

New to agile and Scrum? Or just wanting a quick refresher? You've come to the right place. Learn about agile, and popular frameworks like Scrum, SAFe, and Kanban that share similar values and principles.

Agile is a lighter weight approach to project management and product development. It differs from traditional waterfall or other phased approaches that rely on upfront research, multiple stages, and gated handoffs, which usually result in products that take a year or more to reach their users.

An [agile approach to product development is appropriate](#) on any urgent project with significant complexity and novelty. [Contrary to the many myths](#) out there, old and new agile methodologies allow for managers, plan, design, allow for specialists, are for products beyond software, and embrace change (but not whenever anyone wants).

Popular approaches to agile development (Scrum, Kanban, and XP) all share a similar set of principles and values.

How Long Has Agile Been Around?

The term agile was first applied to Scrum and similar development processes in early 2001 with the publication of the Agile Manifesto. You can think of agile as an umbrella term that encompasses other processes, such as Scrum, SAFe, Extreme Programming, Adaptive System Development, DSDM, Feature Driven Development, Kanban, Crystal and more.

Agile vs Scrum: What's The Difference

Of all the agile methodologies, Scrum is the most widely used framework. One way to think about the relationship between agile and Scrum is to use a refrigerator metaphor.

If your refrigerator were to break, you would go to an appliance store and be shown various refrigerators.

You might see refrigerators from Maytag, General Electric, Viking, Whirlpool, Frigidaire, SubZero, Bosch and so on. You would leave the store, let's say with a Maytag, because its unique features best fit your needs. In the same way that Maytag is a brand of refrigerator, the Scrum process is a brand of agile.

Unlike refrigerators, however, you can customize a framework to better fit your team. You can choose to primarily use Scrum, for instance, but also incorporate some of the desirable features of XP, Kanban, and others.

For example, many teams that use Scrum also employ test-driven development and pair programming, both of which are components of Extreme Programming.

This flexibility is a large part of the appeal, and the reason why I don't typically say Scrum vs Agile vs Waterfall or Kanban vs Scrum. It's not a competition; it's cooperation and collaboration.

Animosity between agile frameworks is unwarranted. Most people who choose any agile process (scrum or otherwise) them want the same thing: To be more productive, do better work, and feel fulfilled (and respected) by the work they do.

It's like choosing somewhere to go out and eat. The ultimate goal is to enjoy a meal out with good food (and probably good company). You might have a favorite restaurant that you know you'll enjoy but you probably don't eat every meal there.

Perhaps you love an Italian restaurant, but sometimes you're in the mood for Thai food. Or some nights you're in the mood for some live music but on others, a quiet atmosphere is what you need.

Choosing an agile method is not exactly the same as choosing somewhere to eat, the point is that it's doubtful any organization or team wants to use exactly the same approach every single time.

How Agility Helps Team Manage Change

Waterfall and other non-agile processes aren't inherently bad. If your process is working, by all means stick with it. The reality, however, is that the rate of change has accelerated dramatically over the past 30 years and especially over the past 10. Product development cycles that were acceptable 10 years ago would be laughable now.

All signs point to this quickening trend continuing. Today's "fast enough" will likely not be fast enough tomorrow. In order to remain competitive, companies developing software need a process that can help them keep up with the accelerating rate of change.

Agile frameworks help teams deliver products sooner, and at lower costs, giving them a competitive advantage in a fast-paced market. Plus, organizations are able to rapidly adapt to meet the true needs of the market.

A hallmark of agile projects are the self-organizing, [cross-functional teams](#) that are empowered to achieve specific business objectives. These teams work with a focus on rapid and frequent deliverables of partial solutions that can be evaluated and used to determine next steps.

In this way, solutions are built in an [iterative and incremental](#) manner. Agile methodologies have been shown to deliver higher quality products in less time, resulting in improved customer satisfaction.

Agile Values

Along with 12 agile principles, the [2001 Agile Manifesto](#) defined 4 agile values. Although written specifically about agile software development, these principles and values are applicable to any agile product development methods:

"Individuals and interactions over processes and tools

Working software over comprehensive documentation

Customer collaboration over contract negotiation

Responding to change over following a plan"

The writers go on to say that "while there is value in the items on the right, we value the items on the left more."

Scrum adds five values of its own to this list: commitment, courage, focus, openness, and respect.

Benefits of Agile

Transitioning to a new process is hard. The [benefits of adopting agile](#) must outweigh the cost. Organizations that have made the switch report the following benefits, including these from [Forbes](#):

- Faster feedback cycle
- Embraces reality of constant change
- Easy to adapt and pivot
- Exposes risks early
- Less waste
- More purpose
- Improved predictability

In an [early study on the impacts of moving to an agile process](#), Michael Mah found measurable differences in these areas:

- Higher productivity and lower costs
- Higher quality
- Accelerated time-to-market

A recent [State of Agile report](#) found that 69% of respondents had seen increased collaboration. 51% had seen better alignment to business needs.

And [Scott Ambler's report in Dr. Dobb's Journal](#), also found these benefits:

- Improved stakeholder satisfaction
- Increased job satisfaction
- More engaged employees

Having more engaged employees leads to more productivity gains, initiating a virtuous cycle of continuous improvement.

About Scrum

In a nutshell, [Scrum](#) is a way to manage work on any project that aims to quickly create something new, complex, and somewhat ambiguous.

Scrum happens in a repeating inspect and adapt loop (iteration or sprint) that starts with sprint planning, has daily scrums, and ends with a sprint review and retrospective before beginning again.

Scrum team members include developers (those who are creating the product) a Scrum Master, and a product owner.

Scrum tools include a task board, product backlog (often expressed as [user stories](#)), release burndown chart, and sprint backlog.

Scrum is not a fad. It has been around a lot longer than you might think. The first paper on Scrum ([The New New Product Development Game](#)) appeared in Harvard Business Review in January 1986. Software teams started using Scrum in 1993. Scrum and agile development are quickly becoming the predominant way to work.

In fact, a [recent State of Agile survey](#) found that 4 out of 5 respondents were using an agile process on at least some of their projects. And they aren't just applying these frameworks to IT and software development teams. 61% are adopting agile for company-wide digital transformation.

About Kanban

Kanban is all about workflow: how to visualize it and how to manage it. Taiichi Ohno is credited with starting in Japan at Toyota. Its two distinguishing hallmarks are limiting the amount of work in process and visualizing work through a kanban board (similar to Scrum's taskboard).

One of the main benefits [touted by David Anderson](#) is that Kanban is a better choice for most because it requires no organizational changes to implement. That's a good point, can allow large issues to remain unaddressed because [poor software development outcomes are often a symptom](#) of larger scale changes needed at the organizational level.

Besides, when it comes to Kanban vs Scrum, there's no need for a competition. Some teams choose Scrum; some choose Kanban, some combine them. There are definitely times when [Kanban might be the better choice for your team](#).

XP & Pair Programming

Agile, Scrum, and XP share a set of commonalities but have [some key differences](#) as well.

People often ask, **is pair programming required on an agile team?** Scrum and other frameworks don't require pair programming. But it is likely worthwhile to at least try on some projects and for some Scrum teams.

Pair programming is one of the Extreme Programming (XP) practices. But because it can often be a great idea, it has expanded beyond XP to become a popular agile practice—just as Kanban boards (taskboards) have expanded beyond Kanban.

Every team can experiment with pair programming and determine when it's appropriate for them and their project. It's quite possible that it's not 100 percent of the time, but it's even more likely that it's not 0 percent.

Getting Started with Agile and Scrum

Change begins when an awareness that the status quo could be improved turns into a desire to do something different. All of the awareness and desire in the world, however, won't get you anywhere if you do not also acquire the ability to be agile.

You will need not only to learn new skills but also to unlearn old ones, including:

- New technical skills, such as test automation and design evolution
- How to [think and work as a team](#)
- How to create working product within short timeboxes

Beyond an introduction to agile development and Scrum, training along with on-site coaching or mentoring is usually required. What seems to work best for most companies is some initial training, oriented at creating a willingness to try Scrum and to understanding its core principles.

This general training is usually then followed up with practice-specific training or coaching, such as bringing a test-driven development expert on-site to work hands-on with teams in their code.

Mountain Goat Software offers individual training, private team training, and onsite coaching to help you and your team get started and get better at agile

and Scrum.

To reinforce training, companies often provide opportunities (wikis, informal lunch-and-learns, cross-team exchanges) for teams to share information with each other.

Above all, don't stall, waiting to know all the answers before you start. The best way to develop the ability to do something is to start doing it.

[Take the Assessment](#)

Recommended Resources

- [An Iterative Waterfall Isn't Agile](#)
- [Introducing An Agile Process to an Organization](#)
- [How To Fail With Agile](#)
- [Transitioning to Agile](#)
- [ADAPTING to Agile](#)

Related Courses

[Succeeding with Agile](#)

[Certified ScrumMaster®](#)

[Certified Scrum Product Owner®](#)

[Agile Estimating and Planning](#)

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

Learn About Agile

An Introduction to Agile and Scrum

[The Scrum Framework](#)

Scrum Roles: An Overview

- [Scrum Master Role and Responsibilities](#)
- [Product Owner Role and Responsibilities](#)
- [Scrum Team Role & Responsibilities](#)
- [The Chicken and the Pig](#)

Scrum Activities: An Overview

- [Sprint Planning Meeting](#)
- [Daily Scrum Meeting](#)
- [Sprint Review Meeting](#)
- [Sprint Retrospective](#)

Scrum Tools: An Overview

- [Scrum Product Backlog](#)
- [Sprint Backlog](#)
- [Scrum Task Board](#)
- [Release Burndown Chart](#)

Free Scrum Resources

- [Reusable Scrum Presentation](#)

User Stories

- [Planning Poker](#)
- [Agile Software Development](#)
- [Agile Project Management](#)

Understand Scrum in 60 minutes.

[Watch The Scrum Foundations](#)

First Name

Email Address

[Privacy Policy](#)

What Is Scrum? What Is the Definition of Scrum?

What is Scrum? Scrum is an [agile](#) way to manage work. Every few weeks (typically two to four), teams deliver a fully functional chunk of work (an increment). Teams and the business use the feedback from each delivery to determine what to build next, or how to adapt what they've already built.

Put very simply, Scrum works through a series of events that happen over a defined period of time: that time period is called a sprint. Sprints are short timeboxes during which the team turns ideas into working product. The events then repeat every sprint.

The Scrum Framework

When talking about the Scrum framework, one of the first things to understand is that we purposefully define a Scrum *framework* rather than a Scrum *methodology*, Scrum *model*, or Scrum *development process*. A methodology or process will specify essentially everything about how work should be

performed when using that approach.

An agile framework is different. Unlike a process or methodology, a framework is purposefully incomplete.

For example, Jeff Sutherland's and Ken Schwaber's [Scrum Guide](#) says that a team needs to produce high quality work, but doesn't say how the team is to achieve that.

Teams are the heart of any agile project management framework. Instead of providing complete, detailed descriptions of how everything is to be done on a project, much of it is left up to the team. This is because the team will know best how to solve the problem they are presented.

This is also why the activities of a sprint are described in terms of the desired outcome instead of a set of entry criteria, task definitions, validation criteria, exit criteria (ETVX) and so on, as would be provided in most methodologies.

The definition of Scrum is simple and small. Remember, though, that the parts that are there have been shown to work over and over again. Be very careful, therefore, about deciding to remove something, especially before you have sufficient experience to make very educated guesses about how removing something would affect the other parts.

Scrum Team Roles

Scrum relies on a [self-organizing, cross-functional team](#) of developers. By that, I do not just mean software developers—anyone involved in creating work for the sprint is called a developer.

The team is [self-organizing](#) in that there is no overall team leader who decides which person will do which task or how a problem will be solved. Those are issues that are decided by the team as a whole.

A team is [cross-functional](#), meaning everyone is needed to take a feature from idea to implementation.

A typical team has between five and nine people, but projects can easily scale into hundreds of people divided among many teams. Individuals may join the team with various job titles; those titles are insignificant. Each person contributes in whatever way they can to complete the work of each sprint.

This does not mean that a tester will be expected to re-architect the system; individuals will spend most (and sometimes all) of their time working in whatever discipline they worked before they joined the team. But individuals are expected to work beyond their preferred disciplines whenever doing so would be for the good of the team. In this way, teams develop a deep form of camaraderie and a feeling that “we’re all in this together.”

Development teams are supported by two specific roles. The first is a [Scrum Master](#), who can be thought of as a coach for the team. Scrum Masters help team members use the process to perform at their highest level, and remove impediments to progress.

A good ScrumMaster also shelters the team from outside distractions, allowing team members to focus maniacally during the sprint on the goal they have selected.

A [ScrumMaster](#) differs from a traditional project manager in many ways, including that this role does not provide day-to-day direction to the team and does not assign tasks to individuals.

While the ScrumMaster focuses on helping the team be the best that it can be, the [product owner](#) works to direct the team to the right goal.

The product owner is the project’s key stakeholder and represents users, customers and others in the process. Because of this, many product owners come from product management or marketing. Product owners guide the team toward building the right product.

Product owners do this by creating a compelling vision of the product, and then conveying that vision to the team through an artifact called the [product backlog](#).

The product owner is responsible for prioritizing the backlog, to ensure it’s up to date as more is learned about the system being built and its users.

One way to think of the interlocking nature of these three roles in this agile methodology is as a racecar.

The team of developers is the car itself, ready to speed along in whatever direction it is pointed. The product owner is the driver, making sure that the car is always going in the right direction. And the ScrumMaster is the chief mechanic, keeping the car well tuned and performing at its best.

Scrum Activities

Each sprint begins with a [sprint planning meeting](#), where the product owner presents the top items on the product backlog to the team, and team members figure out how much work they can commit to during the coming sprint.

During each sprint, the team takes a small set of features from idea to fully implemented and tested functionality. At the end, these features are done and could potentially be released to customers.

On each day of the sprint, all team members attend a [daily scrum meeting](#). Daily scrums are a way for team members to synchronize their work and collaborate to move that work to done. The daily meetings last no more than 15 minutes and are intended to give the team a time to share what they worked on the prior day, will work on that day, and identify any impediments to progress.

At the end of a sprint, the team conducts a [sprint review](#) during which the team demonstrates the new functionality to the product owner or any other stakeholder who wishes to provide feedback that could influence the next sprint. It’s critical that the sprint review remains informal and doesn’t become its own task, distracting from the work itself. On some teams, for example, PowerPoint slides are not allowed. Other teams ask attendees to take each feature

for a hands-on test drive .

This feedback loop may result in changes to the freshly delivered functionality, but it may just as likely result in revising or adding items to the work planned for the future.

Another inspect-and-adapt activity is the [sprint retrospective](#) at the end of each sprint. The whole team participates in this meeting. The meeting is an opportunity to reflect on the sprint that has ended and to identify what changes or improvements the team may wish to make to the process itself.

Both of these end-of-sprint activities are consistent with the agile value of continuously improving.

Scrum Artifacts & Tools

Scrum has several artifacts and tools. The primary artifact is, of course, the product itself.

The [product backlog](#) is another tool. The product backlog is the list of the functionality that remains to be added to the product. The product owner prioritizes the backlog so the team always works on the most valuable features first.

The most popular and successful way to create a product backlog is to populate it with [user stories](#), which are short descriptions of functionality described from the perspective of a user or customer.

On the first day of a sprint and during the planning meeting, team members create the [sprint backlog](#). The sprint backlog can be thought of as the team's to-do list for the sprint. The sprint backlog is the list of tasks the team needs to perform in order to deliver the functionality it committed to deliver during the sprint.

Unlike the sprint backlog, the product backlog is a list of features to be built over many sprints.

During the sprint, most teams use a [task board](#) (virtual or physical) to visually represent the flow of sprint backlog work as it moves from To Do, to Doing, to Done.

Though not required, teams often use and create sprint burndown charts and [release burndown charts](#). Burndown charts show the amount of work remaining either in a sprint or a release, and are an effective tool to determine whether a sprint or release is on schedule to have all planned work finished by the desired date.

Visual Introduction to Scrum

To understand how all of the pieces of Scrum work together, let's look at how the essential elements of Scrum are depicted graphically.



On the left, we see the product backlog, which has been prioritized by the product owner and contains everything desired in the product that's known at the time. The product backlog is depicted by a tall stack of large blue cubes.

The two-to-four week sprints are shown by the larger green circle.

At the start of each sprint, the team selects some amount of work from the product backlog and commits to completing that work during the sprint. Part of figuring out how much they can commit to is creating the sprint backlog, which is the list of tasks (and an estimate of how long each will take) needed to deliver the selected set of product backlog items to be completed in the sprint.

The sprint backlog is shown just to the left of the green circle, and is represented by a short stack of small blue cubes.

At the end of each sprint, the team produces a potentially shippable product increment — i.e. working, high-quality software. That increment is shown to the right of the sprint (an output) and represented as a brown package.

Each day during the sprint, team members meet to discuss their progress and any impediments to completing the work for that sprint. This is known as the daily scrum, and is shown as the smaller green circle above the sprint.

FAQs

What Does Scrum Stand For?

Uniquely agile, the Scrum methodology does come with a funny name. People often ask "What does Scrum stand for? It isn't an acronym and doesn't

stand for anything. The term comes from the rugby approach described by Hirotaka Takeuchi and Ikujiro Nonaka in their 1986 "The New New Product Development Game."

Is Scrum a Methodology?

Is Scrum a methodology? No, Scrum is not a methodology. And it isn't called a Scrum model. Models and methodologies give step-by-step guidance about exactly how work should be performed.

Scrum is a framework. It provides a structure to work within and desired outcomes to achieve, but leaves teams room to decide how best to achieve those outcomes in their specific context.

Agile vs Scrum: What's the Difference? Is Scrum Agile?

When it comes to Scrum vs agile or Scrum in agile contexts, it helps to think of agile as an umbrella. Under the agile umbrella are many frameworks: XP, Kanban, Scrum, and more.

So, Scrum is agile. But just because a team is agile, doesn't necessarily mean they are using Scrum project management techniques or a framework that fits the Scrum definition. They could well be using another agile technique.

I started doing agile software development with Scrum more than 20 years ago, and I can tell you that it [works for most complex projects](#). Scrum software development has evolved over the years to encompass all types of products, well beyond software.

Introduction to Scrum Terms

An introduction to Scrum would not be complete without knowing the terms you'll be using.

Scrum team: A typical team has between five and nine people, but projects can easily scale into the hundreds, or just be one-or two-person teams. Everyone on the project works together to complete the set of work they have collectively committed to complete within a sprint. Teams develop a deep form of camaraderie and a feeling that "we're all in this together."

Product owner: The product owner is the project's key stakeholder and represents users, customers and others in the process. The product owner is often someone from product management or marketing, a key stakeholder or a key user.

Scrum Master: The Scrum Master is responsible for making sure the team is as productive as possible. They do this by helping the team use the Scrum process, by removing impediments to progress, by protecting the team from outside, and so on.

Product backlog: The product backlog is a prioritized features list containing every desired feature or change to the product. Note: The term "backlog" can get confusing because it's used for two different things. To clarify, the product backlog is a list of desired features for the product. The [sprint backlog](#) is a list of tasks to be completed in a sprint.

Sprint planning meeting: At the start of each sprint, a sprint planning meeting is held, during which the product owner presents the top items on the product backlog to the team. The team selects the work they can complete during the coming sprint. That work is then moved from the product backlog to a sprint backlog, which is the list of tasks needed to complete the product backlog items the team has committed to complete in the sprint.

Daily scrum: Each day during the sprint, a brief meeting called the daily scrum is conducted. This meeting helps set the context for each day's work and helps the team stay on track. All team members are required to attend the daily scrum.

Sprint review meeting: At the end of each sprint, the team demonstrates the completed functionality at a sprint review meeting, during which, the team shows what they accomplished during the sprint.

Sprint retrospective: Also at the end of each sprint, the team conducts a sprint retrospective, which is a meeting during which the team reflects on how well their process is working for them and what changes they may wish to make for it to work even better.

Recommended Resources

[Introduction to Scrum PPT](#)

[5 Free Agile & Scrum Tools for Project Planning and Prioritizing](#)

Related Courses

Certified ScrumMaster®

Certified Scrum Product Owner®

Scrum Repair Guide

Learn About Agile

New to Agile and Scrum?

Agile & Scrum Training

Online Certifications

More...

About Us

Scrum

Video Courses

Contact Us

User Stories

In-Person Certifications

Our Students

Planning Poker

On-Site Courses

Blog

Agile Software Development

Training Schedule

Books by Mike Cohn

Development

Compare Courses

Agile Tools

Agile Project Management

PDUs and SEUs

Agile Presentations

Mountain Goat on YouTube

Agile Mentors Podcast

Elements of Agile

Assessment

Learn About Agile

An Introduction to Agile and Scrum

The Scrum Framework

[User Stories](#)

Planning Poker

Agile Software Development

Agile Project Management

What is a user story?

A **user story** is a short, simple description of a feature told from the perspective of the person who desires the new capability, usually a user or customer of the system. [User stories typically follow a simple template](#):

As a < type of user >, I want < some goal > so that < some reason >.

Historically user stories were deliberately kept informal, written on index cards or sticky notes, stored in a shoe box, and arranged on walls or tables to facilitate planning and discussion. Their impermanence made it easy to tear them up, throw them away, and replace them with new stories as more was learned about the product being developed.

Nowadays, user stories might just as easily be stored in a Jira issue or Trello board. Don't let the fact that a user story exists in a tool make you any less

willing to discard stories when they are no longer needed!

User stories are designed to strongly shift the focus from writing about features to discussing them. In fact, these discussions are more important than whatever text is written.

What Is a Good User Story?

Agile user stories are composed of three aspects that Ron Jeffries named in 2001 with the wonderful alliteration of card, conversation, and confirmation:

1. **Card**: Written description of the story, used for planning and as a reminder
2. **Conversation**: Conversations about the story that serve to flesh out the details of the story
3. **Confirmation**: Tests that convey and document details that can be used to determine when a story is complete.

User stories have many advantages, but the most important might be that every user story is a placeholder for a future conversation.

How to write a user story

Writing good user stories in Scrum requires an understanding of the basic user story template, a focus on the user or customer, and a clear picture of the desired functionality.

User Story Template

When writing a user story, remember that user stories follow a standard template:

As a < type of user >, I want < some goal > so that < some reason >.

Examples of User Stories

One of the best ways to learn how to write a user story in agile is to see examples. Below is an example user story or two. These are a few real examples of user stories that describe the desired functionality in an early version of the Scrum Alliance website.

As a site member, I can fill out an application to become a Certified Scrum Trainer so that I can teach [Certified Scrum Master \(CSM\)](#) and [Certified Scrum Product Owner \(CSPO\)](#) courses and certify others.

As a trainer, I want my profile to list my upcoming classes and include a link to a detailed page about each so that prospective attendees can find my courses.

As a site visitor, I can access old news that is no longer on the home page, so I can access things I remember from the past or that others mention to me.

As a site visitor, I can see a list of all upcoming "Certification Courses" and can page through them if there are a lot, so I can choose the best course for me.

Note that you don't see any user story, "As a product owner, I want a list of certification courses so that..." The [product owner](#) is an essential stakeholder, but is not the end user/customer. When creating user stories, it's best to be as specific as possible about the type of user.

200 User Stories Examples

See user stories Mike Cohn wrote as part of several real product backlogs.

First Name

Email Address

[Privacy Policy](#)

Who writes user stories?

Who writes user stories? Anyone can write user stories.

Does the product owner write user stories?

It's the product owner's responsibility to make sure a product backlog of agile user stories exists, but that doesn't mean that the product owner is the one who writes them. Over the course of a good agile project, you should expect to have user stories written by each team member.

Also, note that who writes a user story is far less important than who is involved in the discussions of it.

When are user stories written?

User stories are written throughout the agile project. Usually a story-writing workshop is held near the start of the agile project. Everyone on the team participates with the goal of creating a product backlog that fully describes the functionality to be added over the course of the project or a three- to six-month release cycle within it.

Some of these agile user stories will undoubtedly be epics. Epics will later be decomposed into smaller stories that fit more readily into a single iteration. Additionally, new stories can be written and added to the product backlog at any time and by anyone.

Can you show other user story examples?

As a more generic example of writing user stories in Scrum, these are some typical user stories for a job posting and search site:

- A user can post her resume to the web site.
- A user can search for jobs.
- A company can post new job openings.
- A user can limit who can see her résumé.

Examples of Epics

One of the [benefits of agile](#) user stories is that they can be written at [varying levels of detail](#). We can write a user story to cover large amounts of functionality or only a small distinct feature.

Large user stories are less detailed, and are generally known as [epics](#).

Here is an epic agile user story example from a desktop backup product:

- As a user, I can backup my entire hard drive.

Because an epic is generally too large for an agile team to complete in one iteration, it is [split into multiple smaller user stories](#) before it is worked on. The epic above could be split into dozens (or possibly hundreds), including these two example user stories:

- As a power user, I can specify files or folders to backup based on file size, date created and date modified.
- As a user, I can indicate folders not to backup so that my backup drive isn't filled up with things I don't need saved.

How is detail added to user stories?

Detail can be added to user stories in two ways:

- By splitting a user story into multiple, smaller user stories.
- By adding [conditions of satisfaction](#) (acceptance criteria).

When a relatively large story is split into multiple, smaller agile user stories, it is natural to assume that detail has been added. After all, more has been written.

The conditions of satisfaction is simply a high-level acceptance test that will be true after the agile user story is complete. Consider the following as another agile user story example:

As a vice president of marketing, I want to select a holiday season to be used when reviewing the performance of past advertising campaigns so that I can identify profitable ones.

Detail could be added to that user story example by adding the following conditions of satisfaction:

Make sure it works with major retail holidays: Christmas, Easter, President's Day, Mother's Day, Father's Day, Labor Day, New Year's Day.

Support holidays that span two calendar years (none span three).

Holiday seasons can be set from one holiday to the next (such as Thanksgiving to Christmas).

Holiday seasons can be set to be a number of days prior to the holiday.

Do user stories replace a requirements document?

Agile projects, especially Scrum ones, use a product backlog, which is a prioritized list of the functionality to be developed in a product or service. Although product backlog items can be whatever the team desires, user stories have emerged as the best and most popular form of product backlog items.

While a product backlog can be thought of as a replacement for the requirements document of a traditional project, it is important to remember that the written part of an agile user story ("As a user, I want ...") is incomplete until the discussions about that story occur.

It's often best to think of the written part as a pointer to the real requirement. User stories could point to a diagram depicting a workflow, a spreadsheet showing how to perform a calculation, or any other artifact the product owner or team desires.

Recommended Resources

[Advantages of the "As a user, I want" user story template.](#)

[A Sample Format for a Spreadsheet-Based Product Backlog](#)

[Advantages of User Stories over Requirements and Use Cases](#)

[Non-functional Requirements as User Stories](#)

[Introduction to User Stories](#)

Related Courses

Certified Scrum Product Owner®

Better User Stories

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		

Learn About Agile

An Introduction to Agile and Scrum

The Scrum Framework

User Stories

[Planning Poker](#)

Agile Software Development

Agile Project Management

Scrum Foundations Video Series

All the foundational knowledge of Scrum including: the framework, values, different roles, meetings, backlogs, and improving efficiency & quality.

What is Planning Poker?

Planning Poker is an agile estimating and planning technique that is consensus based. To start a poker planning session, the product owner or customer reads one of their [user stories](#) or describes a feature to the estimators.

Each estimator is holding a deck of Planning Poker cards with values like 0, 1, 2, 3, 5, 8, 13, 20, 40 and 100, which is the sequence we recommend. The values represent the number of story points, ideal days, or other units in which the team estimates.

The estimators discuss the feature, asking questions of the product owner as needed. When the feature has been fully discussed, each estimator privately selects one card to represent his or her estimate. All cards are then revealed at the same time.

If all estimators selected the same value, that becomes the estimate. If not, the estimators discuss their estimates. The high and low estimators should

especially share their reasons. After further discussion, each estimator reselects an estimate card, and all cards are again revealed at the same time.

The poker planning process is repeated until consensus is achieved or until the estimators decide that agile estimating and planning of a particular item needs to be deferred until additional information can be acquired.

Our Planning Poker Tool

Did you know we have a specific Planning Poker tool as part of an [Agile Mentors Membership](#)?

If you're a member of our Agile Mentors Community you already have access to a Planning Poker® tool that we developed for you to use with your team.

What members like about this tool is that you can invite unlimited guests into sessions and they **don't** have to be members.

Other highlights include:

- Choose any number sequence you like

- Save favorite user stories

- Import stories from tools such as JIRA and Pivotal Tracker

- Export stories and estimates in CSV format

When should we engage in Planning Poker?

Most teams will hold a Planning Poker session shortly after an initial product backlog is written. This session (which may be spread over multiple days) is used to create initial estimates useful in scoping or sizing the project.

Because product backlog items (usually in the form of user stories) will continue to be added throughout the project, most teams will find it helpful to conduct subsequent agile estimating and planning sessions once per iteration. Usually this is done a few days before the end of the iteration and immediately following a daily standup, since the whole team is together at that time anyway.

How does poker planning work with a distributed team?

A product owner, ScrumMaster or agile coach can log in to our tool and preload a set of items to be estimated. A private URL can then be shared with estimators who log in and join a conference call or Skype session. Agile estimating and planning then proceeds as it would in person.

Does Planning Poker work?

Absolutely. Teams estimating with Planning Poker consistently report that they arrive at more accurate estimates than with any technique they'd used before.

One reason Planning Poker leads to better estimates is because it brings together multiple expert opinions. Because these experts form a cross-functional team from all disciplines on a software project, they are better suited to the estimation task than anyone else.

After completing a thorough review of the literature on software estimation, Magne Jørgensen, Ph.D., of the Simula Research Lab concluded that “the people most competent in solving the task should estimate it.”

Second, a lively dialogue ensues during poker planning, and estimators are called upon by their peers to justify their estimates. Researchers have found that this improves estimate accuracy, especially on items with a lot of uncertainty as we find on most software projects.

Further, being asked to justify estimates has also been shown to result in estimates that better compensate for missing information. This is important on an agile project because the user stories being estimated are often intentionally vague.

Additionally, studies have shown that averaging individual estimates during agile estimating and planning leads to better results as do group discussions of estimates.

How can I get Planning Poker cards?

Planning Poker cards are available in the [Mountain Goat Software store](#). Mountain Goat Software's branded Planning Poker cards are sold at cost as a courtesy to the agile community.

Our full-color cards are the absolute highest-quality cards available anywhere. They are manufactured by the same company that prints many of the world's most popular playing card brands, including Bicycle, Bee, and the World Poker Tour.

We also offer royalty-free licenses to organizations that wish to produce their own cards. The license is available here: <https://www.mountaingoatsoftware.com/agile/planning-poker/license>

Recommended Resources

[How Can We Get the Best Estimates of Story Size?](#)

[The Best Way to Establish a Baseline When Playing Planning Poker](#)

[Don't Average During Planning Poker](#)

[Agile Estimating](#)

Related Courses

[Certified ScrumMaster®](#)

[Certified Scrum Product Owner®](#)

[Agile Estimating and Planning](#)

Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile Assessment		

Learn About Agile

[An Introduction to Agile and Scrum](#)

[The Scrum Framework](#)

[User Stories](#)

[Planning Poker](#)

[Agile Software Development](#)

[Agile Project Management](#)

Scrum Foundations Video Series

All the foundational knowledge of Scrum including: the framework, values, different roles, meetings, backlogs, and improving efficiency & quality.

What is an agile methodology?

An agile methodology—such as Scrum—is a lighter weight approach to software development than many of the traditional approaches.

Agile methodologies feature self-organized teams that are empowered to achieve specific business objectives. Agile methodologies focus on rapid and frequent deliverables of partial solutions that can be evaluated and used to determine next steps.

In this way, solutions are built in an iterative and incremental manner. Agile methodologies have been shown to deliver higher quality products in less time, resulting in improved customer satisfaction.

Is it best to run an agile software development pilot project first?

It is not necessary to run an **agile software development** pilot project. Pilot projects are commonly done for two reasons: To see *if* something will work or to learn *how* to make it work.

By now, enough other companies—very likely including some of your competitors—are using agile approaches like Scrum so that there is no longer any question of *if* it works.

The real question most organizations face is how to make agile or Scrum work for them. One or more software development pilot projects can be very helpful in providing those answers.

Should we convert all at once to agile methodologies?

Most organizations still begin with a handful of teams and spread their Scrum or agile software development process across the organization. But, *all-in* transitions are becoming more common.

One example of a company that successfully transitioned all at once is Mountain Goat Software client Salesforce.com. For them, it worked.

During their first year of using Scrum, Salesforce.com reported releasing 94 percent more features, delivering 38 percent more features per developer and providing over 500 percent more value to customers compared to the previous year.

How quickly can we become agile?

You'll see the benefits of adopting agile software development certainly within the first year and most likely within the first three to six months.

Keep in mind that many organizations go through a period of turmoil during the first one to three months as some people resist the change and others face legitimate hurdles in overcoming years of ingrained behavior.

Remember, though, that agile is not something you work at and then suddenly *become*. As you approach what you think of today as agile, you'll realize there's always further to go.

How do I get started with agile software development?

The approach we advocate is to *iterate toward agility*. Create an improvement backlog of all the things your organization could do better or could do in a more agile manner. Invite others to join this effort, particularly those with a passion for improving the development process or with agile experience either in your company or with a prior employer.

If your improvement backlog is long, consider grouping the items to create multiple improvement backlogs. Publicize the change effort to company employees and seek volunteers to work as a community toward making these improvements. As these *improvement communities* form, let each own one of the subdivided improvement backlogs.

If the transition effort warrants it, establish an *Enterprise Transition Community* that provides energy, support, resources, guidance and occasional direction to the improvement communities who are doing the real work of driving the agile transition across the company.

Recommended Resources

[Four Types of Resistors When Adopting Agile](#)

[Four Attributes of the Ideal Pilot Project](#)

[Choosing to Start Small or Go All In when Adopting Agile](#)

[Selecting a Development Process](#)

[Agile and Scrum for Video Game Development](#)

Related Courses

[Certified ScrumMaster®](#)

[Certified Scrum Product Owner®](#)

[Agile Estimating and Planning](#)

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

Learn About Agile

An Introduction to Agile and Scrum

The Scrum Framework

User Stories

Planning Poker

Agile Software Development

[Agile Project Management](#)

Scrum Foundations Video Series

All the foundational knowledge of Scrum including: the framework, values, different roles, meetings, backlogs, and improving efficiency & quality.

How are agile projects managed?

Within agile development, Scrum has the most to say about *exactly what is* agile project management. So let's use Scrum as our model for answering this question. On a Scrum project, there are three roles: product owner, ScrumMaster and team.

The product owner is responsible for the business aspects of the project, including ensuring the right product is being built and in the right order. A good product owner can balance competing priorities, is available to the team, and is empowered to make decisions about the product.

The Scrum Master serves as the team's coach, helping team members work together in the most effective manner possible. A good Scrum Master views the role as one of providing a service to the team, removing impediments to progress, facilitating meetings and discussions, and performing typical project management duties such as tracking progress and issues.

The team itself assumes agile project management roles when determining how to best achieve the product goals (as established by the product owner). Team members will collaboratively decide which person should work on which tasks, which technical practices are necessary to achieve stated quality goals, and so on.

So, what is “agile” about this process? Agile project management divides responsibility among more than one team member. In the case of Scrum, it’s a project’s product owner, ScrumMaster and the rest of the team.

Where does the Scrum Master fit in agile project manager roles and responsibilities?

In agile project management, the world may come to view the Scrum Master as a 21st century version of the project manager. But unlike a traditional project manager, the Scrum Master is not viewed as the person to credit (or blame) for the success (or failure) of the project.

The Scrum Master’s authority extends only to the process. The Scrum Master is an expert on the process, and on using it to get a team to perform to its highest level. But, a Scrum Master does not have many of the traditional responsibilities – scope, cost, personnel, risk management – that a project manager does.

Who handles conventional project manager duties in agile development?

Traditional project managers usually take on a great deal of responsibility. They are responsible for managing scope, cost, quality, personnel, communication, risk, procurement and more.

Agile project management often puts the traditional project manager in a difficult position. He or she is told, for example, to make scope/schedule tradeoff decisions knowing that a product manager or customer might second-guess those decisions if the project goes poorly.

Agile acknowledges this difficult position, and distributes the traditional project manager’s responsibilities. What is agile about this new paradigm is that many of these duties, such as task assignment and day-to-day project decisions revert back to the team where they rightfully belong.

Responsibility for scope and schedule tradeoff goes to the product owner. Quality management becomes a responsibility shared among the team, a product owner and Scrum Master. Other traditional tasks are distributed as well among a team’s agile project management roles.

Do agile projects scale with agile project management?

Agile processes like Scrum are definitely scalable. While the typical agile project has between five and 20 people across one to three teams, agile implementation has been successful on projects with 200 to 500 – even 1,000 people.

As you might expect, projects of that size must introduce more points of coordination and agile project management than small-scale implementation.

To coordinate the work of many teams, larger projects sometimes include a role called “project manager.” While involving someone on the project with this title or background can be very helpful, we need to be careful of the baggage associated with the project manager title.

Even on a very large agile project, the team will still do much of the project management. For example, teams decide how to allocate tasks, not a project manager; so the project manager role becomes more of a project coordinator.

Duties would include allocating and tracking the budget; communicating with outside stakeholders, contractors and others; maintaining the risk census with guidance from the teams, Scrum Masters and product owners; and so on. This is a true agile project management role.

Recommended Resources

[The Role of Leaders on a Self-Organizing Team](#)

[The Roles of the Project Management Office in Scrum](#)

[The Need for Agile Project Management](#)

[Leading a Self-Organizing Team](#)

[Agile and the Seven Sins of Project Management](#)

Related Courses

[Agile Estimating and Planning](#)

[Certified ScrumMaster®](#)

[Succeeding with Agile](#)

Learn About Agile

[New to Agile and Scrum?](#)

[Scrum](#)

[User Stories](#)

[Planning Poker](#)

[Agile Software Development](#)

[Agile Project Management](#)

[Agile Presentations](#)

[Mountain Goat on YouTube](#)

[Agile Mentors Podcast](#)

[Elements of Agile Assessment](#)

Agile & Scrum Training

[Online Certifications](#)

[Video Courses](#)

[In-Person Certifications](#)

[On-Site Courses](#)

[Training Schedule](#)

[Compare Courses](#)

[PDUs and SEUs](#)

More...

[About Us](#)

[Contact Us](#)

[Our Students](#)

[Blog](#)

[Books by Mike Cohn](#)

[Agile Tools](#)

[Succeeding with Agile](#)

FEATURED POSTS

[Introduction to Scrum PPT](#)

[Transitioning to Agile](#)

[Agile Distributed Teams - Scaling Agile](#)

[Prioritizing Your Product Backlog](#)

[Leading a Self-Organizing Team](#)

[Introduction to User Stories](#)

[Agile Estimating](#)

[Advanced Topics in Agile Planning](#)

[Take the Assessment](#)

MORE PRESENTATIONS POSTS:

[Let Go of Knowing: How Holding onto Views May Be Holding You Back](#)

You undoubtedly have a firmly held set of convictions about what is necessary to do agile well. ...

[Introduction to Scrum PPT](#)

You may have heard Scrum is one of the leading agile software development processes. With more than ...

[Agile Planning and Project Management](#)

In this session we will shatter the myth that agile teams can't plan. We'll start by looking at the ...

[GASPing Toward the Future: A Look at What's In Store for Scrum](#)

Scrum has never sat still. It evolves. One team tries something new, and it works so another team ...

Reported Benefits of Agile

This presentation summarizes a great deal of research on the benefits of becoming agile. Included ...

ADAPTing to Agile for Continued Success

You believe that agility is right for your team and for your business. You've implemented Scrum or ...

Agile Product Management

The vast majority of what has been written about agile processes is intended for programmers and ...

[Experiencing Agility From Requirements to Planning](#)

You know that before the development team writes their first line of code, they need a funded ...

[Transitioning to Agile](#)

You believe an agile software development method might be the right answer for your team or ...

[Succeeding with Agile](#)

Too many teams view planning as something to be avoided, and too many organizations view plans as ...

[Self-Organization & Subtle Control: Friends or Enemies?](#)

Agile leaders and managers have difficulty coming to grips with their role on self-organizing ...

[Selecting a Development Process](#)

With so many agile processes available, it can be hard to tell what is right for you. When the ...

[Agile Distributed Teams - Scaling Agile](#)

The early agile literature was adamant about two things: stick with small teams and put everyone in ...

[Project Economics](#)

Building the right product at the right time is just as important as building a quality product. ...

[Prioritizing Your Product Backlog](#)

The biggest risk to most projects is building the wrong product. Regardless of how fast your agile ...

[Planning for Contract Agile Projects](#)

Maybe you work for a vendor who must bid for work in response to an RFP. Or perhaps your company ...

[Leading a Self-Organizing Team](#)

One of the challenges of agile development is coming to grips with the role of leaders and managers ...

[Introduction to User Stories](#)

The technique of expressing requirements as user stories is one of the most broadly applicable ...

[Overcoming Waterfallacies & Agilephobias](#)

You've heard all the hype about agile software development processes, but you are left wondering, ...

[Incorporating Learning and Expected Cost of Change](#)

Product owners are typically asked to prioritize based on the nebulous term "business value." But ...

[Presentation for Product Owners: Storytelling with the Prioritized Product Backlog](#)

The vast majority of what has been written about agile processes is intended for programmers and ...

[Assessing Your Agility](#)

Are you curious how "agile" your organization is? Do you wonder how you compare with other ...

[Agile Estimating and Planning](#)

You've probably heard some people say, "Agile teams don't plan." Nothing could be further from the ...

[Agile Estimating](#)

Maybe you've heard about agile software development projects but aren't sure if they allow for the ...

[Agile and Scrum for Video Game Development](#)

An agile process like Scrum has been applied successfully in many industries. But does it work for ...

[Agile and the Seven Sins of Project Management](#)

Agile approaches to software development promise many advantages: shorter schedules, more ...

[Advanced Topics in Agile Planning](#)

When will you be done? Though this question doesn't go away on agile projects, it does get a little ...

[ADAPTING to Enterprise Agile](#)

You've likely heard of agile development processes like Scrum and XP. You might have even begun to ...

[ADAPTing to Agile](#)

You know that you want to become a more agile development organization, but you aren't sure how to ...

[Google / bayXP Presentation on YouTube](#)

[New Version of Redistributable Intro to Scrum Slides](#)

[Learn About Agile](#)

New to Agile and Scrum?

Scrum

[Agile & Scrum Training](#)

Online Certifications

Video Courses

[More...](#)

About Us

Contact Us

User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile Assessment		

Practical advice for making agile work in the real world

About Your Host

Brian Milner is Senior VP of Agile Training at Mountain Goat Software. Each episode features a guest co-host to discuss different aspects of agile as well as address your questions.

Latest Episodes

[#40: Is it Time to Go Out on Your Own? Tips and Insights with Chris](#)

Li

March 22, 2023 • 38 minutes

Chris Li joins Brian to talk about making informed decisions about going out on your own and navigating the challenges of entrepreneurship.

#39: The Art of Writing User Stories with Mike Cohn

March 15, 2023 • 29 minutes

Mike Cohn joins Brian to share his experience facilitating story-writing workshops and offers insights on creating effective user stories that deliver value to customers and focus to your team.

#38: Using Agile for Social and Societal Transformation with Kubair Shirazee

Shirazee

March 08, 2023 • 39 minutes

Kubair Shirazee, Founder of Peace through Prosperity, joins Brian to share his experiences using Agile tools for social and societal transformation, helping to empower marginalized communities and break the cycle of poverty.

#37: Servant Leadership, Not Spineless Leadership with Brad Swanson

Swanson

March 01, 2023 • 32 minutes

Brad Swanson joins Brian to explore the concept of servant leadership and share actionable takeaways to help you lead with compassion and empathy.

#36: Working with Humans with Dallas Jackson

February 22, 2023 • 39 minutes

Dallas Jackson joins Brian to explore the human aspect of work and the challenges that come with prioritizing the team while ensuring everyone is heard.

#35: Metrics with Lance Dacy

February 15, 2023 • 34 minutes

Join Lance Dacy and Brian Milner as they discuss the use of metrics in an Agile environment to ensure optimal performance without taking things in the wrong direction.

#34: I'm Trained, Now What? with Julie Chickering

February 08, 2023 • 33 minutes

Join Julie Chickering and Brian Milner as they provide exclusive insight on utilizing your Scrum training, expanding your expertise, and passing your knowledge on to others.

#33 Mob Programming with Woody Zuill

February 01, 2023 • 40 minutes

Join Woody Zuill and Brian Milner as they discuss the benefits of teams working together through Mob Programming.

#32: Scrum in High School Sports with Cort Sharp

January 25, 2023 • 34 minutes

Join Cort Sharp and Brian Milner as they discuss experimenting with Scrum in other out-of-the-box environments, including how Cort uses it to train the high school swim team he coaches.

#31: Starting Strong: Tips for Successfully Starting with a New

Organization with Julie Chickering

January 18, 2023 • 35 minutes

Join Julie Chickering and Brian Milner as they discuss strategies you can use to get started on the right foot with your new organization.

#30: How to Get the Best Out of the New Year with Lance Dacy

January 11, 2023 • 29 minutes

Join Lance Dacy and Brian Milner as they discuss how to get the best out of the new year.

#29: Influencing Up with Scott Dunn

December 21, 2022 • 30 minutes

Join Scott Dunn and Brian Milner as they discuss how to influence up, including the tools you can use to overcome difficulties and step into a partnership with the influential people in your organization for influence that creates lasting change.

#28: The Most Valuable Books for Leadership, Learning, and Sharing with Julie Chickering

December 14, 2022 • 34 minutes

Julie Chickering sits down with Brian to share the best gift books for the Scrum masters in your life.

#27: Leading Without Blame with Tricia Broderick

December 07, 2022 • 32 minutes

Tricia Broderick joins Brian to discuss how to lead without blame.

#26: How Getting to Small Helps Teams Get Things Done with Lance Dacy

November 30, 2022 • 35 minutes

Lance Dacy joins Brian to discuss breaking down stories to get things done.

#25: Scaling with Henrik Kniberg

November 16, 2022 • 37 minutes

Henrik Kniberg joins Brian to talk about creating the Spotify Model.

#24: How Agile Organizations Respond to Challenging Economic Times with Scott Dunn

November 09, 2022 • 34 minutes

Scott Dunn joins Brian to talk about how Agile teams and organizations respond in difficult economic times.

#23 How Agile Works in Education with John Miller

November 02, 2022 • 36 minutes

John Miller joins Brian to talk about Agile in the classroom.

#22: How to Create Helpful Product Roadmaps with Roman Pichler

October 26, 2022 • 44 minutes

Roman Pichler joins Brian to talk about Product Roadmaps.

#21: Agile Marketing Teams with Stacey Ackerman

October 19, 2022 • 31 minutes

Brian speaks with Stacey Ackerman about working with Marketing teams using Agile.

#20: Best of Coaching Calls with Mike Cohn

October 05, 2022 • 31 minutes

Mike and Brian take audience questions in the “best of” from the Agile Mentors Community’s monthly coaching calls.

#19: How does project management work in Agile? with Julie Chickering

September 28, 2022 • 38 minutes

This week, Brian Milner is joined by Julie Chickering to talk about the wild world of Project Management.

#18: Coaching in an Agile world with Lyssa Adkins

September 21, 2022 • 34 minutes

Lyssa Adkins joins Brian to talk about the wonderful world of Agile coaching.

#17: Getting There From Here: Agile Transformations with David Hawks

September 14, 2022 • 40 minutes

David Hawks joins Brian to discuss the process of an organization becoming Agile.

#16: Quality: The Hidden Secret Ingredient with Mitch Lacey

September 07, 2022 • 31 minutes

Mitch Lacey joins Brian to talk about building quality into our work.

OLDER POSTS

Learn About Agile

New to Agile and Scrum?

Agile & Scrum Training

Online Certifications

More...

About Us

Scrum

Video Courses

Contact Us

User Stories

In-Person Certifications

Our Students

Planning Poker

On-Site Courses

Blog

Agile Software

Training Schedule

Books by Mike Cohn

Development

Compare Courses

Agile Tools

Agile Project Management

PDUs and SEUs

Agile Presentations

Mountain Goat on YouTube

Agile Mentors Podcast

Elements of Agile

Assessment

Get Your Customized Elements of AgileSM Assessment

Find out how your team is progressing in their mastery of the 20 key Elements of Agile.

Find out how your team is progressing in their mastery ~~Elements of Agile~~.

What Are the Elements of Agile?

Through two decades of leading agile transitions, we've identified twenty key Elements for success with agile. These Elements span six areas that are essential for teams undergoing an agile transformation, from transition management to agile team structure to culture. By focusing and continually improving in each of these areas you can spot and overcome impediments to agility.

Transition	Culture	Collaboration	Teamwork	Team Structure	Delivery
Management	Do your teams have an agile and you have advocates to champion change.	See how to get people to work together Find out what they need to do to cultivate one.	Not just about working together, teamwork is about being accountable and input. Is it working at your work?	Ensure teams are built in a way that will help them and striving for improvement.	Are your teams focused on the most valuable tasks? Discover what you need to do to get rapid feedback and deliver functionality on every iteration.
Make sure everyone understands agile and mindset?					
you have advocates to champion change.					

How the Elements of Agile Assessment Works

In the last 20+ years, we've helped companies take their agile transitions from good to great. Based on our experience with organizations in every phase of agility, we've identified a specific framework that will help you succeed with fewer struggles: the Elements of Agile.

To discover how your organization is doing in each of these Elements, we've created a free agile assessment tool. By taking the survey, you can discover how your team or organization is progressing towards agility. You'll get a free report that explains where you're doing well and where you might want to make some improvements.

The agile assessment questions are designed to help uncover places where your team might be showing signs of struggle or incomplete understanding. They will also spotlight places where your organization is likely demonstrating agility and solid growth.

To be clear, this isn't an agility test or an agile maturity model; instead it is a tool to help you measure progress as you and your teams work continually toward being more adaptive and responsive.

What's in Your Customized Report?

After answering the questions, you'll receive your own, customized agile team assessment report immediately. We designed the Elements of Agile report to show you where you are today, and the specific things you need to do to be a truly agile organization long term.

The report isn't just a measure of progress. It's filled with practical ideas you can take away and implement today. Each Element is defined and explained, so that you know both what to do, and why.

Every Element includes symptoms to watch for, for example, signs that your team might not be as cross functional as you hope, or that you aren't dealing with fully empowered teams. Your report might also suggest some things to try to help your teams improve at key agile practices, such as working with user stories, getting more out of your sprint reviews, or ways to tweak your organizational structure or culture to

help development teams deliver.

How to Get Your Customized Report

Growing more agile takes time and patience. Getting started with the Elements of Agile Assessment, however, is free and easy.

1. Answer the assessment questions
2. Receive your customized report
3. Put your plan into action

Next Steps

After receiving your customized report, we recommend you share it with your team to get their reaction. Ask them questions to gauge the accuracy of how you answered the questions:

- Do they agree with the areas that were flagged for improvement?
- Have they noticed progress in the areas that are flagged as good / growing?
- Where would they like to focus first? Could they commit to trying one or two things in the next sprint?

Then, work with them (and with leadership as necessary) to create an improvement plan that you can put into action. You can even come back in several months and take the assessment again.

No matter where you are on your agile journey or which agile methodology you are using, there is always room for improvement. As the [agile manifesto](#) explains one of the 12 principles of agile project management: "At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly." This tool will give you new ways to tune and adjust what you and your teams are doing.

If you'd like help with guiding teams through these conversations, understanding key elements of agility, or with implementing certain changes, you can reach out to us at Mountain Goat Software. We offer video training, public and private agile training, and agile consulting services. Our decades of experience can help you overcome the obstacles that might be standing in the way of your success with agile.

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	

Get Your Customized Elements of Agile Assessment

Mountain Goat on YouTube

Agile Mentors Podcast

Elements of Agile
Assessment

Live Online Training for Individuals

Live, online training courses provide flexible training options you can access from anywhere.

Join the more than 2,000 people who've enjoyed live, online certified courses with Mountain Goat in the last year.

Click on a specific course to view the training schedule, or [view our full training schedule here](#).

Certified	Certified	Advanced	Advanced
ScrumMaster®	Scrum	Certified	Certified
	Product	ScrumMaster®	Scrum
	Owner®		Product
			Owner®

Our Certified Scrum Master training is a two-day in-person course taught by one of our Certified Scrum Trainers. This Scrum training for certification not only teaches the fundamental principles of Scrum, but also gives participants hands-on experience. If you're a Certified ScrumMaster® with 12 months of experience, take this next stage of training and deepen your facilitation and coaching skills as a ScrumMaster with this advanced certification. Stand out from other Product Owners with this advanced certification. Lifetime access to course materials and the only A-CSPO that uses our interactive software for breakout exercises. 12 months group coaching calls included.

project
success.

after the
course.

Need help deciding which course is best for you?

Want to make sure you're choosing the best course for you?

We've outlined different learning paths for your role, browse our recommendations using the link below:

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile Assessment		

On-Demand Video Training for Individuals

On-demand video courses put you in control of what you want to learn and when. Each course provides immediate access to all materials so you can work through them at your own pace. And with lifetime access to all videos, you can refresh your memory and be confident you'll always have lessons to hand when you need them most.

Estimating With Story Points	Better User Stories	Agile Estimating and Planning	Scrum Repair Guide	Scrum Foundations	Let Go of Knowing
This course is ideal for anyone who wants to gain a solid understanding and how to use them correctly. It's perfect for introducing story points to a team, (and stakeholders)	Overcome the challenge of writing user stories to join the ranks of high-performing agile teams, deliver the right products to market, and delight your customers.	This agile training shows you how to go about agile estimating and planning, planning, and why it's crucial even in a fluid and iterative process.	ScrumMasters and their teams will learn how to overcome the most common and vexing Scrum challenges with this online troubleshooting course.	The perfect introduction to Scrum. You'll get an overview of the key roles, ceremonies, and artifacts, and see how they work together.	Watch Mike Cohn's popular conference keynote session and learn how holding onto views may be holding you back.

fixing
teams
that
have
developed
bad
habits.

deciding which course is best for you?

You're choosing the best course for you?

nt learning paths for your role, browse our recommendations using the link below:

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile Assessment		

Certified Scrum Training for Individuals

Certified Scrum Training can help your professional development, and with classes offered live, online, or in-person, you can get certified in the way that works best for you. These courses also count for continuing education units (SEUs) from the Scrum Alliance, as well as PDUs within the PMI.

Click on a specific course to view the training schedule, or [view our full training schedule here](#).

More than 24,000 people have chosen to train with Mountain Goat Software! Why not join them?

Certified	Certified	Advanced	Advanced
ScrumMaster®	Scrum	Certified	Certified
	Product	ScrumMaster®	Scrum
	Owner®		Product
			Owner®
Our Certified Scrum Master training is a two-day in-person course taught by one of our Certified Scrum Trainers. This Scrum training for certification not only teaches the fundamental principles of Scrum, but also gives participants hands-on experience.	Our Certified Scrum training is a two-day in-person course taught by one of our Certified Scrum Trainers. This Scrum training for certification not only teaches the fundamental principles of Scrum, but also gives participants hands-on experience.	If you're a Certified ScrumMaster® with 12 months of experience, take this next stage of training and deepen your facilitation and coaching skills as a ScrumMaster with this advanced certification.	Stand out from other Product Owners with this advanced certification. Lifetime access to course materials and the only A-CSPO that uses our interactive software for breakout exercises. 12 months group coaching

backlog as
a tool for
project
success.

calls
included
after the
course.

We love hearing compliments like this from companies who are recruiting for agile professionals. Our training does more than teach you the theory of Scrum and agile, you also get the practical, real-world advice that makes you a valuable addition to any scrum team.

Deciding which course is best for you?

When you're choosing the best course for you?

To help you find the right learning paths for your role, browse our recommendations using the link below:

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile Assessment		

Rather than sending only a few representatives to a public course, consider training your entire team with our on-site training. All of our public courses can be brought to your site, minimizing both scheduling challenges and also the expense and hassle of out of town travel. We can certify team members in a Scrum Alliance-certified course such as Certified ScrumMaster (CSM) or Certified Scrum Product Owner (CSPO). We can also teach any of our other agile training and Scrum training courses, or we can offer both a certified course and another of our agile and Scrum courses through onsite training.

[Schedule an On-site Course](#)

Onsite Agile Coaching

Scrum teaches us to inspect and adapt. It's a good idea, therefore, sometimes to plan for periodic Agile coaching check-ups. Combining a training class with follow-on coaching a month or two later is a wonderful way to ensure that the key concepts presented in the Agile training or Scrum training class have become daily habits on your teams. Whatever your needs, we can devise an onsite training and Agile coaching plan that will help you succeed.

Some Popular Combinations

One-day Agile Estimating and Planning class followed by a day of assisted estimating and planning with your backlog.

Two-day Certified ScrumMaster course with a day or two of coach-assisted backlog generation and sprint planning.

Certified Scrum Product Owner can be combined with coaching that emphasises user stories or release planning.

Sample Coaching Engagements

Conduct a user story-writing workshop

Work with you to create a reliable release plan

Facilitate a sprint planning meeting, helping coach team members to plan a sprint at the right level of detail

Facilitate end of iteration reviews and retrospectives

Attend daily scrum meetings and provide advice on how to get the most value from them

Help initiate an agile transition effort using the iterative ADAPT approach outlined in Mike Cohn's Succeeding with Agile book.

Meet with and educate executives and key stakeholders about agile and their role in its successful adoption.

Help convince reluctant agile team members to give agile a try.

Specialized Consultants in Scrum & Agile

Mountain Goat Software consultants specialize in Scrum and agile practices. Mike Cohn co-founded the Scrum Alliance, which is a non-profit organization whose mission is to support Scrum and Scrum practitioners worldwide. Mike ran his first Scrum project in 1995, making him among the most experienced of Scrum trainers and coaches. In fact, in 2003 he co-taught the first-ever Certified ScrumMaster course. Our well-respected and popular Agile training courses, especially when used as onsite training and combined with team Agile coaching or mentoring, will help put you on the path to success.

"Mike's classes at Yahoo! have been incredibly useful. I recommend him to anyone who is serious about implementing Agile in their organization."
– Gabrielle Benefield, Director of Agile Development, Yahoo!

Let us ensure your team and organization get off on the right start, move to the next level, or reenergize a stalled transition with our onsite Agile Training.

Schedule an On-Site Course

Please fill out the following information and we will be in touch with you about scheduling an on-site course.

Course Information

Course Type	Certified ScrumMaster® Certified Scrum Product Owner® Unsure; Let's Talk
--------------------	--

Number of Participants

Location

Date

Contact Information

First Name

Last Name

Email Address

Company Name

Phone Number

Additional Information

Comments

About Mike Cohn

Mike is CEO of Mountain Goat Software, and one of the industry's most well-respected Certified Scrum Trainers. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile*. He is a co-founder and former board member of the Scrum Alliance, and a co-founder of the non-profit Agile Alliance, home of the Agile Manifesto.

With 20+ years of agile training experience, Mike has honed a talent for explaining agile concepts with clear illustrations and real-life examples. Participants enjoy his passion for teaching the agile methodologies in a relatable and digestible way.

Scrum Certifications include: CSM, CST, CSPO, A-CSM, CSP-SM, CSP-PO

[Read More About Mike](#)

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

	Certified ScrumMaster®	Certified Scrum Product Owner®
Course Type	Live online	Live online
Duration	2 days	2 days
Certification	Certified ScrumMaster (CSM) awarded by the Scrum Alliance	Certified Scrum Product Owner (CSPO) awarded by the Scrum Alliance
Exam Required	Online multiple-choice exam required. Passing score is 74%. Test fee is included in your course registration. You can re-take the exam (if needed) at no charge one time. The Scrum Alliance charges \$25 per attempt after that.	No exam required.
Scrum Overview	Agile Scrum	Scrum
Product Backlog	Responsibilities & Attributes User & Job Stories Two Ways of Adding Detail Progressive Refinement Organizing the Backlog Story Splitting	What Is a Product Backlog? Progressive Elaboration User Stories Adding Detail To User Stories Job Stories Technical Stories Themes & Epics Getting it Right-Sized Who Contributes Items? Story Writing Workshop Story Mapping
The Product Owner	What is a Product Owner? Rights & Responsibilities Managing Stakeholders	What Is a Product Owner? Characteristics Responsibilities Product Owners in Different Contexts Quarterly Activities Involvement Over Time Asking For Clarifications, Not Changes Dealing With Difficult Situations Business Analysts & Product Managers Scaling the Product Owner Role

The Scrum Master	Rights & Responsibilities Four Facilitation Techniques Coaching, Mentoring, Teaching & Facilitating The Scrum Master in Dual Roles Servant Leadership	Responsibilities Description of Role
The Development Team	Rights & Responsibilities Self-Organizing Cross-functional Collaborating Component and Feature Teams	Responsibilities Component and Feature Teams
Managers	Project Managers Functional Managers	
Customers and Users		Validating Assumptions User Roles User Role Modeling Personas Roles vs. Personas Decorated Roles & Extreme Characters
Visioning		Specifying The Problem, Not The Solution Creating a Concise Vision Five Techniques for Communicating Vision
Prioritizing		Key Concepts in Prioritization Factors in Prioritization Formal Approaches What Different Stakeholders Value Collaborative Prioritization Techniques
Estimating	Relative Estimating Story Points Ideal Time Affinity Estimation 3 Ways to Estimate Velocity	
Planning	Fixed-Date Plans Fixed-Scope Plans Creating the Plan Burndown Charts Technical Debt	Accuracy and Precision Velocity Using a Velocity Range Fixed-Date Plans Fixed-Scope Plans
Sprinting	Time Boxes No Changes Allowed Reciprocal Commitments Defining Done Sprint Types Potentially Shippable Product Increments Introducing Change into a Sprint	
Sprint Planning	Meeting Responsibilities The Sprint Goal The Purpose Capacity-Driven Sprint Planning Velocity-Driven Sprint Planning	

Decomposing Stories into Tasks
Working Out of Order
What To Do When the Product Owner is Missing

Other Scrum Meetings	Daily Scrum Common Dysfunctions Sprint Review Responsibilities When to Demo Sprint Retrospective Retro Technique Product Backlog Refinement	Story Writing Workshop
Scaling Scrum (video only)	Scaling Guidelines Scaling the Planning Meeting Scrum of Scrums Communities of Practice	

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile Assessment		

[Scrum Education Units \(SEUs\) And Professional Development Units \(PDUs\)](#)

Scrum Education Units (SEUs) and Professional Development Units (PDUs)

If you are a PMP®, PgMP®, PMI-SP®, PMI-RMP® or PMI-ACP, you can earn **professional development units (PDUs)** through our educational services. You can also earn **Scrum education units (SEUs)** from the Scrum Alliance.

Check out the following agile training or Scrum certification in person from author, educator and Certified Scrum Trainer Mike Cohn at any one of his workshops worldwide or online.

In-Person PDUs and SEUs

Choose from a variety of courses year-round on agile estimation and planning, succeeding with agile, user stories for agile software development, the certified Scrum product owner and ScrumMaster certification. Two-day courses are 15 PDUs and 16 SEUs and one-day classes are 7.5 PDUs and 8 SEUs.

Agile and Scrum Courses

- [Certified ScrumMaster®](#)
- [Certified Scrum Product Owner®](#)
- [Succeeding with Agile](#)

Live Online PDUs and SEUs

Learn Scrum from Mountain Goat Software no matter where in the world you live. Our online agile training gives you a front row seat in a virtual classroom led by certified ScrumMaster, Scrum trainer and author, Mike Cohn.

Live Online Courses

- [Certified ScrumMaster®](#)
- [Certified Scrum Product Owner®](#)
- [Advanced Certified ScrumMaster®](#)
- [Advanced Certified Scrum Product Owner®](#)

Online PDUs and SEUs

Learn Scrum from Mountain Goat Software no matter where in the world you live. Our online agile training gives you a front row seat in a virtual classroom led by certified ScrumMaster, Scrum trainer and author, Mike Cohn.

On-Demand Video Courses

- [Estimating With Story Points](#)
- [Better User Stories](#)
- [Agile Estimating and Planning](#)
- [Scrum Repair Guide](#)
- [Scrum Foundations](#)
- [Let Go of Knowing](#)

Conference Videos for PDUs and SEUs

Check out footage of educational seminars led by Mike Cohn at conferences throughout the year. Viewing videos from our library of conference events earns you professional development units, too. PDUs vary for this type of continuing education.

Available Videos

- [Leading a Self-Organizing Team](#)
- [Getting Agile with Scrum](#)
- [Scaling Agile with Distributed Teams](#)
- [Agile Estimating](#)
- [User Stories](#)
- [Advanced Topics in Agile Planning](#)
- [Agile and the Seven Deadly Sins of Project Managing](#)

[Join Agile Mentors »](#)

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

Hi! I'm Mike Cohn

For over twenty years I've been building high-performing software development teams and organizations through the use of agile and Scrum. I've worked with startups and some of the largest organizations in the world.

My in-person training, coaching, and online video courses work together to give you everything you need to deliver the right product on time every time.

Direct Access to My Best Content

To find answers to specific questions about Scrum and agile, search or browse my blog. I have been blogging since 2005 and now have one of the most active and popular blogs about agile or Scrum. I add about one new post a week to the hundreds of posts already in existence. To make sure you don't miss my newest posts, you can subscribe to my blog by email. It's quick, easy, and you can unsubscribe at any time.

If you want even more agile guidance, sign up for my "Tips to Succeed with Agile" emails. I send one tip via email every Thursday to subscribers. These exclusive tips are not posted publicly or shared anywhere else — the only way to get them is to subscribe. I consider it an honor to be invited into your inbox, so these tips represent my most recent thoughts and best advice on Scrum and agile. We never sell your contact information to anyone. And you can unsubscribe any time.

"Mike's wealth of experience and exuberant style of training leads

*to courses that always deliver a heavy-weight punch. There is
always useful material to apply to your project."*

Ian Wermeling

[Hear from others we've trained »](#)

We Wrote the Book(s) on Agile

If you need information on agile or Scrum, you can find it here. For a thorough introduction to Scrum, agile planning, and user stories, start with one of my three books: [Succeeding with Agile: Software Development Using Scrum](#), [Agile Estimating and Planning](#), and [User Stories Applied for Agile Software Development](#).

Trusted by Companies Worldwide

[See a partial list of clients »](#)

My Background

I began my career as a programmer working in C and C++. I wrote some of the first books available on Java. From there, I moved onto a variety of executive roles in technology at companies, from startups to the Fortune 40.

I ran my first Scrum project in 1995, and have been a vocal proponent of Scrum ever since. I am a co-founder of both the Agile

Alliance and Scrum Alliance and served for many years on the boards of directors of each. I am a frequent keynote speaker at industry and professional conferences.

In addition to my writing, in-person training, and coaching, I have created several [online video training courses](#).

Interviews and Coverage

As agile increased in popularity, the Wall Street Journal asked Mike to explain the purpose and practise of the daily stand-up.

[Read the full article »](#)

In this interview with US National Public Radio, Mike shares how the daily stand-up is a remedy for 'meeting misery'.

[Read the full article »](#)

"The point is to come together very focused, just a short period, and let people get right back to work. No delays."

[Read the full article »](#)

In the Economist Intelligence Unit report: *Reimagining Agility: Change, empowerment and the organisation of tomorrow*, Mike shares his experience of working remotely with clients.

[Read the full article »](#)

[...also seen in:](#)

Agile Events Featuring Mike Cohn

See past events that Mike Cohn has spoken at, including agile conferences, user group meetings, and more.

[View Conferences & Events »](#)

Learn About Agile

New to Agile and Scrum?

Agile & Scrum Training

Online Certifications

More...

About Us

Scrum

Video Courses

Contact Us

User Stories

In-Person Certifications

Our Students

Planning Poker

On-Site Courses

Blog

Agile Software Development

Training Schedule

Books by Mike Cohn

Agile Project Management

Compare Courses

Agile Tools

Agile Presentations

PDUs and SEUs

Mountain Goat on YouTube

Agile Mentors Podcast

Elements of Agile

Assessment

Mountain Goat Software provides a variety of services that can help any organization adopt and improve their use of agile processes and techniques in order to build extremely high performance development organizations. Whether you are interested in [coaching](#), [consulting](#) or [training](#), we can help.

Please contact us to discuss how we can help you Succeed With Agile.

Certified Scrum Trainer, [Mike Cohn](#)

Let's discuss how we can help you Succeed With Agile.

Mike enjoys hearing from his audience but is unable to personally answer every question. If you'd like to ask a specific question about Agile, then please ask it [here](#).

[GENERAL INQUIRIES](#)

[BOOK AN ON-SITE CLASS](#)

First Name

Last Name

Email Address

Company

Phone Number

Comments

To get you a faster reply, your email may first be routed to a member of my support team.

[Take the Assessment](#)

Connect with Mike

You can contact Mike via e-mail or follow him on [Facebook](#), [Twitter](#), [LinkedIn](#)

Mountain Goat Software

2600A East Seltice Way #377
Post Falls, ID 83854
phone: 1-888-61-AGILE (24453)
hello@mountaingoatsoftware.com

Learn About Agile **Agile & Scrum Training** **More...**

New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile Assessment		

[Books](#)

Books

Agile books are a great way to advance your agile development and provide you with reference material. Here at Mountain Goat Software we are pleased to offer the following books from leading agile authority Mike Cohn. With three books on agile development topics, [Mike Cohn](#) is frequently sought after as a speaker, instructor, and agile coach. Mike's in-depth knowledge comes from 25 years of experience as a technology expert in companies of various sizes. He has also written articles for Better Software, IEEE Computer, Cutter IT Journal, Software Test and Quality Engineering, Agile Times, and the C/C++ Users Journal. He has a long-standing commitment to the agile and Scrum communities and was a founding member of both the Agile Alliance and the Scrum Alliance. These agile books are just the thing for anyone wanting to know more about agile development. Click on the links below to discover more about these agile software development books.

An eBook version including EPUB, MOBI (Kindle), and PDF versions can be purchased at [InformIT](#).

Books by Mike Cohn:

Succeeding with Agile: Software Development Using Scrum

[More Info](#)[Get free chapters](#)

Agile Estimating and Planning

[More Info](#)[Get free](#)

User Stories Applied

[More Info](#)

Get free
chapters

Highly Recommended by Mike

Implementing	Management	Essential	Coaching	Agile	Agile
Agility:	3.0:	Serum:	Agile	Testing:	Product
Retrospectives:	Leading	A	Teams:	A	Management
Helping's	Agile	Practical	A	Practical	with
Teach Inside	Developers,	Guide	Companion	Guide	Scrum:
Bed on Agile	Developing	to the	for	for	Creating
Minimize	Agile	Most	Scrum	Testers	Products
Efficient	Leaders	Popular	Masters,	and	that
by Maly	by	Agile	Agile	Agile	Customers
Agile	Jurgen	Development	Coaches	Teams	Love
DeGroot	Appelo	Process	and	by	Agile
Garske		by	Project	Lisa	Game
Scrum		Kenny	Managers	Crispin,	Roman
Useless		Rubin	in	Janet	Development
Agile			Transition	Gregory	with
Habits			by		Scrum
Playwith			Lyssa		by
Adventures			Adkins		Clinton
Replay					Keith
Software					
Zwana					
Advice					
for Larhe					
by by Šochová					
Clifton Alexander	Yoti & While				
Keilharf	First Team				
Bas	Year by				
Vodde	and Lisa				
Beyond	Crispin,				
by	Janet				
Mitch	Mitch Gregory				
Lacey					

Learn About Agile Agile & Scrum Training More...

New to Agile and Scrum? Online Certifications About Us

Scrum Video Courses Contact Us

User Stories In-Person Certifications Our Students

Planning Poker On-Site Courses Blog

Agile Software Training Schedule Books by Mike Cohn

Development Compare Courses Agile Tools

Agile Project Management PDUs and SEUs

Agile Presentations

Mountain Goat on YouTube

Agile Mentors Podcast

Elements of Agile
Assessment

[Tools](#)

Tools

We are providing this collection of tools for your use on agile projects. These tools are based on [ideas described in Mike Cohn's books](#). You can read more information about agile project management training expert and Certified Scrum Trainer, [Mike Cohn](#)

If you find that these free agile tools are not enough and you would like more hands-on experience with Scrum and agile projects, please check out the [agile and Scrum training courses we offer](#).

Velocity Range Calculator

The Mountain Goat Software velocity range calculator is used to predict how much work a team will complete during a planned number of upcoming iterations using a range rather than a specific value.

Theme Screening

Theme screening is a useful and easy prioritization technique that can be used to prioritize themes and/or epics against one another. It can also be used to prioritize entire projects or products.

Relative Weighting

Relative weighting is a prioritization approach that considers both the positive benefit of a feature and the negative impact of its absence to prioritize user stories, features, or even projects.

Theme Scoring

Theme scoring is a prioritization technique that can be used to prioritize themes (groups of user stories) and epics (large user stories) against one another. It can also be used to prioritize projects or products against one another.

Project Success Sliders

Project Success Sliders are a way for key stakeholders or a product owner to convey their expectations to a Scrum or agile team. There are six sliders, each of which reflects some dimension by

which agile project success can be determined.

Planning Poker

Planning Poker® is a collaborative approach used by agile and Scrum teams to estimate their product backlogs. When played in person, these values are printed on cards and held by each estimator, giving Planning Poker its name.

Learn About Agile

New to Agile and Scrum?

Agile & Scrum Training

Online Certifications

More...

About Us

Scrum

Video Courses

Contact Us

User Stories

In-Person Certifications

Our Students

Planning Poker

On-Site Courses

Blog

Agile Software Development

Training Schedule

Books by Mike Cohn

Agile Project Management

Compare Courses

Agile Tools

Agile Presentations

PDUs and SEUs

Mountain Goat on YouTube

Agile Mentors Podcast

Elements of Agile

Assessment

Terms and Conditions of Service

Course Licensing

You may be able to purchase a license to stream certain course videos through the website (a "Streaming License"). For each Streaming License you purchase, you (and only you) may view the licensed course video through the streaming functions on the website. You may not sell or otherwise transfer this Streaming License to any other person, nor may you utilize any screen grabbing or similar technologies to make any copies of any portions of the course videos. Streaming Licenses are valid so long as the applicable course video is available through the website.

Mountain Goat Software either owns or licenses all rights, title and interests in and to the copyrighted material used in connection with the course videos. Under no circumstances will you have any rights of any kind in or to the course videos, other than the right to use those videos in accordance with the Mountain Goat Software Website Terms of Service.

Terms of Service

Welcome to the Mountain Goat Software, LLC ("MGS") web site located at www.mountaingoatsoftware.com (the "Site"). MGS makes the Site and the related services on the Site available to you subject to the following terms and conditions in this Agreement ("Terms"). By accessing, viewing, or using the content, material, or services available on or through this Site, you indicate that you have read and understand these Terms, and that you agree to them and intend to be legally bound by them. If you do not agree to these Terms, you are not granted permission to use this Site and must exit immediately.

For the purposes of this Agreement the following definitions apply:

"You" means you, the person using the Site.

"MGS," "we" or "us" means Mountain Goat Software, LLC.

1. Use of Site. Subject to these Terms, we grant you the personal, limited, revocable, non-exclusive and non-transferable right to use this Site. You may download one temporary copy of the Site on any single computer, and you may print one copy of these materials, for your own personal use only.

2. Proprietary Rights. As between you and MGS, MGS owns or licenses all data, content, graphics, forms, artwork, images, photographs, functional components and any software concepts and documentation and other material on, in or made available through the Site ("Site Materials"), as well as the selection, coordination, arrangement, and organization and enhancement of the Site Materials. All Site Materials are protected pursuant to copyright, trademark, patent, and other applicable laws. You agree not to remove or alter any copyright notice or any other proprietary notice on any Site Materials. As between any user and MGS, all names, trademarks, service marks, certification marks, symbols, slogans or logos appearing on the

Site are proprietary to MGS or its affiliates, licensors, or suppliers. Use or misuse of these trademarks is expressly prohibited and may violate federal and state trademark law. Under no circumstances will you have any rights of any kind in or to the Site Materials, other than the right to use the Site Materials in accordance with these Terms.

3. Unauthorized Activities. You agree that you will not use the Site or the Site Materials for (a) any illegal or unauthorized purposes that violate any local, national, or international laws (including but not limited to import, export, copyright, and trademark laws); (b) modifying, copying, distributing, displaying, performing, reproducing, publishing, licensing, creating derivative works from, transferring, selling any of the Site Materials, unless otherwise authorized by these Terms or in a separate written agreement with MGS; (c) attempting to gain unauthorized access to MGS's computer system or engaging in any activity that interferes with the performance of, or impairs the functionality of the Site or any services provided through the Site; (d) any resale or commercial or institutional purposes, including use of the Site or Site Materials to further the purposes of any library, corporation, governmental entity, or any other institution or entity; (e) any downloading or copying of the Site Materials for any reason other than your personal use, or any use of data mining, robots or similar data gathering and extraction tools; (f) using the Site to access or collect any personally identifiable information, including any names, email addresses or other such information for any purpose, including, without limitation, commercial purposes; or (g) removing, circumventing, disabling, damaging or otherwise interfering in any way with any security-related features of the Site aimed at preventing or restricting the unauthorized use of the Site or any of the Site Materials. Only you are permitted to access the Site and the Site Materials through your account. MGS may monitor whether your account is being accessed concurrently through multiple devices. Without limiting anything else herein, if MGS discovers concurrent access to the Site or the Site Materials through your account, MGS may, in its sole discretion, suspend or terminate your account and/or your ability to use the Site. You may use the Site and the Site Materials solely for your personal use in a manner consistent with these Terms, and such rights may not be transferred or assigned to any other person or entity. Any other use of the Site or Site Materials, including but not limited to the aforementioned unauthorized uses, without prior written permission of MGS is strictly prohibited. You acknowledge and agree that the unauthorized use of the Site or the Site Materials could cause irreparable harm to MGS and that in the event of such unauthorized use, MGS shall be entitled to an injunction in addition to any other remedies available at law or in equity.

4. Materials Submitted to the Site. By transmitting or submitting to MGS any suggestions, information, posting to blogs, or message boards, or any other materials to the Site ("User Content"), you represent that you have the full legal right to provide the User Content and that use of the User Content by the Site and all other persons and entities will not (a) infringe any intellectual property rights of any person or entity or any rights of publicity, personality, or privacy of any person or entity, including, but not limited to, as a result of your failure to obtain consent to post personally identifying or otherwise private information about a person; (b) violate any law, statute, ordinance, or regulation; (c) be defamatory, libelous or trade libelous, unlawfully threatening, or unlawfully harassing; (d) be obscene, child pornographic, or indecent; (e) violate any community or Internet standard; (f) contain any viruses, Trojan horses, worms, time bombs, cancelbots, or other computer programming routines that damage, detrimentally interfere with, surreptitiously intercept, or expropriate any system, data or personal information, or that facilitate or enable such or that are intended to do any of the foregoing; (g) constitute misappropriation of any trade secret or know-how; or (h) constitute disclosure of any confidential information owned by any third party.

Upon your submission of User Content or other material or information to MGS, you grant MGS a worldwide, perpetual, non-terminable, irrevocable, transferable, license to access, use, distribute, perform, reproduce, display, modify, create derivative works based upon, and sublicense, and to permit others to access, use, distribute, perform, reproduce, display, modify and create derivative works based upon the User Content, all

without any compensation to you whatsoever. MGS shall not be responsible for changes or modifications of any User Content that you submit to MGS. If you believe that any content or postings on the Site violate your intellectual property or other rights, please follow our Complaint Procedure in Section 14 of these Terms. MGS, its subsidiaries, and affiliates are not responsible for and do not guarantee the accuracy or completeness of any User Content that is submitted to MGS.

5. Other Sites. The Site is available for informational purposes only. The Site may contain links to other Internet Web sites for the convenience of users in locating information, products, or services that may be of interest. Use of such third party links, the Site and the Site Materials and any other material or content on and made available through the Site is entirely at your own risk. MGS does not recommend and expressly disclaims any responsibility for the content, the accuracy of the information, or quality of products or services provided by or advertised on third party sites or the transactions you conduct or enter into with third parties. Your use of any third party's website is at your own risk, and subject to the terms and conditions of such other websites. MGS does not endorse any product, service, or treatment provided on a third party website or advertised or provided on the Site.

6. Payment. There are products or services on the Site (such as the Planning Poker®† cards) which are available for purchase. By ordering a product or service from MGS, you represent that you are eighteen (18) years of age or older. You are responsible for all charges incurred under your account, whether made by your or another person (including your children, family, friends, or others) and all payments are irrevocable and nonrefundable. In order for you to make a purchase, MGS uses the services of third parties, including Shopify. For information regarding Shopify and its Terms of Use and Privacy Policy, contact privacy@shopify.com. MGS expressly disclaims any and all responsibility for or related to the service provided by any such third party or the transactions you conduct or enter into with any such third party. If for any reason MGS does not receive payment for a purchase or service, MGS may: (a) seek collection of the outstanding amount owed and/or (b) seek legal action against you for the breach of these Terms of Use. You are also responsible for paying any governmental taxes imposed on your purchase from or use of the Site, including but not limited to, sales, use or value-added taxes. To the extent that MGS is obligated to collect such taxes, the applicable tax will be added to your billing account.

7. Privacy Policy. Any personal information that you provide to us through the Site is subject to our Privacy Policy. For more information, [click here to view the Privacy Policy](#), which is incorporated into these Terms by reference, as if set forth fully herein.

8. Disclaimer of Warranty and Liability. THIS SITE AND THE SITE MATERIAL ON AND MADE AVAILABLE THROUGH THIS SITE, AND THE SERVICES AND PRODUCTS OFFERED IN CONNECTION WITH THIS SITE, including any software, ARE MADE AVAILABLE ON AN "AS IS" and "as available" BASIS ONLY. USE OF THIS SITE IS ENTIRELY AT YOUR OWN RISK. MGS MAKES NO REPRESENTATIONS OR WARRANTIES, AND HEREBY DISCLAIMS ALL REPRESENTATIONS AND WARRANTIES, WITH RESPECT TO THIS SITE AND THE SITE MATERIAL ON AND MADE AVAILABLE THROUGH THIS SITE, AND THE SERVICES AND PRODUCTS OFFERED IN CONNECTION WITH THIS SITE, EXPRESS AND IMPLIED, WRITTEN AND ORAL, ARISING FROM COURSE OF DEALING, COURSE OF PERFORMANCE, USAGE OF TRADE, AND OTHERWISE, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, ACCURACY, TITLE, QUALITY, SYSTEMS INTEGRATION, AND NON-INFRINGEMENT, including that the Site will be uninterrupted, timely, secure, or error free, that the Site or its server(s) is free of viruses or other harmful components, that the Site, including the services will be available, or that content created on the Site is secure from unauthorized access. ANY AND ALL INFORMATION

PROVIDED BY MGS OR UNDER OR IN CONNECTION WITH THIS SITE IS PROVIDED WITH ALL FAULTS, AND THE ENTIRE RISK AS TO SATISFACTORY QUALITY, PERFORMANCE, ACCURACY, AND EFFORT IS WITH YOU. MGS makes no warranty regarding any software, goods, services, promotions, or the delivery of any software, goods or services, purchased, accessed or obtained through the Site or advertised through the MGS Site. No advice or information given by MGS, its employees or affiliates shall create a warranty.

9. Limitation of Liability. MGS AND ITS LICENSORS SHALL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, CONSEQUENTIAL, EXEMPLARY, EXTRA-CONTRACTUAL, OR PUNITIVE DAMAGES OF ANY KIND WHATSOEVER, INCLUDING LOST REVENUES OR LOST PROFITS, WHICH MAY OR DOES RESULT FROM THE USE OF, ACCESS TO, OR INABILITY TO USE THE SITE, THE USER CONTENT, THE SITE MATERIALS, SERVICES, PRODUCTS, DATA AND OTHER MATERIALS ON, IN AND MADE AVAILABLE THROUGH THE SITE, REGARDLESS OF LEGAL THEORY, WHETHER OR NOT YOU OR MGS HAD BEEN ADVISED OF THE POSSIBILITY OR PROBABILITY OF SUCH DAMAGES, AND EVEN IF THE REMEDIES OTHERWISE AVAILABLE FAIL OF THEIR ESSENTIAL PURPOSE. UNDER NO CIRCUMSTANCES WILL THE TOTAL LIABILITY OF MGS AND ITS LICENSORS TO YOU OR ANY OTHER PERSON OR ENTITY IN CONNECTION WITH, BASED UPON, OR ARISING FROM THE SITE, USER CONTENT, THE SITE MATERIALS ON, IN AND MADE AVAILABLE THROUGH THE SITE, OR THE SERVICES, PRODUCTS, DATA OR OTHER MATERIALS OFFERED IN CONNECTION THEREWITH EXCEED THE PRICE PAID BY YOU DURING THE PRECEDING YEAR FOR USE OF THE SITE AND THE SERVICES AND PRODUCTS. SOME STATES DO NOT ALLOW THE LIMITATION OR EXCLUSION OF LIABILITY FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES, SO THE ABOVE LIMITATION OR EXCLUSION MAY NOT APPLY TO YOU. IF ANY PART OF THIS LIMITATION ON LIABILITY IS FOUND TO BE INVALID OR UNENFORCEABLE FOR ANY REASON, THEN THE TOTAL LIABILITY OF MGS AND ITS LICENSORS SHALL NOT EXCEED TEN DOLLARS (\$10). If you are dissatisfied with the Site or with any of these Terms, or feel MGS has breached these Terms, your sole and exclusive remedy is to discontinue using the Site.

10. Indemnification. You shall indemnify MGS and its directors, officers, employees, agents, contractors and licensors ("MGS Indemnitees") against all claims, actions, suits, and other proceedings ("Claims") arising out of or incurred in connection with the Site and your use of the Site, the Site Materials or any services, product or data obtained through the Site, your fraud, violation of law, negligence, willful misconduct, or any other use of the Site, the User Content, the Site Materials, the services, products, information and other materials on, in and made available through the Site, except to the extent attributable to MGS, or any breach by you of these Terms and shall indemnify and hold MGS Indemnitees harmless from and against all judgments, losses, liabilities, damages, costs, and expenses (including reasonable attorneys' fees and attorneys' disbursements) arising out of or incurred in connection with such Claims. You may not settle any Claim without the prior written consent of MGS. MGS or its licensors may assume the defense of any Claim, at your sole cost and expense, and you shall cooperate in all reasonable respects in such defense. You shall have the right to employ separate counsel in any Claim and to participate in the defense thereof. If MGS or its licensors do not notify you that it elects to undertake the defense thereof, you shall have the right to defend the Claim with counsel reasonably acceptable to MGS, subject to the right of MGS to assume, at their sole cost and expense, the defense of any Claim at any time prior to the settlement or final determination thereof.

11. Availability. We use reasonable efforts to ensure that the Site is generally available. However, there will be occasions when access to the Site will be interrupted or unavailable. We will use reasonable commercial efforts to minimize such disruption where it is within our reasonable control. You agree that we shall not be liable to you for any modification, suspension or discontinuance of the Site. You are responsible for obtaining

access to the Site and that access may involve third-party fees (such as Internet service provider or airtime charges). You are responsible for those fees. In addition, you must provide and are responsible for all equipment necessary to access the Site.

12. Security. MGS uses reasonable efforts to ensure that the Site is generally available. However, there will be occasions when access to the Site will be interrupted or unavailable. MGS will use reasonable efforts to minimize such disruption where it is within its reasonable control. You agree that MGS shall not be liable to you for any modification, suspension or discontinuance of the Site. You understand that the technical processing and transmission of any Site content may be transferred unencrypted and involve (a) transmissions over various networks; and (b) changes to conform and adapt to technical requirements of connecting networks or devices. Please be advised that we do not guarantee that any information sent from our Site will be secure during transmission, nor can we guarantee the confidentiality of any communication or material transmitted to MGS via the Site or the Internet, including, for example, personal information such as your name or address.

13. Complaint Procedures. If you believe that any content or postings on this Site violates your intellectual property or other rights, please send to MGS, at the address listed in Section 18, a comprehensive detailed message setting forth the following information: (1) your name and the name of your company, if any; (2) your contact information, including your e-mail address; (3) the nature and substance of your complaint, the specific rights at issue, and your basis for making the complaint, including the content or posting that is objectionable; and (4) the following statement: The statements, representations, and assertions made in this message are true, complete, and accurate and I have the full legal authority to make each and every such statement, representation, and assertion and to make and be granted any demand made in this message.

14. Changes; Termination. You agree that we retain the right to amend these Terms and the Site at any time, for any reason, and with or without notice. Your continued use of the Site after any change is made to the Terms signifies your understanding of and intention and agreement to be bound by such change, whether or not you have actual notice thereof. It is your responsibility to monitor these Terms for changes. MGS may suspend or terminate your account and/or your ability to use the Site, or any services on the Site, for failure to comply with these Terms, for providing MGS with untrue or inaccurate information about yourself, for infringement upon MGS proprietary rights, or for any other reason whatsoever or for no reason.

15. Governing Law and Jurisdiction. These Terms represent the entire agreement between you and MGS with respect to the subject matter hereof, supersede any and all prior and contemporaneous agreements between us, and will be governed by and construed in accordance with the laws of the State of Colorado without reference to its conflict of law rules. By accessing, viewing, or using this Site or the Site Material, you consent to the jurisdiction of the federal and state courts located in the State of Colorado, and agree to accept service of process by mail and hereby waive any and all jurisdictional and venue defenses otherwise available. This Site is controlled and operated by MGS from within the United States. MGS makes no representation that this Site or the Site Material is appropriate or available for use in any location or territory, and access to this Site or the Site Material from locations or territories where it is illegal is prohibited. The waiver or failure of MGS to exercise in any respect any right provided hereunder shall not be deemed a waiver of such right in the future or a waiver of any of other rights established under these Terms.

16. Severability; Interpretation. If any provision of this Agreement is found by a court of competent jurisdiction to be invalid, the parties nevertheless agree that the court should endeavor to give effect to the parties' intentions as reflected in the provision, and the other provisions of this Agreement remain in full force and effect. When used herein, the words "includes" and "including" and their syntactical variations shall be deemed followed by the words "without limitation."

17. Contact. Mountain Goat Software, LLC is a Colorado limited liability company, headquartered in Broomfield, Colorado U.S.A. Any questions, comments or suggestions, including any report of violation of this Agreement should be provided to the Administrator as follows:

By Postal Mail: 1140 US Hwy 287 Suite 400-108 Broomfield, CO 80020

By E-mail: hello@mountaingoatsoftware.com

18. Miscellaneous. The Site is controlled and operated from within the United States. Without limiting anything else, MGS makes no representation that the Site, Site Materials, User Content, services, products, information or other materials available on, in, or through the Site is appropriate or available for use in other locations, and access to them from territories where they are illegal is prohibited. Those who choose to access the Site from other locations do so on their own volition and are responsible for compliance with applicable laws. The waiver or failure of MGS to exercise in any respect any right provided hereunder shall not be deemed a waiver of such right in the future or a waiver of any of other rights established under these Terms. Headings used in these Terms are for reference only and shall not affect the interpretation of these Terms. No person or entity not party to this agreement will be deemed to be a third party beneficiary of these Terms or any provision hereof. When used herein, the words "includes" and "including" and their syntactical variations shall be deemed followed by the words "without limitation."

Messaging Terms & Conditions

You agree to receive recurring automated promotional and personalized marketing text (e.g., SMS and MMS) messages (e.g. cart reminders, sale notices) from MGS, including text messages that may be sent using an automatic telephone dialing system, to the mobile telephone number you provided when signing up or any other number that you designate. Consent to receive automated marketing text messages is not a condition of any purchase. Msg & Data rates may apply.

Message frequency will vary. MGS reserves the right to alter the frequency of messages sent at any time, so as to increase or decrease the total number of sent messages. MGS also reserves the right to change the short code or phone number from which messages are sent and we will notify you when we do so.

Not all mobile devices or handsets may be supported and our messages may not be deliverable in all areas. MGS, its service providers and the mobile carriers supported by the program are not liable for delayed or undelivered messages.

You also agree to our MGS Terms of Use and MGS Privacy Policy.

We are able to deliver messages to the following mobile phone carriers: Major carriers: AT&T, Verizon Wireless, Sprint, T-Mobile, MetroPCS, U.S. Cellular, Alltel, Boost Mobile, Nextel, and Virgin Mobile. Minor carriers: Alaska Communications Systems (ACS), Appalachian Wireless (EKN), Bluegrass Cellular, Cellular One of East Central IL (ECIT), Cellular One of Northeast Pennsylvania, Cincinnati Bell Wireless, Cricket, Coral Wireless (Mobi PCS), COX, Cross, Element Mobile (Flat Wireless), Epic Touch (Elkhart Telephone), GCI, Golden State, Hawkeye (Chat Mobility), Hawkeye (NW Missouri), Illinois Valley Cellular, Inland Cellular, iWireless (Iowa Wireless), Keystone Wireless (Immix Wireless/PC Man), Mosaic (Consolidated or CTC Telecom), Nex-Tech Wireless, NTelos, Panhandle Communications, Pioneer, Plateau (Texas RSA 3 Ltd), Revol, RINA, Symmetry (TMP Corporation), Thumb Cellular, Union Wireless, United Wireless, Viaero Wireless, and West Central (WCC or 5 Star Wireless).

Cancellation

Text the keyword STOP, STOPALL, END, CANCEL, UNSUBSCRIBE or QUIT to our shortcode to cancel. After texting STOP, STOPALL, END, CANCEL, UNSUBSCRIBE or QUIT to our shortcode you will receive one additional message confirming that your request has been processed. You acknowledge that our text message platform may not recognize and respond to unsubscribe requests that do not include the STOP, STOPALL, END, CANCEL, UNSUBSCRIBE or QUIT keyword commands and agree that MGS and its service providers will have no liability for failing to honor such requests. If you unsubscribe from one of our text message programs, you may continue to receive text messages from MGS through any other programs you have joined until you separately unsubscribe from those programs.

Help

Text the keyword HELP to our shortcode to return customer care contact information.

Messaging Dispute Resolution

1. General. In the interest of resolving disputes between you and MGS in the most expedient and cost effective manner, you and MGS agree that any dispute arising out of or in any way related to these messaging terms and conditions ("Messaging Terms") or your receipt of text messages from MGS or its service providers will be resolved by binding arbitration. Arbitration is less formal than a lawsuit in court. Arbitration uses a neutral arbitrator instead of a judge or jury, may allow for more limited discovery than in court, and can be subject to very limited review by courts. Arbitrators can award the same damages and relief that a court can award. This agreement to arbitrate disputes includes all claims arising out of or in any way related to these Messaging Terms, or your receipt of text messages from MGS or its service providers whether based in contract, tort, statute, fraud, misrepresentation, or any other legal theory, and regardless of when a claim arises. YOU UNDERSTAND AND AGREE THAT, BY AGREEING TO THESE MESSAGING TERMS, YOU AND MGS ARE EACH WAIVING THE RIGHT TO A TRIAL BY JURY OR TO PARTICIPATE IN A CLASS ACTION AND THAT THESE MESSAGING TERMS SHALL BE SUBJECT TO AND GOVERNED BY THE FEDERAL ARBITRATION ACT.
2. Exceptions. Notwithstanding subsection (a) above, nothing in these Messaging Terms will be deemed to waive, preclude, or otherwise limit the right of you or MGS to: (i) bring an individual action in small claims court; (ii) pursue an enforcement action through the applicable federal, state, or local agency if that action is available; (iii) seek injunctive relief in aid of arbitration from a court of competent jurisdiction; or (iv) file suit in a court of law to address an intellectual property infringement claim.
3. Arbitrator. Any arbitration between you and MGS will be governed by the Federal Arbitration Act and the Commercial Dispute Resolution Procedures and Supplementary Procedures for Consumer Related Disputes (collectively, "AAA Rules") of the American Arbitration Association ("AAA"), as modified by these Messaging Terms, and will be administered by the AAA. The AAA Rules and filing forms are available online at www.adr.org, by calling the AAA at 1-800-778-7879. The arbitrator has exclusive authority to resolve any dispute relating to the interpretation, applicability, or enforceability of this binding arbitration agreement.
4. Notice; Process. If you or MGS intends to seek arbitration, then the party seeking arbitration must first send a written notice of the dispute to the other party by U.S. Mail ("Notice"). MGS address for Notice is above. The Notice must: (i) describe the nature and basis of the claim or dispute; and (ii) set forth the specific relief sought ("Demand"). You and MGS will make good faith efforts to resolve the claim directly, but if you and MGS do not reach an agreement to do so within 30 days after the Notice is received, you or MGS may commence an arbitration proceeding. During the arbitration, the amount of any settlement offer made by you or MGS must not be disclosed to the arbitrator until after the

arbitrator makes a final decision and award, if any. (e) Fees. If you commence arbitration in accordance with these Messaging Terms, MGS will reimburse you for your payment of the filing fee, unless your claim is for more than \$1,000 or as set forth below, in which case the payment of any fees will be decided by the AAA Rules. If the claim is for \$1,000 or less, you may choose whether the arbitration will be conducted: (i) solely on the basis of documents submitted to the arbitrator; (ii) through a non-appearance based telephone hearing; or (iii) by an in-person hearing as established by the AAA Rules. If the arbitrator finds that either the substance of your claim or the relief sought in the Demand is frivolous or brought for an improper purpose (as measured by the standards set forth in Federal Rule of Civil Procedure 11(b)), then the payment of all fees will be governed by the AAA Rules. In that case, you agree to reimburse MGS for all monies previously disbursed by it that are otherwise your obligation to pay under the AAA Rules. Regardless of the manner in which the arbitration is conducted, the arbitrator must issue a reasoned written decision sufficient to explain the essential findings and conclusions on which the decision and award, if any, are based. You and MGS agree that such written decision, and information exchanged during arbitration, will be kept confidential except to the extent necessary to enforce or permit limited judicial review of the award. The arbitrator may make rulings and resolve disputes as to the payment and reimbursement of fees or expenses at any time during the proceeding and upon request from you or MGS made within 14 days of the arbitrator's ruling on the merits.

5. No Class Actions. YOU AND MGS AGREE THAT EACH MAY BRING CLAIMS AGAINST THE OTHER ONLY IN AN INDIVIDUAL CAPACITY AND NOT AS A PLAINTIFF OR CLASS MEMBER IN ANY PURPORTED CLASS OR REPRESENTATIVE PROCEEDING. Further, unless both you and MGS agree otherwise in a signed writing, the arbitrator may not consolidate more than one person's claims, and may not otherwise preside over any form of a representative or class proceeding.
6. Modifications to this Arbitration Provision. Notwithstanding anything to the contrary in these Messaging Terms, if MGS makes any future change to this arbitration provision, other than a change to MGS address for Notice, you may reject the change by sending us written notice within 30 days of the change to MGS address for Notice, in which case this arbitration provision, as in effect immediately prior to the changes you rejected, will continue to govern any disputes between you and MGS.
7. Enforceability. If an arbitrator decides that applicable law precludes enforcement of any of the limitations of subsection (f) above (addressing class, representative and consolidated proceedings) as to a particular claim for relief, then that claim (and only that claim) must be severed from the arbitration and brought in court. If any other provision of these Messaging Terms is found to be unenforceable, the applicable provision shall be deemed stricken and the remainder of these Messaging Terms shall remain in full force and effect.

This page was last modified on: September 14th, 2021

Copyright © 2021 Mountain Goat Software, LLC; All rights reserved.

Learn About Agile

New to Agile and Scrum?

Agile & Scrum Training

Online Certifications

More...

About Us

Scrum

Video Courses

Contact Us

User Stories

In-Person Certifications

Our Students

Planning Poker

On-Site Courses

Blog

Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations		PDUs and SEUs
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

Mountain Goat Software Privacy Policy

Privacy Policy of www.mountaingoatsoftware.com

This Application collects some Personal Data from its Users.

Users may be subject to different protection standards and broader standards may therefore apply to some. In order to learn more about the protection criteria, Users can refer to the applicability section.

This document contains a section dedicated to California consumers and their privacy rights.

This document contains a section dedicated to Virginia consumers and their privacy rights.

This document contains a section dedicated to Brazilian Users and their privacy rights.

This document can be printed for reference by using the print command in the settings of any browser.

POLICY SUMMARY

Personal Data processed for the following purposes and using the following services:

Analytics

[Google Analytics](#), [Amplitude Analytics](#) and [Google Ads conversion tracking](#)

Personal Data: Cookies; Usage Data

Contacting the User

Contact form

Personal Data: email address; first name

Mailing list or newsletter

Personal Data: email address; first name; Usage Data

Content commenting

Disqus

Personal Data: Cookies; Usage Data; various types of Data as specified in the privacy policy of the service

Displaying content from external platforms

Adobe Fonts

Personal Data: Usage Data; various types of Data as specified in the privacy policy of the service

Wistia widget

Personal Data: Tracker; Usage Data

Interaction with external social networks and platforms

[Twitter Tweet button and social widgets](#), [Facebook Like button and social widgets](#) and [LinkedIn button and social widgets](#)

Personal Data: Cookies; Usage Data

Managing contacts and sending messages

Drip

Personal Data: email address; first name; last name; Usage Data; username

Managing support and contact requests

Help Scout

Personal Data: various types of Data as specified in the privacy policy of the service

Platform services and hosting

Shopify

Personal Data: various types of Data as specified in the privacy policy of the service

Remarketing and behavioral targeting

Facebook Custom Audience

Personal Data: Cookies; email address

Facebook Remarketing

Personal Data: Cookies; Usage Data

Tag Management

Google Tag Manager

Personal Data: Usage Data

Traffic optimization and distribution

Cloudflare

Personal Data: Cookies; various types of Data as specified in the privacy policy of the service

Information on opting out of interest-based advertising

In addition to any opt-out feature provided by any of the services listed in this document, Users may learn more on how to generally opt out of interest-based advertising within the dedicated section of the Cookie Policy.

Further information about the processing of Personal Data

Scrum Alliance

For students that attend certified courses, i.e. Certified ScrumMaster or Certified Scrum Product Owner, your name and email address will be provided to the Scrum Alliance to process your certification.

<https://www.scrumalliance.org/privacy-policy>

Memberful

Members of our Agile Mentors Community and customers taking our classes use a service from Memberful for account management including name and email address information.

Contact information

Owner and Data Controller

Mountain Goat Software
1140 US Hwy 287
Suite 400-108
Broomfield, CO 80020

Owner contact email: hello@mountaingoatsoftware.com

FULL POLICY

Owner and Data Controller

Mountain Goat Software
1140 US Hwy 287
Suite 400-108
Broomfield, CO 80020

Owner contact email: hello@mountaingoatsoftware.com

Types of Data collected

Among the types of Personal Data that this Application collects, by itself or through third parties, there are: Cookies; Usage Data; first name; email address; last name; username.

Complete details on each type of Personal Data collected are provided in the dedicated sections of this privacy policy or by specific explanation texts displayed prior to the Data collection.

Personal Data may be freely provided by the User, or, in case of Usage Data, collected automatically when using this Application. Unless specified otherwise, all Data requested by this Application is mandatory and failure to provide this Data may make it impossible for this Application to provide its services. In cases where this Application specifically states that some Data is not mandatory, Users are free not to communicate this Data without consequences to the availability or the functioning of the Service. Users who are uncertain about which Personal Data is mandatory are welcome to contact the Owner. Any use of Cookies or of other tracking tools by this Application or by the owners of third-party services used by this Application serves the purpose of providing the Service required by the User, in addition to any other purposes described in the present document and in the Cookie Policy, if available. Users are responsible for any third-party Personal Data obtained, published or shared through this Application and confirm that they have the third party's consent to provide the Data to the Owner.

Mode and place of processing the Data

Methods of processing

The Owner takes appropriate security measures to prevent unauthorized access, disclosure, modification, or unauthorized destruction of the Data.

The Data processing is carried out using computers and/or IT enabled tools, following organizational procedures and modes strictly related to the purposes indicated. In addition to the Owner, in some cases, the Data may be accessible to certain types of persons in charge, involved with the operation of this Application (administration, sales, marketing, legal, system administration) or external parties (such as third-party technical service providers, mail carriers, hosting providers, IT companies, communications agencies) appointed, if necessary, as Data Processors by the Owner. The updated list of these parties may be requested from the Owner at any time.

Legal basis of processing

The Owner may process Personal Data relating to Users if one of the following applies:

- Users have given their consent for one or more specific purposes. Note: Under some legislations the Owner may be allowed to process Personal Data until the User objects to such processing (Opt-out), without having to rely on consent or any other of the following legal bases. This, however, does not apply, whenever the processing of Personal Data is subject to European data protection law;
- provision of Data is necessary for the performance of an agreement with the User and/or for any pre-contractual obligations thereof;
- processing is necessary for compliance with a legal obligation to which the Owner is subject;
- processing is related to a task that is carried out in the public interest or in the exercise of official authority vested in the Owner;
- processing is necessary for the purposes of the legitimate interests pursued by the Owner or by a third party.

In any case, the Owner will gladly help to clarify the specific legal basis that applies to the processing, and in particular whether the provision of Personal Data is a statutory or contractual requirement, or a requirement necessary to enter into a contract.

Place

The Data is processed at the Owner's operating offices and in any other places where the parties involved in the processing are located.

Depending on the User's location, data transfers may involve transferring the User's Data to a country other than their own. To find out more about the place of processing of such transferred Data, Users can check the section containing details about the processing of Personal Data.

If broader protection standards are applicable, Users are also entitled to learn about the legal basis of Data transfers to a country outside the European Union or to any international organization governed by public international law or set up by two or more countries, such as the UN, and about the security measures taken by the Owner to safeguard their Data.

If any such transfer takes place, Users can find out more by checking the relevant sections of this document or inquire with the Owner using the information provided in the contact section.

Retention time

Personal Data shall be processed and stored for as long as required by the purpose they have been collected for.

Therefore:

Personal Data collected for purposes related to the performance of a contract between the Owner and the User shall be retained until such contract has been fully performed.

Personal Data collected for the purposes of the Owner's legitimate interests shall be retained as long as needed to fulfill such purposes. Users may find specific information regarding the legitimate interests pursued by the Owner within the relevant sections of this document or by contacting the Owner.

The Owner may be allowed to retain Personal Data for a longer period whenever the User has given consent to such processing, as long as such consent is not withdrawn. Furthermore, the Owner may be obliged to retain Personal Data for a longer period whenever required to do so for the performance of a legal obligation or upon order of an authority.

Once the retention period expires, Personal Data shall be deleted. Therefore, the right of access, the right to erasure, the right to rectification and the right to data portability cannot be enforced after expiration of the retention period.

The purposes of processing

The Data concerning the User is collected to allow the Owner to provide its Service, comply with its legal obligations, respond to enforcement requests, protect its rights and interests (or those of its Users or third parties), detect any malicious or fraudulent activity, as well as the following: Analytics, Contacting the User, Content commenting, Traffic optimization and distribution, Interaction with external social networks and platforms, Platform services and hosting, Tag Management, Displaying content from external platforms, Managing support and contact requests, Remarketing and behavioral targeting and Managing contacts and sending messages.

For specific information about the Personal Data used for each purpose, the User may refer to the section "Detailed information on the processing of Personal Data".

Detailed information on the processing of Personal Data

Personal Data is collected for the following purposes and using the following services:

Analytics

The services contained in this section enable the Owner to monitor and analyze web traffic and can be used to keep track of User behavior.

Google Analytics (Google Inc.)

Google Analytics is a web analysis service provided by Google Inc. ("Google"). Google utilizes the Data collected to track and examine the use of this Application, to prepare reports on its activities and share them with other Google services. Google may use the Data collected to contextualize and personalize the ads of its own advertising network.

Personal Data processed: Cookies; Usage Data.

Place of processing: US □ Privacy Policy □ Opt Out.

Category of personal information collected according to the CCPA: internet information.

This processing constitutes:

a sale according to the CCPA and the VCDPA

Amplitude Analytics (Amplitude Inc.)

Amplitude Analytics is an analytics service provided by Amplitude Inc.

Personal Data processed: Cookies; Usage Data.

Place of processing: United States □ Privacy Policy.

Category of personal information collected according to the CCPA: internet information.

This processing constitutes:

a sale according to the CCPA and the VCDPA

Google Ads conversion tracking (Google LLC)

Google Ads conversion tracking is an analytics service provided by Google LLC that connects data from the Google Ads advertising network with actions performed on this Application.

Personal Data processed: Cookies; Usage Data.

Place of processing: United States □ Privacy Policy.

Category of personal information collected according to the CCPA: internet information.

This processing constitutes:

a sale according to the CCPA and the VCDPA

Contacting the User

Contact form (This Application)

By filling in the contact form with their Data, the User authorizes this Application to use these details to reply to requests for information, quotes or any other kind of request as indicated by the form's header.

Personal Data processed: email address; first name.

Category of personal information collected according to the CCPA: identifiers.

This processing constitutes:

a sale according to the VCDPA

Mailing list or newsletter (this Application)

By registering on the mailing list or for the newsletter, the User's email address will be added to the contact list of those who may receive email messages containing information of commercial or promotional nature concerning this Application. Your email address might also be added to this list as a result of signing up to this Application or after making a purchase.

Personal Data processed: email address; first name; Usage Data.

Category of personal information collected according to the COPA: identifiers; internet information.

This processing constitutes:

a sale according to the VCDPA

Content commenting

Content commenting services allow Users to make and publish their comments on the contents of this Application.

Depending on the settings chosen by the Owner, Users may also leave anonymous comments. If there is an email address among the Personal Data provided by the User, it may be used to send notifications of comments on the same content. Users are responsible for the content of their own comments.

If a content commenting service provided by third parties is installed, it may still collect web traffic data for the pages where the comment service is installed, even when Users do not use the content commenting service.

Disqus (Disqus)

Disqus is a content commenting service provided by Big Heads Labs Inc.

Personal Data processed: Cookies; Usage Data; various types of Data as specified in the privacy policy of the service.

Place of processing: US | Privacy Policy | Opt out.

Category of personal information collected according to the COPA: internet information.

This processing constitutes:

a sale according to the CCPA

Displaying content from external platforms

This type of service allows you to view content hosted on external platforms directly from the pages of this Application and interact with them.

This type of service might still collect web traffic data for the pages where the service is installed, even when Users do not use it.

Adobe Fonts (Adobe Systems Incorporated)

Adobe Fonts is a typeface visualization service provided by Adobe Systems Incorporated that allows this Application to incorporate content of this kind on its pages.

Personal Data processed: Usage Data; various types of Data as specified in the privacy policy of the service.

Place of processing: US | Privacy Policy.

Category of personal information collected according to the COPA: internet information.

This processing constitutes:

a sale according to the CCPA and the VCDPA

Wistia widget (Wistia, Inc.)

Wistia is a video content visualization service provided by Wistia, Inc. that allows this Application to incorporate content of this kind on its pages.

Personal Data processed: Tracker; Usage Data.

Place of processing: United States | Privacy Policy.

Category of personal information collected according to the COPA: internet information.

This processing constitutes:

a sale according to the CCPA and the VCDPA

Interaction with external social networks and platforms

This type of service allows interaction with social networks or other external platforms directly from the pages of this Application. The interaction and information obtained through this Application are always subject to the User's privacy settings for each social network.

This type of service might still collect traffic data for the pages where the service is installed, even when Users do not use it. It is recommended to log out from the respective services in order to make sure that the processed data on this Application isn't being connected back to the User's profile.

Twitter Tweet button and social widgets (Twitter, Inc.)

The Twitter Tweet button and social widgets are services allowing interaction with the Twitter social network provided by Twitter, Inc.

Personal Data processed: Cookies; Usage Data.

Place of processing: US & Privacy Policy.

Category of personal information collected according to the CCPA: internet information.

This processing constitutes:

a sale according to the CCPA and the VCDPA

Facebook Like button and social widgets (Facebook, Inc.)

The Facebook Like button and social widgets are services allowing interaction with the Facebook social network provided by Facebook, Inc.

Personal Data processed: Cookies; Usage Data.

Place of processing: US & Privacy Policy.

Category of personal information collected according to the CCPA: internet information.

This processing constitutes:

a sale according to the CCPA and the VCDPA

LinkedIn button and social widgets (LinkedIn Corporation)

The LinkedIn button and social widgets are services allowing interaction with the LinkedIn social network provided by LinkedIn Corporation.

Personal Data processed: Cookies; Usage Data.

Place of processing: US & Privacy Policy.

Category of personal information collected according to the CCPA: internet information.

This processing constitutes:

a sale according to the CCPA and the VCDPA

Managing contacts and sending messages

This type of service makes it possible to manage a database of email contacts, phone contacts or any other contact information to communicate with the User.

These services may also collect data concerning the date and time when the message was viewed by the User, as well as when the User interacted with it, such as by clicking on links included in the message.

Drip (Avenue 81, Inc.)

Drip is an email address management and message sending service provided by Avenue 81, Inc.

Personal Data processed: email address; first name; last name; Usage Data; username.

Place of processing: United States & Privacy Policy.

Category of personal information collected according to the CCPA: identifiers; internet information.

This processing constitutes:

a sale according to the CCPA and the VCDPA

Managing support and contact requests

This type of service allows this Application to manage support and contact requests received via email or by other means, such as the contact form.

The Personal Data processed depend on the information provided by the User in the messages and the means used for communication (e.g. email address).

Help Scout (Brightworks Inc.)

Help Scout is a support and contact request management service provided by Help Scout Inc. .

Personal Data processed: various types of Data as specified in the privacy policy of the service.

Place of processing: US & Privacy Policy.

Category of personal information collected according to the CCPA: internet information.

This processing constitutes:

a sale according to the CCPA and the VCDPA

Platform services and hosting

These services have the purpose of hosting and running key components of this Application, therefore allowing the provision of this Application from within a unified platform. Such platforms provide a wide range of tools to the Owner (e.g. analytics, user registration, commenting, database management, e-commerce, payment processing) that imply the collection and handling of Personal Data. Some of these services work through geographically distributed servers, making it difficult to determine the actual location where the Personal Data are stored.

Shopify (Shopify Inc.)

Shopify is a platform provided by Shopify Inc. that allows the Owner to build, run and host an e-commerce website.

Personal Data processed: various types of Data as specified in the privacy policy of the service.

Place of processing: Canada (Privacy Policy).

Category of personal information collected according to the CCPA: internet information.

This processing constitutes:

a sale according to the CCPA

Remarketing and behavioral targeting

This type of service allows this Application and its partners to inform, optimize and serve advertising based on past use of this Application by the User.

This activity is facilitated by tracking Usage Data and by using Trackers to collect information which is then transferred to the partners that manage the remarketing and behavioral targeting activity.

Some services offer a remarketing option based on email address lists.

In addition to any opt-out feature provided by any of the services below, Users may opt out by visiting the Network Advertising Initiative opt-out page.

Users may also opt-out of certain advertising features through applicable device settings, such as the device advertising settings for mobile phones or ads settings in general.

Facebook Custom Audience (Facebook, Inc.)

Facebook Custom Audience is a remarketing and behavioral targeting service provided by Facebook, Inc. that connects the activity of this Application with the Facebook advertising network.

Personal Data processed: Cookies; email address.

Place of processing: US (Privacy Policy) (Opt Out).

Category of personal information collected according to the CCPA: identifiers; internet information.

This processing constitutes:

a sale according to the CCPA and the VCDPA

a sharing according to the CCPA

targeted advertising according to the VCDPA

Facebook Remarketing (Facebook, Inc.)

Facebook Remarketing is a remarketing and behavioral targeting service provided by Facebook, Inc. that connects the activity of this Application with the Facebook advertising network.

Personal Data processed: Cookies; Usage Data.

Place of processing: US (Privacy Policy) (Opt Out).

Category of personal information collected according to the CCPA: internet information.

This processing constitutes:

a sale according to the CCPA and the VCDPA

a sharing according to the CCPA

targeted advertising according to the VCDPA

Tag Management

This type of service helps the Owner to manage the tags or scripts needed on this Application in a centralized fashion.

This results in the Users' Data flowing through these services, potentially resulting in the retention of this Data.

Google Tag Manager (Google Inc.)

Google Tag Manager is a tag management service provided by Google LLC or by Google Ireland Limited, depending on how the Owner manages the Data processing.

Personal Data processed: Usage Data.

Place of processing: US & Privacy Policy.

Category of personal information collected according to the CCPA: internet information.

This processing constitutes:

a sale according to the CCPA

Traffic optimization and distribution

This type of service allows this Application to distribute their content using servers located across different countries and to optimize their performance.

Which Personal Data are processed depends on the characteristics and the way these services are implemented. Their function is to filter communications between this Application and the User's browser.

Considering the widespread distribution of this system, it is difficult to determine the locations to which the contents that may contain Personal Information of the User are transferred.

Cloudflare (Cloudflare)

Cloudflare is a traffic optimization and distribution service provided by Cloudflare Inc.

The way Cloudflare is integrated means that it filters all the traffic through this Application, i.e., communication between this Application and the User's browser, while also allowing analytical data from this Application to be collected.

Personal Data processed: Cookies; various types of Data as specified in the privacy policy of the service.

Place of processing: US & Privacy Policy.

Category of personal information collected according to the CCPA: internet information.

This processing constitutes:

a sale according to the CCPA and the VCDPA

Information on opting out of interest-based advertising

In addition to any opt-out feature provided by any of the services listed in this document, Users may learn more on how to generally opt out of interest-based advertising within the dedicated section of the Cookie Policy.

Further information about the processing of Personal Data

Scrum Alliance

For students that attend certified courses, i.e. Certified ScrumMaster or Certified Scrum Product Owner, your name and email address will be provided to the Scrum Alliance to process your certification.

<https://www.scrumalliance.org/privacy-policy>

This processing constitutes a sale based on the definition under the CCPA. In addition to the information in this clause, the User can find information regarding how to opt out of the sale in the section detailing the rights of Californian consumers.

Memberful

Members of our Agile Mentors Community and customers taking our classes use a service from Memberful for account management, including name and email address information.

The rights of Users

Users may exercise certain rights regarding their Data processed by the Owner.

Users entitled to broader protection standards may exercise any of the rights described below. In all other cases, Users may inquire with the Owner to find out which rights apply to them.

In particular, Users have the right to do the following:

Withdraw their consent at any time. Users have the right to withdraw consent where they have previously given their consent to the processing of their Personal Data.

Object to processing of their Data. Users have the right to object to the processing of their Data if the processing is carried out on a legal basis other than consent. Further details are provided in the dedicated section below.

Access their Data. Users have the right to learn if Data is being processed by the Owner, obtain disclosure regarding certain aspects of the processing and obtain a copy of the Data undergoing processing.

Verify and seek rectification. Users have the right to verify the accuracy of their Data and ask for it to be updated or corrected.

Restrict the processing of their Data. Users have the right, under certain circumstances, to restrict the processing of their Data. In this case, the Owner will not process their Data for any purpose other than storing.

Have their Personal Data deleted or otherwise removed. Users have the right, under certain circumstances, to obtain the erasure of their Data from the Owner.

Receive their Data and have it transferred to another controller. Users have the right to receive their Data in a structured, commonly used and machine readable format and, if technically feasible, to have it transmitted to another controller without any hindrance. This provision is applicable provided that the Data is processed by automated means and that the processing is based on the User's consent, on a contract which the User is part of or on pre-contractual obligations thereof.

Lodge a complaint. Users have the right to bring a claim before their competent data protection authority.

Details about the right to object to processing

Where Personal Data is processed for a public interest, in the exercise of an official authority vested in the Owner or for the purposes of the legitimate interests pursued by the Owner, Users may object to such processing by providing a ground related to their particular situation to justify the objection.

Users must know that, however, should their Personal Data be processed for direct marketing purposes, they can object to that processing at any time without providing any justification. To learn, whether the Owner is processing Personal Data for direct marketing purposes, Users may refer to the relevant sections of this document.

How to exercise these rights

Any requests to exercise User rights can be directed to the Owner through the contact details provided in this document. These requests can be exercised free of charge and will be addressed by the Owner as early as possible and always within one month.

Applicability of broader protection standards

While most provisions of this document concern all Users, some provisions expressly only apply if the processing of Personal Data is subject to broader protection standards.

Such broader protection standards apply when the processing:

- is performed by an Owner based within the EU;
- concerns the Personal Data of Users who are in the EU and is related to the offering of paid or unpaid goods or services, to such Users;
- concerns the Personal Data of Users who are in the EU and allows the Owner to monitor such Users' behavior taking place in the EU.

Cookie Policy

This Application uses Trackers. To learn more, the User may consult the Cookie Policy.

Additional information about Data collection and processing

Legal action

The User's Personal Data may be used for legal purposes by the Owner in Court or in the stages leading to possible legal action arising from improper use of this Application or the related Services.

The User declares to be aware that the Owner may be required to reveal personal data upon request of public authorities.

Additional information about User's Personal Data

In addition to the information contained in this privacy policy, this Application may provide the User with additional and contextual information concerning particular Services or the collection and processing of Personal Data upon request.

System logs and maintenance

For operation and maintenance purposes, this Application and any third-party services may collect files that record interaction with this Application (System logs) use other Personal Data (such as the IP Address) for this purpose.

Information not contained in this policy

More details concerning the collection or processing of Personal Data may be requested from the Owner at any time. Please see the contact information at the beginning of this document.

How 'Do Not Track' requests are handled

This Application does not support "Do Not Track" requests.

To determine whether any of the third-party services it uses honor the "Do Not Track" requests, please read their privacy policies.

Changes to this privacy policy

The Owner reserves the right to make changes to this privacy policy at any time by notifying its Users on this page and possibly within this Application and/or - as far as technically and legally feasible - sending a notice to Users via any contact information available to the Owner. It is strongly recommended to check this page often, referring to the date of the last modification listed at the bottom.

Should the changes affect processing activities performed on the basis of the User's consent, the Owner shall collect new consent from the User, where required.

Information for Californian consumers

This part of the document integrates with and supplements the information contained in the rest of the privacy policy and is provided by the business running this Application and, if the case may be, its parent, subsidiaries and affiliates (for the purposes of this section referred to collectively as "We", "Us", "Our").

The provisions contained in this section apply to all Users (Users are referred to below, simply as "You", "Your", "Yours"), who are consumers residing in the state of California, United States of America, according to the "California Consumer Privacy Act of 2018" (the "CCPA"), as updated by the "California Privacy Rights Act" (the "CPRA") and subsequent regulations. For such consumers, these provisions supersede any other possibly divergent or conflicting provisions contained in the privacy policy.

This part of the document uses the terms "personal information" (and "sensitive personal information") as defined in the California Consumer Privacy Act (CCPA).

Notice at collection

Categories of personal information collected, used, sold, or shared

In this section we summarize the categories of personal information that we've collected, used, sold, or shared and the purposes thereof.

You can read about these activities in detail in the section titled "Detailed information on the processing of Personal Data" within this document.

Information we collect: the categories of personal information we collect

We have collected the following categories of personal information about you: identifiers and internet information.

We have collected the following categories of sensitive personal information: username

We will not collect additional categories of personal information without notifying you.

YOUR RIGHT TO LIMIT THE USE OR DISCLOSURE OF YOUR SENSITIVE PERSONAL INFORMATION AND HOW YOU CAN EXERCISE IT

You have the right to request that we limit the use or disclosure of your sensitive personal information to only that which is necessary to perform the services or provide the goods, as is reasonably expected by an average consumer.

We can also use your sensitive personal information to perform specific purposes set forth by the law (such as, including but not limited to, helping to ensure security and integrity; undertaking activities to verify or maintain the quality or safety of our service) and as authorized by the relevant regulations.

Outside of the aforementioned specific purposes, you have the right to freely request, at any time, that we do not use or disclose your sensitive personal information. This means that whenever you ask us to stop using your sensitive personal information, we will abide by your request and we will instruct our service providers and contractors to do the same.

To fully exercise your right to limit the use or disclosure of your sensitive personal information you can contact us at any time, using the contact details provided in this document.

For a simplified opt-out method you can also use the privacy choices link provided on this Application.

We use any personal information collected from you in connection with the submission of your request solely for the purposes of complying with the request.

Once you have exercised this right, we are required to wait at least 12 months before asking whether you have changed your mind.

What are the purposes for which we use your personal information?

We may use your personal information to allow the operational functioning of this Application and features thereof ("business purposes"). In such cases, your personal information will be processed in a fashion necessary and proportionate to the business purpose for which it was collected, and strictly within the limits of compatible operational purposes.

We may also use your personal information for other reasons such as for commercial purposes (as indicated within the section "Detailed information on the processing of Personal Data" within this document), as well as for complying with the law and defending our rights before the competent authorities where our rights and interests are threatened or we suffer an actual damage.

We won't process your information for unexpected purposes, or for purposes incompatible with the purposes originally disclosed, without your consent.

How long do we keep your personal information?

Unless stated otherwise inside the **Detailed information on the processing of Personal Data** section, we will not retain your personal information for longer than is reasonably necessary for the purpose(s) they have been collected for.

How we collect information: what are the sources of the personal information we collect?

We collect the above-mentioned categories of personal information, either directly or indirectly, from you when you use this Application.

For example, you directly provide your personal information when you submit requests via any forms on this Application. You also provide personal information indirectly when you navigate this Application, as personal information about you is automatically observed and collected.

Finally, we may collect your personal information from third parties that work with us in connection with the Service or with the functioning of this Application and features thereof.

How we use the information we collect: disclosing of your personal information with third parties for a business purpose

For our purposes, the word **third party** means a person who is not any of the following: a service provider or a contractor, as defined by the CCPA.

We disclose your personal information with the third parties **listed in detail in the section titled "Detailed information on the processing of Personal Data" within this document**. These third parties are grouped and categorized in accordance with the different purposes of processing.

Sale or sharing of your personal information

For our purposes, the word **Sale** means any **selling, renting, releasing, disclosing, disseminating, making available, transferring or otherwise communicating orally, in writing, or by electronic means, a consumer's personal information by the business to a third party, for monetary or other valuable consideration**, as defined by the CCPA.

This means that, for example, a sale can happen whenever an application runs ads, or makes statistical analyses on the traffic or views, or simply because it uses tools such as social network plugins and the like.

For our purposes, the word **Sharing** means any **sharing, renting, releasing, disclosing, disseminating, making available, transferring, or otherwise communicating orally, in writing, or by electronic or other means, a consumer's personal information by the business to a third party for cross-context behavioral advertising, whether or not for monetary or other valuable consideration, including transactions between a business and a third party for cross-context behavioral advertising for the benefit of a business in which no money is exchanged**, as defined by the CCPA.

Please note that the exchange of personal information with a service provider pursuant to a written contract that meets the requirements set by the CCPA, does not constitute a sale or sharing of your personal information.

YOUR RIGHT TO OPT OUT OF THE SALE OR SHARING OF YOUR PERSONAL INFORMATION AND HOW YOU CAN EXERCISE IT

We sell or share your personal information with the third parties **listed in detail in the section titled "Detailed information on the processing of Personal Data" within this document**. These third parties are grouped and categorized in accordance with the different purposes of processing.

You have the right to opt out of the sale or sharing of your personal information. This means that whenever you request us to stop selling or sharing your personal information, we will abide by your request.

Such requests can be made freely, at any time, without submitting any verifiable request.

To fully exercise your right to opt out, you can contact us at any time using the contact details provided in this document.

For a simplified opt-out method you can also use the privacy choices link provided on this Application.

If you want to submit requests to opt out of the sale or sharing of personal information via a user-enabled global privacy control, like the Global Privacy Control (GPC), you are free to do so and we will abide by such request. The GPC consists of a setting or extension in the browser or mobile device and acts as a mechanism that websites can use to indicate they support the GPC signal. If you want to use GPC, you can download and enable it via a participating browser or browser extension. More information about downloading GPC is available [here](#).

We use any personal information collected from you in connection with the submission of your opt-out request solely for the purposes of complying with the opt-out request.

Once you have opted out, we are required to wait at least 12 months before asking whether you have changed your mind.

Your privacy rights under the California Consumer Privacy Act and how to exercise them

The right to access personal information: the right to know and to portability

You have the right to request that we disclose to you:

- the categories of personal information that we collect about you;
- the sources from which the personal information is collected;
- the purposes for which we use your information;
- to whom we disclose such information;
- the specific pieces of personal information we have collected about you.

You also have **the right to know what personal information is sold or shared and to whom**. In particular, you have the right to request two separate lists from us where we disclose:

- the categories of personal information that we sold or shared about you and the categories of third parties to whom the personal information was sold or shared;
- the categories of personal information that we disclosed about you for a business purpose and the categories of persons to whom it was disclosed for a business purpose.

The disclosure described above will be limited to the personal information collected or used over the past 12 months.

If we deliver our response electronically, the information enclosed will be "portable", i.e. delivered in an easily usable format to enable you to transmit the information to another entity without hindrance – provided that this is technically feasible.

The right to request the deletion of your personal information

You have the right to request that we delete any of your personal information, subject to exceptions set forth by the law (such as, including but not limited to, where the information is used to identify and repair errors on this Application, to detect security incidents and protect against fraudulent or illegal activities, to exercise certain rights etc.).

If no legal exception applies, as a result of exercising your right, we will delete your personal information and notify any of our service providers and all third parties to whom we have sold or shared the personal information to do so – provided that this is technically feasible and doesn't involve disproportionate effort.

The right to correct inaccurate personal information

You have the right to request that we correct any inaccurate personal information we maintain about you, taking into account the nature of the personal information and the purposes of the processing of the personal information.

The right to opt out of sale or sharing of personal information and to limit the use of your sensitive personal information

You have the right to opt out of the sale or sharing of your personal information. You also have the right to request that we limit our use or disclosure of your sensitive personal information.

The right of no retaliation following opt-out or exercise of other rights (the right to non-discrimination)

We will not discriminate against you for exercising your rights under the CCPA. This means that we will not discriminate against you, including, but not limited to, by denying goods or services, charging you a different price, or providing a different level or quality of goods or services just because you exercised your consumer privacy rights.

However, if you refuse to provide your personal information to us or ask us to delete or stop selling your personal information, and that personal information or sale is necessary for us to provide you with goods or services, we may not be able to complete that transaction.

To the extent permitted by the law, we may offer you promotions, discounts, and other deals in exchange for collecting, keeping, or selling your personal information, provided that the financial incentive offered is reasonably related to the value of your personal information.

How to exercise your rights

To exercise the rights described above, you need to submit your verifiable request to us by contacting us via the details provided in this document.

For us to respond to your request, it's necessary that we know who you are. Therefore, you can only exercise the above rights by making a verifiable request which must:

- provide sufficient information that allows us to reasonably verify you are the person about whom we collected personal information or an authorized representative;
- describe your request with sufficient detail that allows us to properly understand, evaluate, and respond to it.

We will not respond to any request if we are unable to verify your identity and therefore confirm the personal information in our possession actually relates to you.

Making a verifiable consumer request does not require you to create an account with us. We will use any personal information collected from you in connection with the verification of your request solely for the purposes of verification and shall not further disclose the personal information, retain it longer than necessary for purposes of verification, or use it for unrelated purposes.

If you cannot personally submit a verifiable request, you can authorize a person registered with the California Secretary of State to act on your behalf.

If you are an adult, you can make a verifiable request on behalf of a minor under your parental authority.

You can submit a maximum number of 2 requests over a period of 12 months.

How and when we are expected to handle your request

We will confirm receipt of your verifiable request within 10 days and provide information about how we will process your request.

We will respond to your request within 45 days of its receipt. Should we need more time, we will explain to you the reasons why, and how much more time we need. In this regard, please note that we may take up to 90 days to fulfill your request.

Our disclosure(s) will cover the preceding 12-month period. Only with regard to personal information collected on or after January 1, 2022, you have the right to request that we disclose information beyond the 12-month period, and we will provide them to you unless doing so proves impossible or would involve a disproportionate effort.

Should we deny your request, we will explain you the reasons behind our denial.

We do not charge a fee to process or respond to your verifiable request unless such request is manifestly unfounded or excessive. In such cases, we may charge a reasonable fee, or refuse to act on the request. In either case, we will communicate our choices and explain the reasons behind it.

Information for Virginia consumers

This part of the document integrates with and supplements the information contained in the rest of the privacy policy and is provided by the controller running this Application and, if the case may be, its parent, subsidiaries and affiliates (for the purposes of this section referred to collectively as "We", "Us", "Our").

The provisions contained in this section apply to all Users (Users are referred to below, simply as "You", "Your", "Yours"), who are consumers residing in the Commonwealth of Virginia, according to the "Virginia Consumer Data Protection Act" (the "VCDPA"), and, for such consumers, these provisions supersede any other possibly divergent or conflicting provisions contained in the privacy policy.

This part of the document uses the term "Personal data" as defined in the VCDPA.

Categories of personal data processed

In this section, we summarize the categories of personal data that we've processed and the purposes thereof. **You can read about these activities in detail in the section titled "Detailed information on the processing of Personal Data" within this document**

Categories of personal data we collect

We have collected the following categories of personal data: Identifiers and internet information

We do not collect sensitive data.

We will not collect additional categories of personal data without notifying you.

Why we process your personal data

To find out why we process your personal data, you can read the sections titled "Detailed information on the processing of Personal Data" and "The purposes of processing" within this document.

We won't process your information for unexpected purposes, or for purposes incompatible with the purposes originally disclosed, without your consent.

You can freely give, deny, or withdraw such consent at any time using the contact details provided in this document.

How we use the data we collect: sharing of your personal data with third parties

We share your personal data with the third parties listed in detail in the section titled "Detailed information on the processing of Personal Data" within this document. These third parties are grouped and categorized in accordance with the different purposes of processing.

For our purposes, the word "third party" means "a natural or legal person, public authority, agency, or body other than the consumer, controller, processor, or an affiliate of the processor or the controller" as defined by the VCDPA.

Sale of your personal data

For our purposes, the word "Sale" means any "exchange of personal data for monetary consideration by us to a third party" as defined by the VCDPA.

Please note that according to the VCDPA, the disclosure of personal data to a processor that processes personal data on behalf of a controller does not constitute a sale.

As specified in the "Detailed information on the processing of Personal Data" section of this document, our use of your personal information may be considered a sale under VCDPA.

Your right to opt out of the sale of your personal data and how you can exercise it

You have the right to opt out of the sale of your personal data. This means that whenever you request us to stop selling your data, we will abide by your request. To fully exercise your right to opt out you can contact us at any time using the contact details provided in this document.

We use any personal data collected from you in connection with the submission of your opt-out request solely for the purpose of complying with the request.

Processing of your personal data for targeted advertising

For our purposes, the word "targeted advertising" means "displaying advertisements to you where the advertisement is selected based on personal data obtained from your activities over time and across nonaffiliated websites or online applications to predict your preferences or interests" as defined by the VCDPA.

To find out more details on the processing of your personal data for targeted advertising purposes, you can read the section titled "Detailed information on the processing of Personal Data" within this document.

Your right to opt out of the processing of your personal data for targeted advertising and how you can exercise it

You have the right to opt out of the processing of your personal data for targeted advertising. This means that whenever you ask us to

stop processing your data for targeted advertising, we will abide by your request. To fully exercise your right to opt out you can contact us at any time, using the contact details provided in this document.

We use any personal data collected from you in connection with the submission of your opt-out request solely for the purposes of complying with the opt-out request.

Your privacy rights under the Virginia Consumer Data Protection Act and how to exercise them

You may exercise certain rights regarding your data processed by us. In particular, you have the right to do the following:

access personal data: the right to know. You have the right to request that we confirm whether or not we are processing your personal data. You also have the right to access such personal data.

correct inaccurate personal data. You have the right to request that we correct any inaccurate personal data we maintain about you, taking into account the nature of the personal data and the purposes of the processing of the personal data.

request the deletion of your personal data. You have the right to request that we delete any of your personal data.

obtain a copy of your personal data. We will provide your personal data in a portable and usable format that allows you to transfer data easily to another entity, provided that this is technically feasible.

opt out of the processing of your personal data for the purposes of targeted advertising, the sale of personal data, or profiling in furtherance of decisions that produce legal or similarly significant effects concerning you.

non-discrimination. We will not discriminate against you for exercising your rights under the VCDPA. This means that we will not, among other things, deny goods or services, charge you a different price, or provide a different level or quality of goods or services just because you exercised your consumer privacy rights. However, if you refuse to provide your personal data to us or ask us to delete or stop selling your personal data, and that personal data or sale is necessary for us to provide you with goods or services, we may not be able to complete that transaction. To the extent permitted by the law, we may offer a different price, rate, level, quality, or selection of goods or services to you, including offering goods or services for no fee, if you have exercised your right to opt out; or our offer is related to your voluntary participation in a bona fide loyalty, rewards, premium features, discounts, or club card program.

How to exercise your rights

To exercise the rights described above, you need to submit your request to us by contacting us via the contact details provided in this document.

For us to respond to your request, we need to know who you are.

We will not respond to any request if we are unable to verify your identity using commercially reasonable efforts and therefore confirm that the personal data in our possession actually relates to you. In such cases, we may request that you provide additional information which is reasonably necessary to authenticate you and your request.

Making a consumer request does not require you to create an account with us. However, we may require you to use your existing account. We will use any personal data collected from you in connection with your request solely for the purposes of authentication, without further disclosing the personal data, retaining it longer than necessary for purposes of authentication, or using it for unrelated purposes.

If you are an adult, you can make a request on behalf of a minor under your parental authority.

How and when we are expected to handle your request

We will respond to your request without undue delay, but in all cases and at the latest within 45 days of its receipt. Should we need more time, we will explain to you the reasons why, and how much more time we need. In this regard, please note that we may take up to 90 days to fulfill your request.

Should we deny your request, we will explain to you the reasons behind our denial without undue delay, but in all cases and at the latest within 45 days of receipt of the request. It is your right to appeal such decision by submitting a request to us via the details provided in this document. Within 60 days of receipt of the appeal, we will inform you in writing of any action taken or not taken in response to the appeal, including a written explanation of the reasons for the decisions. If the appeal is denied you may contact the Attorney General to submit a complaint.

We do not charge a fee to respond to your request, for up to two requests per year. If your request is manifestly unfounded, excessive or repetitive, we may charge a reasonable fee or refuse to act on the request. In either case, we will communicate our choices and explain the reasons behind them.

Information for Users residing in Brazil

This part of the document integrates with and supplements the information contained in the rest of the privacy policy and is provided by the entity running this Application and, if the case may be, its parent, subsidiaries and affiliates (for the purposes of this section referred to collectively as "OuviO, OusO, OOurO").

The provisions contained in this section apply to all Users who reside in Brazil, according to the "Lei Geral de Proteção de Dados" (Users are referred to below, simply as "OyouO, OyourO, OyoursoO"). For such Users, these provisions supersede any other possibly divergent or conflicting provisions contained in the privacy policy.

This part of the document uses the term "Opersonal informationO" as it is defined in the Lei Geral de Proteção de Dados ([LGPD](#)).

The grounds on which we process your personal information

We can process your personal information solely if we have a legal basis for such processing. Legal bases are as follows:

your consent to the relevant processing activities;

compliance with a legal or regulatory obligation that lies with us;

the carrying out of public policies provided in laws or regulations or based on contracts, agreements and similar legal instruments;

studies conducted by research entities, preferably carried out on anonymized personal information;

the carrying out of a contract and its preliminary procedures, in cases where you are a party to said contract;

the exercising of our rights in judicial, administrative or arbitration procedures;

protection or physical safety of yourself or a third party;

the protection of health in procedures carried out by health entities or professionals;

our legitimate interests, provided that your fundamental rights and liberties do not prevail over such interests; and

credit protection.

To find out more about the legal bases, you can contact us at any time using the contact details provided in this document.

Categories of personal information processed

To find out what categories of your personal information are processed, you can read the section titled "Detailed information on the processing of Personal Data" within this document.

Why we process your personal information

To find out why we process your personal information, you can read the sections titled "Detailed information on the processing of Personal Data" and "The purposes of processing" within this document.

Your Brazilian privacy rights, how to file a request and our response to your requests

Your Brazilian privacy rights

You have the right to:

- obtain confirmation of the existence of processing activities on your personal information;
- access to your personal information;
- have incomplete, inaccurate or outdated personal information rectified;
- obtain the anonymization, blocking or elimination of your unnecessary or excessive personal information, or of information that is not being processed in compliance with the LGPD;
- obtain information on the possibility to provide or deny your consent and the consequences thereof;
- obtain information about the third parties with whom we share your personal information;
- obtain, upon your express request, the portability of your personal information (except for anonymized information) to another service or product provider, provided that our commercial and industrial secrets are safeguarded;
- obtain the deletion of your personal information being processed if the processing was based upon your consent, unless one or more exceptions provided for in art. 16 of the LGPD apply;
- revoke your consent at any time;
- lodge a complaint related to your personal information with the ANPD (the National Data Protection Authority) or with consumer protection bodies;
- oppose a processing activity in cases where the processing is not carried out in compliance with the provisions of the law;
- request clear and adequate information regarding the criteria and procedures used for an automated decision; and
- request the review of decisions made solely on the basis of the automated processing of your personal information, which affect your interests. These include decisions to define your personal, professional, consumer and credit profile, or aspects of your personality.

You will never be discriminated against, or otherwise suffer any sort of detriment, if you exercise your rights.

How to file your request

You can file your express request to exercise your rights free from any charge, at any time, by using the contact details provided in this document, or via your legal representative.

How and when we will respond to your request

We will strive to promptly respond to your requests.

In any case, should it be impossible for us to do so, we'll make sure to communicate to you the factual or legal reasons that prevent us from immediately, or otherwise ever, complying with your requests. In cases where we are not processing your personal information, we will indicate to you the physical or legal person to whom you should address your requests, if we are in the position to do so.

In the event that you file an **access** or personal information **processing confirmation** request, please make sure that you specify whether you'd like your personal information to be delivered in electronic or printed form.

You will also need to let us know whether you want us to answer your request immediately, in which case we will answer in a simplified fashion, or if you need a complete disclosure instead.

In the latter case, we'll respond within 15 days from the time of your request, providing you with all the information on the origin of your personal information, confirmation on whether or not records exist, any criteria used for the processing and the purposes of the processing, while safeguarding our commercial and industrial secrets.

In the event that you file a **rectification, deletion, anonymization or personal information blocking** request, we will make sure to immediately communicate your request to other parties with whom we have shared your personal information in order to enable such third parties to also comply with your request except in cases where such communication is proven impossible or involves disproportionate effort on our side.

Transfer of personal information outside of Brazil permitted by the law

We are allowed to transfer your personal information outside of the Brazilian territory in the following cases:

- when the transfer is necessary for international legal cooperation between public intelligence, investigation and prosecution bodies, according to the legal means provided by the international law;
- when the transfer is necessary to protect your life or physical security or those of a third party;
- when the transfer is authorized by the ANPD;
- when the transfer results from a commitment undertaken in an international cooperation agreement;
- when the transfer is necessary for the execution of a public policy or legal attribution of public service;
- when the transfer is necessary for compliance with a legal or regulatory obligation, the carrying out of a contract or preliminary procedures related to a contract, or the regular exercise of rights in judicial, administrative or arbitration procedures.

Definitions and legal references

Personal Data (or Data)

Any information that directly, indirectly, or in connection with other information – including a personal identification number – allows for the identification or identifiability of a natural person.

Usage Data

Information collected automatically through this Application (or third-party services employed in this Application), which can include: the IP addresses or domain names of the computers utilized by the Users who use this Application, the URI addresses (Uniform Resource Identifier), the time of the request, the method utilized to submit the request to the server, the size of the file received in response, the numerical code indicating the status of the server's answer (successful outcome, error, etc.), the country of origin, the features of the browser and the operating system utilized by the User, the various time details per visit (e.g., the time spent on each page within the Application) and the details about the path followed within the Application with special reference to the sequence of pages visited, and other parameters about the device operating system and/or the User's IT environment.

User

The individual using this Application who, unless otherwise specified, coincides with the Data Subject.

Data Subject

The natural person to whom the Personal Data refers.

Data Processor (or Data Supervisor)

The natural or legal person, public authority, agency or other body which processes Personal Data on behalf of the Controller, as described in this privacy policy.

Data Controller (or Owner)

The natural or legal person, public authority, agency or other body which, alone or jointly with others, determines the purposes and means of the processing of Personal Data, including the security measures concerning the operation and use of this Application. The Data Controller, unless otherwise specified, is the Owner of this Application.

This Application

The means by which the Personal Data of the User is collected and processed.

Service

The service provided by this Application as described in the relative terms (if available) and on this site/application.

European Union (or EU)

Unless otherwise specified, all references made within this document to the European Union include all current member states to the European Union and the European Economic Area.

Cookie

Cookies are Trackers consisting of small sets of data stored in the User's browser.

Tracker

Tracker indicates any technology - e.g Cookies, unique identifiers, web beacons, embedded scripts, e-tags and fingerprinting - that enables the tracking of Users, for example by accessing or storing information on the User's device.

Legal information

This privacy statement has been prepared based on provisions of multiple legislations, including Art. 13/14 of Regulation (EU) 2016/679 (General Data Protection Regulation).

This privacy policy relates solely to this Application, if not stated otherwise within this document.

Latest update: December 20, 2022

iubenda hosts this content and only collects the Personal Data strictly necessary for it to be provided.

Cookie Policy of www.mountaingoatsoftware.com

This document informs Users about the technologies that help this Application to achieve the purposes described below. Such technologies allow the Owner to access and store information (for example by using a Cookie) or use resources (for example by running a script) on a User's device as they interact with this Application.

For simplicity, all such technologies are defined as "Trackers" within this document – unless there is a reason to differentiate. For example, while Cookies can be used on both web and mobile browsers, it would be inaccurate to talk about Cookies in the context of mobile apps as they are a browser-based Tracker. For this reason, within this document, the term Cookies is only used where it is specifically meant to indicate that particular type of Tracker.

Some of the purposes for which Trackers are used may also require the User's consent, depending on the applicable law. Whenever consent is given, it can be freely withdrawn at any time following the instructions provided in this document.

This Application uses Trackers managed directly by the Owner (so-called "first-party" Trackers) and Trackers that enable services provided by a third-party (so-called "third-party" Trackers). Unless otherwise specified within this document, third-party providers may access the Trackers managed by them.

The validity and expiration periods of Cookies and other similar Trackers may vary depending on the lifetime set by the Owner or the relevant provider. Some of them expire upon termination of the User's browsing session.

In addition to what's specified in the descriptions within each of the categories below, Users may find more precise and updated information regarding lifetime specification as well as any other relevant information – such as the presence of other Trackers - in the linked privacy policies of the respective third-party providers or by contacting the Owner.

To find more information dedicated to Californian consumers and their privacy rights, Users may read the privacy policy.

Activities strictly necessary for the operation of this Application and delivery of the Service

This Application uses so-called "technical" Cookies and other similar Trackers to carry out activities that are strictly necessary for the operation or delivery of the Service.

Third-party Trackers

Traffic optimization and distribution

This type of service allows this Application to distribute their content using servers located across different countries and to optimize their performance.

Which Personal Data are processed depends on the characteristics and the way these services are implemented. Their function is to filter communications between this Application and the User's browser.

Considering the widespread distribution of this system, it is difficult to determine the locations to which the contents that may contain Personal Information of the User are transferred.

Cloudflare (Cloudflare)

Cloudflare is a traffic optimization and distribution service provided by Cloudflare Inc.

The way Cloudflare is integrated means that it filters all the traffic through this Application, i.e., communication between this Application and the User's browser, while also allowing analytical data from this Application to be collected.

Personal Data processed: Cookies and various types of Data as specified in the privacy policy of the service.

Place of processing: US – Privacy Policy.

Other activities involving the use of Trackers

Experience enhancement

This Application uses Trackers to provide a personalized user experience by improving the quality of preference management options, and by enabling interaction with external networks and platforms.

Content commenting

Content commenting services allow Users to make and publish their comments on the contents of this Application.

Depending on the settings chosen by the Owner, Users may also leave anonymous comments. If there is an email address among the Personal Data provided by the User, it may be used to send notifications of comments on the same content. Users are responsible for the content of their own comments.

If a content commenting service provided by third parties is installed, it may still collect web traffic data for the pages where the comment service is installed, even when Users do not use the content commenting service.

Disqus (Disqus)

Disqus is a content commenting service provided by Big Heads Labs Inc.

Personal Data processed: Cookies, Usage Data and various types of Data as specified in the privacy policy of the service.

Place of processing: US | Privacy Policy | Opt out.

Displaying content from external platforms

This type of service allows you to view content hosted on external platforms directly from the pages of this Application and interact with them.

This type of service might still collect web traffic data for the pages where the service is installed, even when Users do not use it.

Wistia widget (Wistia, Inc.)

Wistia is a video content visualization service provided by Wistia, Inc. that allows this Application to incorporate content of this kind on its pages.

Personal Data processed: Tracker and Usage Data.

Place of processing: United States | Privacy Policy.

Interaction with external social networks and platforms

This type of service allows interaction with social networks or other external platforms directly from the pages of this Application. The interaction and information obtained through this Application are always subject to the User's privacy settings for each social network.

This type of service might still collect traffic data for the pages where the service is installed, even when Users do not use it. It is recommended to log out from the respective services in order to make sure that the processed data on this Application isn't being connected back to the User's profile.

Twitter Tweet button and social widgets (Twitter, Inc.)

The Twitter Tweet button and social widgets are services allowing interaction with the Twitter social network provided by Twitter, Inc.

Personal Data processed: Cookies and Usage Data.

Place of processing: US | Privacy Policy.

Storage duration:

personalization_id: 2 years

Facebook Like button and social widgets (Facebook, Inc.)

The Facebook Like button and social widgets are services allowing interaction with the Facebook social network provided by Facebook, Inc.

Personal Data processed: Cookies and Usage Data.

Place of processing: US | Privacy Policy.

Storage duration:

_fbp: 3 months

LinkedIn button and social widgets (LinkedIn Corporation)

The LinkedIn button and social widgets are services allowing interaction with the LinkedIn social network provided by LinkedIn Corporation.

Personal Data processed: Cookies and Usage Data.

Place of processing: US | Privacy Policy.

Storage duration:

AnalyticsSyncHistory: 1 month

JSESSIONID: duration of the session

UserMatchHistory: 1 month

bcookie: 1 year

bscookie: 1 year
lang: duration of the session
lido: 1 day
lssc: 1 year
lms_ads: 1 month
lms_analytics: 1 month

Measurement

This Application uses Trackers to measure traffic and analyze User behavior with the goal of improving the Service.

Analytics

The services contained in this section enable the Owner to monitor and analyze web traffic and can be used to keep track of User behavior.

Google Analytics (Google Inc.)

Google Analytics is a web analysis service provided by Google Inc. (‘Google’). Google utilizes the Data collected to track and examine the use of this Application, to prepare reports on its activities and share them with other Google services. Google may use the Data collected to contextualize and personalize the ads of its own advertising network.

Personal Data processed: Cookies and Usage Data.

Place of processing: US [D Privacy Policy](#) [D Opt Out](#).

Storage duration:

AMP_TOKEN: 1 hour
_ga: 2 years
_gac*: 3 months
_gat: 1 minute
_gid: 1 day

Amplitude Analytics (Amplitude Inc.)

Amplitude Analytics is an analytics service provided by Amplitude Inc.

Personal Data processed: Cookies and Usage Data.

Place of processing: United States [D Privacy Policy](#).

Storage duration:

amp_*: indefinite

Google Ads conversion tracking (Google LLC)

Google Ads conversion tracking is an analytics service provided by Google LLC that connects data from the Google Ads advertising network with actions performed on this Application.

Personal Data processed: Cookies and Usage Data.

Place of processing: United States [D Privacy Policy](#).

Storage duration:

IDF: 2 years
test_cookie: 15 minutes

Targeting & Advertising

This Application uses Trackers to deliver personalized marketing content based on User behavior and to operate, serve and track ads.

Remarketing and behavioral targeting

This type of service allows this Application and its partners to inform, optimize and serve advertising based on past use of this Application by the User.

This activity is facilitated by tracking Usage Data and by using Trackers to collect information which is then transferred to the partners that manage the remarketing and behavioral targeting activity.

Some services offer a remarketing option based on email address lists.

In addition to any opt-out feature provided by any of the services below, Users may opt out by visiting the Network Advertising Initiative opt-out page.

Users may also opt-out of certain advertising features through applicable device settings, such as the device advertising settings for mobile phones or ads settings in general.

Facebook Custom Audience (Facebook, Inc.)

Facebook Custom Audience is a remarketing and behavioral targeting service provided by Facebook, Inc. that connects the activity of this Application with the Facebook advertising network.

Personal Data processed: Cookies and email address.

Place of processing: US | Privacy Policy | Opt Out.

Storage duration:

_fbp: 3 months

Facebook Remarketing (Facebook, Inc.)

Facebook Remarketing is a remarketing and behavioral targeting service provided by Facebook, Inc. that connects the activity of this Application with the Facebook advertising network.

Personal Data processed: Cookies and Usage Data.

Place of processing: US | Privacy Policy | Opt Out.

Storage duration:

_fbp: 3 months

How to manage preferences and provide or withdraw consent

There are various ways to manage Tracker related preferences and to provide and withdraw consent, where relevant:

Users can manage preferences related to Trackers from directly within their own device settings, for example, by preventing the use or storage of Trackers.

Additionally, whenever the use of Trackers is based on consent, Users can provide or withdraw such consent by setting their preferences within the cookie notice or by updating such preferences accordingly via the relevant consent-preferences widget, if available.

It is also possible, via relevant browser or device features, to delete previously stored Trackers, including those used to remember the User's initial consent.

Other Trackers in the browser's local memory may be cleared by deleting the browsing history.

With regard to any third-party Trackers, Users can manage their preferences and withdraw their consent via the related opt-out link (where provided), by using the means indicated in the third party's privacy policy, or by contacting the third party.

Locating Tracker Settings

Users can, for example, find information about how to manage Cookies in the most commonly used browsers at the following addresses:

Google Chrome

Mozilla Firefox

Apple Safari

Microsoft Internet Explorer

Microsoft Edge

Brave

Opera

Users may also manage certain categories of Trackers used on mobile apps by opting out through relevant device settings such as the device advertising settings for mobile devices, or tracking settings in general (Users may open the device settings and look for the relevant setting).

How to opt out of interest-based advertising

Notwithstanding the above, Users may follow the instructions provided by YourOnlineChoices (EU), the Network Advertising Initiative (US) and the Digital Advertising Alliance (US), DAAC (Canada), DDAI (Japan) or other similar services. Such initiatives allow Users to select their tracking preferences for most of the advertising tools. The Owner thus recommends that Users make use of these resources in addition to the information provided in this document.

The Digital Advertising Alliance offers an application called AppChoices that helps Users to control interest-based advertising on mobile apps.

Consequences of denying consent

Users are free to decide whether or not to grant consent. However, please note that Trackers help this Application to provide a better experience and advanced functionalities to Users (in line with the purposes outlined in this document). Therefore, in the absence of the User's consent, the Owner may be unable to provide related features.

Owner and Data Controller

Mountain Goat Software
1140 US Hwy 287
Suite 400-108
Broomfield, CO 80020

Owner contact email: hello@mountaingoatsoftware.com

Since the use of third-party Trackers through this Application cannot be fully controlled by the Owner, any specific references to third-party Trackers are to be considered indicative. In order to obtain complete information, Users are kindly requested to consult the privacy policies of the respective third-party services listed in this document.

Given the objective complexity surrounding tracking technologies, Users are encouraged to contact the Owner should they wish to receive any further information on the use of such technologies by this Application.

Definitions and legal references

Personal Data (or Data)

Any information that directly, indirectly, or in connection with other information – including a personal identification number – allows for the identification or identifiability of a natural person.

Usage Data

Information collected automatically through this Application (or third-party services employed in this Application), which can include: the IP addresses or domain names of the computers utilized by the Users who use this Application, the URI addresses (Uniform Resource Identifier), the time of the request, the method utilized to submit the request to the server, the size of the file received in response, the numerical code indicating the status of the server's answer (successful outcome, error, etc.), the country of origin, the features of the browser and the operating system utilized by the User, the various time details per visit (e.g., the time spent on each page within the Application) and the details about the path followed within the Application with special reference to the sequence of pages visited, and other parameters about the device operating system and/or the User's IT environment.

User

The individual using this Application who, unless otherwise specified, coincides with the Data Subject.

Data Subject

The natural person to whom the Personal Data refers.

Data Processor (or Data Supervisor)

The natural or legal person, public authority, agency or other body which processes Personal Data on behalf of the Controller, as described in this privacy policy.

Data Controller (or Owner)

The natural or legal person, public authority, agency or other body which, alone or jointly with others, determines the purposes and means of the processing of Personal Data, including the security measures concerning the operation and use of this Application. The Data Controller, unless otherwise specified, is the Owner of this Application.

This Application

The means by which the Personal Data of the User is collected and processed.

Service

The service provided by this Application as described in the relative terms (if available) and on this site/application.

European Union (or EU)

Unless otherwise specified, all references made within this document to the European Union include all current member states to the European Union and the European Economic Area.

Cookie

Cookies are Trackers consisting of small sets of data stored in the User's browser.

Tracker

Tracker indicates any technology - e.g Cookies, unique identifiers, web beacons, embedded scripts, e-tags and fingerprinting - that enables the tracking of Users, for example by accessing or storing information on the User's device.

Legal information

This privacy statement has been prepared based on provisions of multiple legislations, including Art. 13/14 of Regulation (EU) 2016/679 (General Data Protection Regulation).

This privacy policy relates solely to this Application, if not stated otherwise within this document.

Latest update: December 20, 2022

iubenda hosts this content and only collects the Personal Data strictly necessary for it to be provided.

Learn About Agile **Agile & Scrum Training** **More...**

New to Agile and Scrum? Online Certifications About Us

Scrum Video Courses Contact Us

User Stories In-Person Certifications Our Students

Planning Poker On-Site Courses Blog

Agile Software Development Training Schedule Books by Mike Cohn

Development Compare Courses Agile Tools

Agile Project Management PDUs and SEUs

Agile Presentations

Mountain Goat on YouTube

Agile Mentors Podcast

Elements of Agile

Assessment

Commenting Policy

We welcome conversation on our blog posts and love that our community is passionate about agile and Scrum topics, and share their ideas, questions and comments regularly.

Here are some general guidelines to make the discussions great on our blog:

All comments must be submitted through the Disqus system, which means you must register to comment. We've tried many options for comment management, and this is the system that works best for us now.

All comments submitted are approved for publishing (except those caught in our spam filter). However, we reserve the right to edit or delete comments at any time based on our editorial assessment.

Comments should add value to the conversation, be obviously genuine and not spammy. If your comment is too shallow and there are other indicators that you are a spammer, the comment will not be approved.

You may include links within your comment if they add to the conversation and are not self-promotional. For example, if there is a discussion about agile tools and you run an agile tools company, don't link to your product within the comment unless doing so adds to the conversation in a meaningful way.

Keep comments relevant to the original post. Do not hijack a post to an entirely different question. Please see if your question has been addressed by other posts. If not, you can submit it at
<https://www.mountaingoatsoftware.com/ask>.

All links embedded within comments will have a nofollow attribute on them. So if you're looking for us to pass link equity to your site through your comment, you won't find that here.

Finally, please be civil. It's OK to share your unique perspective, and we will share ours. We don't have to agree to have a fruitful conversation, but being rude will get you a one-way ticket to being kicked out of the discussion.

Above all, we thank you for your participation and enjoy having you be a part of the discussion!

Commenting Policy

[User Stories](#)

[Planning Poker](#)

[Agile Software Development](#)

[Agile Project Management](#)

[Agile Presentations](#)

[Mountain Goat on YouTube](#)

[Agile Mentors Podcast](#)

[Elements of Agile](#)

[Assessment](#)

[In-Person Certifications](#)

[On-Site Courses](#)

[Training Schedule](#)

[Compare Courses](#)

[PDUs and SEUs](#)

[Our Students](#)

[Blog](#)

[Books by Mike Cohn](#)

[Agile Tools](#)

by **Tagged:**
Mike transitioning to agile, resistance, agile problems, expectation
Cohn management, core principles, optimize, agility, problems
Comments

My older daughter got married recently. About a month before the wedding, I realized I would dance with her and then with my wife at the reception afterwards. But uh-oh: I haven't danced in quite some time. I needed lessons.

But I didn't panic. How hard could it be? Patrick Swayze sure made it look easy. And I didn't think my daughter or wife would be expecting me to hold them overhead.

It turned out to be quite hard, even though it looked so easy. As I struggled to learn, I realized dancing was much like [Scrum](#): Easy to understand but hard to master.

Let's look at eight reasons why Scrum is hard, probably even harder than learning to dance.

Problem 1: All Change Is Hard

If the new way of doing something were easier, you'd probably already be doing it that way. There's usually a reason we've chosen to do something the way we have. A few months ago I noticed that I always whisk food in a clockwise motion. Just for fun I started whisking in the opposite direction. It's hard.

When a team adopts agile it affects nearly every aspect of how team members do their daily work. A

pervasive change like that is challenging. Even if team members are motivated to succeed at the change, there will be times they question whether it's worth the effort.

It's true: Something that is simple to understand can be difficult to master. To take some of the pain out of learning, give people a clear path to follow and a lot of clarity on the how and the why. My *why* for getting a dance refresher was that I didn't want to embarrass my wife and daughter at the wedding; and my how was lessons.

Problem 2: Sprint Reviews Are Scary at First

Showing your work to others and hearing their opinions about it can feel like a threat to your self-esteem. Will they like it? Will the system crash during the demo portion of the [sprint review](#)? Did we do enough during the sprint? These are intimidating questions.

Yet these questions, and others like them, pop into our heads when [Scrum teams](#) first begin giving demos during sprint reviews.

The good news is that after a while, reviews become second nature. As development teams see the benefit of receiving fast feedback, they can shift to eagerly anticipating reviews rather than fearing them.

Problem 3: Knowing What to Do with Feedback Is Difficult

Even after team members become comfortable showing their work every couple of weeks, knowing what to do with all that feedback can be challenging. With feedback coming fast and furious, teams need to decide which feedback to incorporate and which to ignore.

There's the timing of it, too: With a waterfall process, feedback is solicited at the end, after the team feels like the project is over. When feedback is provided more incrementally, every couple of weeks, teams can feel overwhelmed. They feel like they get further behind every time they ask for feedback.

One solution is to...prioritize your product goals. This will help you sift crucial feedback from details that needn't be dealt with right now. After all, not everything can be Priority A.

Problem 4: All This Collaboration Seems Slower

Before I became a programmer, I worked in a darkroom developing photos. I started each day by taking a bunch of undeveloped film into my darkroom. I didn't open the door until lunch, when I put the morning's photos in a bin. I got lunch, a new pile of film, and sequestered myself in the darkroom until quitting time.

I liked the isolation. And that continued as a programmer when I'd put on headphones, turn up the music, and code in isolation all day.

So I can relate to those on agile teams who wish they could just be given a big task and then go away and do it for a couple of weeks (or longer) with no need to communicate until the task is done.

But the products we build today require much more collaboration than they did when I began my career. And

sometimes collaborating with one's teammates does feel like it slows us down.

The key is realizing that all the talking, emailing, messaging, and meeting helps prevent problems. What you hear and say in a meeting will sometimes not be helpful. But very often, in a well-run meeting, what you hear does solve a problem, and what you say turns out to be helpful to someone else.

Problem 5: Story Points Are a New Way to Estimate

The idea of estimating in [story points](#) can definitely be a challenge for many team members. I can almost hear them thinking, "I have a hard estimating in days and now I have to estimate in an abstract relative unit I've never heard of before?"

Story points are a definite challenge, yet they're worth the effort. As abstract relative estimates of effort, story points enable better conversations about how long work will take.

Without story points, a senior programmer and junior programmer have conversations that devolve into, "That's how long it will take you, but it would take me twice as long." And then the two pick an estimate that is horrible for one of them or, perhaps even worse, they split the difference.

With story points, the senior and junior programmers can consider adding a new feature and both agree it will take twice as long as doing a simpler feature. They then give the bigger item an estimate twice that of the simpler item.

Estimating in this relative manner allows developers to agree even if they would never be able to agree on how many hours or days something would take. Bonus: story points discourage managers from comparing team velocity.

Problem 6: People Complain There Are too Many Meetings

A common complaint about Scrum is that there are too many meetings. It's a fair criticism, but I don't think it's justified, because each can be quite efficient. Let's consider the recommended duration of [the meetings of a two-week sprint](#), as summarized in the table.

MEETING	TIME PER TWO-WEEK SPRINT (HOURS)
Daily Scrum	1.5
Sprint Planning	2
Review	2
Retrospective	1
Backlog Refinement	1.5
Total	8

That's about eight hours. So is eight hours of meeting time in a two-week sprint too much? First, keep in mind these numbers are conservative. Seven hours may be a more realistic total. Eight hours is a lot of time, but I don't think it's excessive.

Meetings in Scrum are like the lines on the highway. The lines are there to help drivers proceed quickly and safely. Scrum's meetings should feel the same way. They keep team members safe by ensuring they each know what the other is doing.

The meetings should help the team move *more quickly* by creating opportunities for communication. If your team's meetings do not feel like the white lines on the highway, consider shortening or eliminating a meeting.

Problem 7: It Isn't Easy Being a Scrum Master

You are right. Scrum isn't easy. And it isn't easy being a [Scrum Master](#) either. But most jobs aren't easy when you're new to them. Stick with it long enough and it does get easier.

And by this point, there are plenty of books, courses, online forums, and more about being a Scrum Master. You're never too far from advice.

Problem 8: Being a Product Owner Is a Tough Job

If you thought being a Scrum Master was tough, try being the [product owner](#). Product owners get it from both sides: Teams ask for clarity and more details, customers and users ask for features. Being a product owner requires balancing these competing demands for their time.

Scrum Is Hard But It's Worth It

Adopting a Scrum approach is hard. There will likely be times you want to throw your hands up and go back to what you were doing before.

Usually those thoughts are quite fleeting. Stick with it, though—I'm sure you're already far better at it than I am at dancing. And I haven't given up yet. I'll stick with it if you will.

What Do You Think?

What aspects of Scrum have you found hard? Were there times when you were tempted to abandon it? Did you stick with it? Was it worth it? Please share your thoughts in the comments below.

[Take the Assessment](#)

Posted: October 11, 2022

Tagged: transitioning to agile, resistance, agile problems, expectation management, core principles, optimize, agility, problems

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

Agile Mentors Podcast: Recap of Opening Series on Scrum

Recapping our agile podcast's initial launch series of topics

Sep 27, 2022

[Read](#)

Elements of Agile: An Agile Assessment Tool for Iterative Improvements

New to agile, or needing an agile re-boot? We've got a new no-cost assessment and actionable ...

Sep 13, 2022

[Read](#)

From Project Manager to Scrum Master - 3 Tips for Making the Transition

What does a good project manager need to do to become a great Scrum Master?

Aug 23, 2022

[Read](#)

For Better Agile Planning, Be Collaborative

The best plans are created by developers and stakeholders working together.

Jul 12, 2022

[Read](#)

3 Ways to Help Agile Teams Plan Despite Uncertainty

We might not like ambiguity, but it's a fact of life. Find out how to plan with uncertainty in mind.

Jun 21, 2022

[Read](#)

What Happens When You Use Agile for the Wrong Reasons?

Knowing why you want to achieve will a goal will help you reach it.

Jan 05, 2021

[Read](#)

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

When Should We Estimate the Product Backlog

by
Mike **Tagged:**
Cohn product backlog, estimating, user stories, product owner, story
59 points, sprint planning, meetings, daily scrum, scrum master
[Comments](#)

I was recently emailed a question asking whether the sprint planning meeting should start with time allocated for putting story point estimates on any user stories that have not yet been estimated. No, I don't think this is a good idea. Keep in mind that we put estimates on product backlog items (which I recommend be user stories) so that:

the product backlog can be prioritized. It is impossible to fully prioritize a set of items without knowing at least their relative cost.

we can make high-level forecasts about how much will be done by when
we can make tradeoff decisions between scope and schedule

We can achieve these goals with approximate, relative estimates such as given by story points. For example, if I decide to buy a new car this weekend it is sufficient for me to know that I can get a Toyota or Honda for around \$30,000 and that a Ferrari goes for closer to \$800,000. I do not need to know a more precise cost of the Honda (\$31,850) before knowing it should be on my short list of cars to evaluate while the Ferrari should not. Sprint planning meetings typically go into deeper detail than is appropriate for product backlog item estimating (whether in story points or ideal days). Since we become accustomed during sprint planning to breaking user stories into tasks and considering those tasks in more detail, there is a chance this will carry over into any story point estimating done during the same meeting. So, when should story point estimating happen? I'll describe the ideal case, which you can easily adjust for the real-world intrusions on your project.

Projects typically start in either of two ways:

1. A reasonably fully stocked product backlog is written before the first sprint begins and all items are estimated before the first sprint planning meeting. [If you do this, be careful not to write all user stories with too much detail. Each product backlog item you write represents an investment. User stories should therefore be elaborated just-in-time and in just-enough detail that they can be turned into functionality in one sprint.]
2. We know we've got to do the project so we dive in. During the first sprint or two, all the user stories are written and estimated just like above.

On an ongoing basis, once per sprint I recommend that the ScrumMaster tell the team something like, "Hey, we've had five new user stories come in this sprint and we need to estimate them. Everyone plan on hanging around for a bit after tomorrow's daily scrum meeting and we'll play Planning Poker to estimate the new items." Doing it right after the daily scrum helps cut down on the number of interruptions in total. I usually aim for having that meeting about two days before the end of the sprint. That way the product owner will have estimates on them so she can prioritize prior to the start of the sprint.

Get 200 Real Life User Stories Examples Written by Mike Cohn

See user stories Mike Cohn wrote as part of several real product backlogs.

First Name

Email Address

[Privacy Policy](#)

Posted: March 16, 2008

Tagged: product backlog, estimating, user stories, product owner, story points, sprint planning, meetings, daily scrum, scrum master

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of User Stories Applied for Agile Software Development, Agile Estimating and Planning, and Succeeding with Agile as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

Why I Don't Emphasize Sprint Goals Sprint goals are considered a mandatory part of Scrum. Here's why I disagree.	User Stories: How to Create Story Maps Story maps help to create a shared understanding of the product, visualize user needs, and elicit ...	How to Run a Successful User Story Writing Workshop What you need to know to conduct a story-writing workshop with your team.	Be a Great Product Owner: Six Things Teams and Scrum Masters Need Learn six ways effective product owners ensure their teams' success.	Six Attributes of a Great ScrumMaster Scrum Masters exhibit a shared set of traits. Discover 6 in-demand qualities, plus a bonus ...	What Happens When During a Sprint Succeeding with Scrum is easier when you know when and why to conduct each of the Scrum events ...		
Mar 21, 2023	Read	Mar 14, 2023	Read	Feb 28, 2023 • 3 Comments Read	Jan 31, 2023 • 16 Comments Read	Jan 17, 2023 • 32 Comments Read	Jan 03, 2023 • 34 Comments Read

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations		
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile Assessment	PDUs and SEUs	

Presentation by: Mike Cohn **Tagged:** estimating, teams, sprinting, story points, sprint planning, management, planning poker, agile software development, release planning, presentations, video recorded

Maybe you've heard about agile software development projects but aren't sure if they allow for the detailed planning and estimation your business requires. You might also worry if your team provides the estimates that management wants, the numbers might come back to haunt you. Whether you are a manager, programmer, tester, or have another role that involves project planning, what you need to know is how agile estimating works and whether the plan's result will be something your team and your business can trust.

In this group of presentations, Certified Scrum Trainer Mike Cohn explains how to create useful estimates that teams are comfortable with and management can rely on for decision-making. Explore story points, ideal days, and how to estimate with [Planning Poker](#). Leave with new insight into both short-term iteration and long-term release planning.

[View the Presentation](#)[Download a PDF](#)[Watch the Video](#)

Watch the video to get 1 PDU credit for:

[Learn more about PDUs](#)

The PMI Registered Education Provider logo is a registered mark of the Project Management Institute, Inc.

[Get my chapters now!](#)

You may also be interested in:

[Why I Don't Emphasize Sprint Goals](#)

Sprint goals are considered a mandatory part of Scrum. Here's why I disagree.

Mar 21, 2023

[Read](#)

[What Is Cross-Functional Collaboration in Agile?](#)

Perhaps the most prevalent and persistent myth in agile is that a cross-functional team is one on ...

49 Comments

Feb 14, 2023

[Read](#)

[What Happens When During a Sprint](#)

Succeeding with Scrum is easier when you know when and why to conduct each of the Scrum events ...

Jan 03, 2023

34 Comments

[Read](#)

[What Are Agile Story Points?](#)

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022

122 Comments

[Read](#)

Don't Equate Story Points to Hours

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022

• 39 Comments

[Read](#)

For Better Agile Planning, Be Collaborative

The best plans are created by developers and stakeholders working together.

Jul 12, 2022

[Read](#)

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile Assessment		

by
Mike **Tagged:**
Cohn product backlog, estimating, story points, agile project
217 management
[Comments](#)

A client asked me, "When will my team be done with this project?" This is probably the bazillionth time I've been asked that agile project management question in one way or another.

I have never once been asked, "How hard will my team have to think to develop this project?"

Clients, bosses, customers and stakeholders care about how long a project will take. They don't care about how hard we have to think to deliver the project, except to the extent that the need to think hard implies schedule or cost risk.

Teams Think Story Points Are Just Complexity

I mention this because I find too many teams who think that story points should be based on the complexity of the user story or feature rather than the effort to develop it.

Such teams often re-label "story points" as "complexity points." I guess that sounds better. More sophisticated, perhaps.

But it's wrong.

Complexity is a factor in the number of points a product backlog item should be given. But it is not the only factor. The amount of work to be done is a factor. So, too, are risk and uncertainty.

Taken together these represent the *effort* involved to develop the product backlog item.

An Example of Why Points Can't Be Just Complexity

In a Scrum Master course a few years back, I was given a wonderful example of why story points cannot be just complexity. Let me share it with you now.

Suppose a team consists of a little kid and a brain surgeon. Their product backlog includes two items: lick 1,000 stamps and perform a simple brain surgery -- snip and done.

These items are chosen to presumably take the same amount of time. If you disagree, simply adjust the number of stamps in the example.

Despite their vastly different complexities, the two items should be given the same number of story points because each is expected to take the same amount of time. In this carefully chosen example, the volume of work (licking 1,000 stamps and one snip to some part of the brain) and the complexity of that work combine such that each will take the same amount of time.

So complexity is a factor in the number of story points assigned, but only to the extent to which that complexity increases the expected effort.

Adding Risk and Uncertainty

Remember I said it was a simple brain surgery. (I'm not even sure that exists, but go with me for another minute.)

If there was a risk the surgeon began the surgery and discovered he also needed to do some extra work during the surgery, that would be factored in and the estimate increased. So if we know a surgery will be simple we might call it 5 points. But if there's a risk the surgery uncovers other work to be done, we would increase the estimate to a higher value.

The Right Person for the Job Should Do The Job

Our example of the surgeon and the little kid points points out another aspect of agile estimating. We should assume that the right person for the job will do the work.

We do not assume the little kid will finish school, go to med school, do a seven-year residency and only then begin the brain surgery while we have a skilled surgeon sitting in a cubicle licking stamps.

Of course reality intrudes and occasionally the "wrong" person for a job does the job, but that will rarely be as dramatic as in this example.

Story Points Are Effort

So, story points are an estimate of the effort involved in doing something. That estimate should be based on a number of factors, including the volume of work, the risk or uncertainty inherent in the work, and the complexity of the work.

But story points are not solely about complexity.

[Download my PDF](#)

Posted: June 21, 2010

Tagged: product backlog, estimating, story points, agile project management

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

[What Are Agile Story Points?](#)

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments

[Read](#)

[Don't Equate Story Points to Hours](#)

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • [39 Comments](#)

[Read](#)

[Epics, Features and User Stories](#)

I've been getting more and more emails lately from people confused about the difference between ...

Nov 08, 2022 • [93 Comments](#)

[Read](#)

For Better Agile Planning, Be Collaborative

The best plans are created by developers and stakeholders working together.

Jul 12, 2022

[Read](#)

Relationship between Definition of Done and Conditions of Satisfaction

I'd like to clarify the relationship between two important concepts: a team's Definition of Done ...

Jun 28, 2022 • [38 Comments](#)

[Read](#)

Four Reasons Agile Teams Estimate Product Backlog Items

Estimating product backlog items provides benefits beyond predicting when a project will be finished.

May 17, 2022

[Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

by

Mike **Tagged:**

Cohn product backlog, estimating, sprinting, meetings, velocity, planning

42 poker, scrum master

Comments

I was recently interviewed for an upcoming agile textbook written by Sondra Ashmore and Kristin Runyan. They asked me questions regarding several areas of agile development and Scrum. Last week, I [explored questions they had](#) about the product backlog. This week, I'd like to tell you about the conversation we had about estimating.

The conversation began as follows: Estimating is really hard. How can we get the best estimates of story size?

Well, estimating is definitely hard. But there are a few things teams can do to help this process:

1. Keep estimates manageable.
2. Estimate in relative terms.
3. Bucket backlog items by story size.

1. Keep Estimates Manageable

Try to keep most estimates, or at least *the most important* estimates within about one [order of magnitude](#), such as from 1-10. There are studies that have shown humans are pretty good across one order of magnitude, but beyond that, we are pretty bad.

That's why in the agile estimating method of Planning Poker, most of the cards are between 1-13. We can estimate pretty well in that range. You can get acquainted with the Planning Poker method to help [make estimating easier here](#).

2. Estimate in Relative Terms

Estimate in relative terms rather than absolute terms. Instead of saying, "This will take five days," say, "This thing will take about as long as that thing."

In my Certified ScrumMaster training class, we do an exercise where we estimate home improvement tasks. We start by agreeing on something small, but not the smallest, and perhaps calling that a 2.

For example we might decide that installing a new oven is a 2. All other items on the list can be estimated relative to this. Something that will take four times as long would get an 8, and so on.

Not only is relative estimating more accurate, teams can do it more quickly. With relative estimating, the team does not need to think of all of the sub-steps and estimate each and then add them up. They only need to find something similar and use the same estimate.

The next step is to use velocity to determine how long each of those things actually takes. Velocity is how fast the team is getting those product backlog items done. Since velocity is volatile, I like to look at data from at least five sprints to get a range.

3. Bucket Backlog Items by Story Size

It's hard to estimate and extremely hard to estimate precisely. It's good then, to make estimating easier by not requiring teams to put exact values on every item they estimate.

I like to do this by pre-identifying a set of values teams will estimate with. A team might choose to estimate using 1, 2, 4, 8 and 16. Another team might use the Fibonacci sequence of 1, 2, 3, 5, 8 and 13, which is my slight preference.

The numbers in these sequences can be viewed as buckets. Using the latter sequence we have a 5-point bucket and an 8-point bucket.

The team doesn't have to precisely estimate how long a given product backlog item will take. Instead, the team has only to put the product backlog item into the right bucket: "Does this item belong in the 5-point bucket or the 8-point bucket?" (Or perhaps some other bucket.)

This saves time in estimating meetings by removing the pressure to be perfect. By removing pressure, it also helps get team members to engage in the process.

How do you make estimating easier within your agile team? I welcome your thoughts below.

[Get my chapters now!](#)

Posted: July 10, 2013

Tagged: product backlog, estimating, sprinting, meetings, velocity, planning poker, scrum master

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

Six Attributes of a Great ScrumMaster

Scrum Masters exhibit a shared set of traits. Discover 6 in-demand qualities, plus a bonus ...

Jan 17, 2023 • 32 Comments [Read](#)

What Happens When During a Sprint

Succeeding with Scrum is easier when you know when and why to conduct each of the Scrum events ...

Jan 03, 2023 • 34 Comments [Read](#)

What Are Agile Story Points?

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

Don't Equate Story Points to Hours

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • 39 Comments [Read](#)

Epics, Features and User Stories

I've been getting more and more emails lately from people confused about the difference between ...

Nov 08, 2022 • 93 Comments [Read](#)

Agile Mentors Podcast: Recap of Opening Series on Scrum

Recapping our agile podcast's initial launch series of topics

Sep 27, 2022 [Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

by
Mike Tagged:
Cohn product backlog, estimating, product owner, teams, story points,
31 planning poker, scrum master
Comments

When estimating your product backlog with [Planning Poker](#), it's important to have the right people participate. Let's go over who needs to be there, and what the job of each is during an agile estimating meeting.

First, of course, we start with the team. All of a Scrum team's members should be present when playing Planning Poker. You may be tempted to estimate without the individuals in some role—don't.

An estimate put on a product backlog item (most frequently a user story) needs to represent the total effort to deliver that item. It's not an estimate of all the work except database changes or all the work except testing.

Those individuals need to be there because they're needed to deliver many of the product backlog items. They're also there because they will often have something to contribute to discussions of work they may not directly be involved in.

I've seen many examples when a comment by a tester about his or her own work made a programmer realize he overlooked something in his own work.

Programmers are, of course, the best at estimating programming work. And testers are best at estimating testing, and so on. But that doesn't mean someone with a specialty in one skill won't have something to add in a discussion about something that will be done by someone else.

Also in a Planning Poker meeting will be the ScrumMaster. A team's ScrumMaster is its facilitator, and so the ScrumMaster should participate in all regularly scheduled meetings, and a Planning Poker session is no exception.

But should the ScrumMaster estimate alongside the rest of the team? If the ScrumMaster is also a technical contributor such as a programmer, tester, designer, analyst, DBA, etc., then definitely.

By right of contributing in that way, the ScrumMaster holds up a Planning Poker card just like the others.

However, a pure ScrumMaster should estimate only by invitation of the rest of the team. If the others say, "Hey, we have to estimate, why don't you?" then the ScrumMaster holds up a card each round.

A ScrumMaster who has a solid technical background (in any role) will be especially likely to estimate along with the others.

That brings us to the product owner—and, yes, by all means the product owner needs to be present during any Planning Poker estimating session. The product owner describes each item to the team before they estimate it.

Also, team members will have questions—"what should we do if ... ?" "What if the user does ... ?"—and the product owner needs to be present to answer those questions.

Product owners, however, generally do not hold up cards during Planning Poker sessions. However, my official answer is the same as for the ScrumMaster: if the team invites the product owner to estimate, the product owner should do so. This is pretty rare, though, and it's far more common for a ScrumMaster to be invited to estimate.

So, like many things in Scrum, Planning Poker is a whole-team activity.

[Take the Assessment](#)

Posted: March 25, 2014

Tagged: product backlog, estimating, product owner, teams, story points, planning poker, scrum master

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile Assessment		

by **Tagged:**
Mike estimating, collaboration, planning, estimates, agile planning,
Cohn expectation management
Comments

I'm cooking something new for dinner tonight. I came across [a recipe for sajiyeh](#) that looks tasty. So I'm giving it a try. The recipe says it will take 40 minutes. That seems reasonable. And in my experience, most recipe estimates are pretty good. I usually take a little longer than they say, but I attribute that to my slowness rather than an error in the recipe.

I find it useful when a recipe includes an estimate of how long it will take. It gives me valuable information about how difficult the recipe is likely to be (and how many dishes I have to wash when I'm done).

I don't find it useful, however, when a boss or client tells my agile team how long something will take. In fact, when [product owners or project managers](#) tell me how long something should take or they provide a deadline, my first instinct is often to reject their estimate, even if the estimate is higher than my own would have been.

The Problem with One-Way Plans

One-way planning, whether it comes from the top-down or the bottom-up, is not ideal. In fact, it works against an organization becoming agile. Bosses, product owners, and clients should not tell a team when something will be done. Similarly, though, a team should not dictate dates without consideration for what the business or client needs.

For an organization to be agile, collaborative planning must be the norm. Creation of the plan may be guided by either the development group or the business stakeholders. But the plan should not be called done until the other side's input has been considered, often resulting in changes to the plan.

Team-Lead Collaborative Planning

A team may create a high-level release plan describing what will be delivered and by when, based on its estimates of the effort required. But that plan may not suffice to meet the organization's needs. The business might have very real deadlines. Sometimes these deadlines are so critical that the project itself makes no sense if it cannot be delivered on time.

When project *plans* and project *needs* conflict, the developers and business stakeholders should review the plans together and negotiate a better solution.

This does not mean stakeholders can reject a plan and force the developers to deliver more, deliver sooner, or both. It means that both parties seek a better alternative than the one in the initial plan. That may mean

- a later date with more features
- an earlier date with fewer features
- additional team members
- relaxing a particular requirement that had an outsized impact on the schedule

These same options should be considered when a team tells stakeholders that what they've asked for is impossible.

3 Ways to Ensure Collaboration

Collaborative planning exists when the organization exhibits three characteristics.

First, **plans are based on data and experience rather than hope**. When data shows that a team's historical velocity has been within, let's say, 20–30 points per iteration, stakeholders cannot insist that a plan be based on a velocity of 40. Everyone involved, including developers, may hope for 40, but the plan needs to be based on facts.

Second, **stakeholders need to be comfortable with plans that are occasionally expressed as ranges**. Just as in the discussion of velocity above, the [most accurate agile estimations use ranges](#). A team may, for example, promise to deliver by a prescribed date but will retain flexibility in how much they promise to deliver by then.

A third characteristic of organizations successfully engaging in collaborative planning is that **plans are updated as more is learned**. Maybe an initial estimate of velocity has turned out wrong. Or perhaps a new team member was added (or removed). Maybe the team learns that certain types of work were over- or under-estimated.

In each of these cases, recognize that the plan is based on outdated, bad information until it is updated to reflect new information.

Things to Try

If collaborative planning is not the norm in your organization, there are some first steps that will improve things. First, make sure that no plan is ever shared before both the team and its stakeholders agree. Both sides of the development equation need to understand the importance of creating plans together.

You should also establish a precedent that plans will be based on agile estimates, meaning estimates [provided by those who will do the work](#). No one likes to be told how long it will take to do something—except perhaps in the case of trying a new recipe.

Additionally, speak with stakeholders about the importance of plans being [accurate, even at the expense of precision](#). It seems human nature to favor precision. I recently scheduled a doctor appointment for 1:25 P.M. My doctor has apparently decided his appointments should all be 25 minutes long, yet he's never once been on time for an appointment.

Similarly, my \$29 scale is precise to the tenth of a pound, yet it often differs by half a pound if I weigh myself twice.

Agile teams and their stakeholders also instinctively love precision. Statements like “in seven sprints we will deliver 161 story points” sounds gloriously precise. A team that can so precisely know how much it will deliver must be well informed and highly attuned to its capabilities.

Or team members multiplied a velocity of 23 by 7 sprints, got 161, and shared that as their plan. Precise, yes. But very likely precisely wrong. What if the team delivers only 160 points in seven sprints? Do stakeholders in that case have the right to be disappointed by the missing one point? Perhaps they do, since the team conveyed 161 as a certainty.

Everyone, stakeholders and team members alike, would have been far better served if the team had conveyed its estimate as a range. A more accurate plan might have stated that the team would deliver between 140 and 180.

Collaborative planning combines the wisdom of those who will do the work with stakeholders' knowledge of where the project has wiggle room. Plans created collaboratively are more likely to be embraced by everyone. And a shared interest in the accuracy and feasibility of the plan means it is far more likely to be achieved.

What Do You Think?

Are plans created collaboratively in your organization? Or is one group allowed to dictate dates and functionality? Has that created any problems? Please share your thoughts in the Comments below.

[Get my chapters now!](#)

Posted: July 12, 2022

Tagged: estimating, collaboration, planning, estimates, agile planning, expectation management

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of User Stories Applied for Agile Software Development, Agile Estimating and Planning, and Succeeding with Agile as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

[What Are Agile Story Points?](#)

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

[Don't Equate Story Points to Hours](#)

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • 39 Comments [Read](#)

[8 Reasons Scrum Is Hard to Learn \(but Worth It\)](#)

Transitioning to Scrum is worth it, but some aspects are challenging.

Oct 11, 2022 [Read](#)

[3 Ways to Help Agile Teams Plan Despite Uncertainty](#)

We might not like ambiguity, but it's a fact of life. Find out how to plan with uncertainty in mind.

Jun 21, 2022 [Read](#)

[Scrum, Remote Teams, & Success: Five Ways to Have All Three](#)

In a remote-first world, how does an agile team thrive working in a virtual, distributed way?

Jun 07, 2022 [Read](#)

[Four Reasons Agile Teams Estimate Product Backlog Item](#)

Estimating product backlog items provides benefits beyond predicting when a project will be finished.

May 17, 2022 [Read](#)

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

by

Mike **Tagged:**

Cohn product backlog, estimating, backlog, planning, agile practices,

9 trust

Comments

When talking about estimating with [story points](#), I'm often asked: Why should a team estimate at all?

Specifically, why should a team estimate its [product backlog](#) items?

I can think of four good reasons to estimate: credibility, deeper thinking, prioritization, and insight.

Estimates Create Credibility with Stakeholders

The most compelling reason is that good estimates can help a team to establish credibility with [stakeholders](#).

To see why this is important, let's consider a scenario that might be familiar to you.

Someone in your organization needs a new project delivered in, let's say, four months. This could be an entirely new product or an enhancement to an existing one.

Your team estimates the effort required to deliver the work, using [planning poker](#), the [fibonacci sequence](#), or some other methodology. And team members conclude that four months is impossible. Six months seems possible but eight seems much more likely. The team goes back to the stakeholders and tells them it can't be done in four months. Instead, they explain, it will take six to eight months.

In response, the stakeholders tell the team to do it anyway, and that they expect it to be delivered in four

months.

The stakeholders are essentially ignoring the team's estimates and plan. They're doing this because the team has probably demonstrated, over and over again, that they aren't very good at estimating. The team's track record is probably one late project after another.

And with a track record like that, the team is not viewed as an equal partner in negotiating dates. Since the team has demonstrated that they don't really know what can be delivered in a given amount of time, stakeholders don't put credence in its estimates.

Now imagine a different situation. The team has gotten better at estimating—not perfect, but better. They said one project would take three to four months and it did.

Another time, they said a project would take four to five months, and they were only two weeks late. They finished an iteration early on a five-to-six-month project. And the team finished yet another project on schedule, despite unexpected new features being added during the project.

This team has established a track record of being good at [agile estimating](#). When this team says the project will take six to eight months, stakeholders listen. They might be disappointed. After all, they wanted it in four months. But stakeholders in this situation with this team are far less likely to steamroll the team and tell them to just go build it in four months anyway.

This team deserves to be treated as equal partners in the project of figuring out what to do when more is wanted than there is time to build. The only way to become this team is to get good at agile estimation and forming plans from estimates.

I think this is a pretty compelling reason to estimate. But let me lay out three more reasons why teams working on agile projects should estimate their product backlogs.

Estimates Ensure Teams Stop and Think

A second reason is that the act of estimating will make the team smarter about the work they estimate. I think it's just human nature to think more carefully about our work if we're giving an estimate to someone.

When I provide that estimate for a [user story](#), I want to be right, or at least close. Accountability makes me think more deeply and thoroughly about the work, especially the work of a cross-functional team. Apart from yielding a good estimate, this thought process eliminates a lot of the big surprises that really blow a schedule.

Our third and fourth reasons won't apply to every team, but if they do apply to your Scrum team, they'll be important reasons for your team to estimate.

Estimates Help Product Owners Prioritize

The third reason teams estimate their product backlogs is to help their [Scrum product owners](#) prioritize. The estimate assigned to a product backlog item during product backlog refinement will influence how the [product owner prioritizes](#) the item.

If an item is estimated at 5 points, the product owner may want the team to do it next iteration.

If it's estimated at 50 points, the product owner will likely put it lower in the product backlog because there are probably a few other items that are more valuable considering that this item costs 50 points.

And if the item is estimated at 500 points, the product owner will likely put it near the bottom of the product backlog—or perhaps just throw it away if it's going to take that long to develop.

Estimates Provide Insight into Delivery

Finally, our fourth reason to estimate is to enable us to answer questions about when and how much can be delivered.

Many—perhaps most—teams are asked questions such as

When can you deliver all of these features?

How much of this can you deliver in three months?

When the product backlog has been estimated, the team can answer these questions.

If the product backlog has not been estimated, the team needs to fall back on traditional task decomposition at the level they'd typically do at [sprint planning](#) to create the [sprint backlog](#). (In the Scrum framework it's common to have [two different levels of planning](#).) They'll have to examine each product backlog item, decompose it into tasks, estimate each task, try to discover what they missed, add some amount of fudge factor, and use all that to try to answer those stakeholder questions.

Breaking each product backlog item into tasks and estimating all those tasks is much, much more work than directly estimating each product backlog item with story points. And we can actually estimate more accurately at that higher level because it's not reliant on completely identifying all the sub-tasks. Tasks change as work progresses, after all.

Credibility Is Key for Agile Team Success

Of these four reasons to estimate the product backlog, I find the first to be the most compelling. Until your team establishes a track record of providing good estimates and plans, the team won't be treated as an equal partner in negotiating deadlines. The consequence is you'll have little or no ability to push back against unrealistic deadlines that are imposed on the team.

What Do You Think?

Which of these four benefits to estimating helps your team the most? What objections to estimating has your team given? Were you able to persuade them to estimate? How? Please share your thoughts in the comments section below.

[Get my chapters now!](#)

Posted: May 17, 2022

Tagged: product backlog, estimating, backlog, planning, agile practices, trust

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of User Stories Applied for Agile Software Development, Agile Estimating and Planning, and Succeeding with Agile as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

What Are Agile Story Points?

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

Don't Equate Story Points to Hours

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • 39 Comments [Read](#)

Epics, Features and User Stories

I've been getting more and more emails lately from people confused about the difference between ...

Nov 08, 2022 • 93 Comments [Read](#)

For Better Agile Planning, Be Collaborative

The best plans are created by developers and stakeholders working together.

Jul 12, 2022 [Read](#)

Relationship between Definition of Done and Conditions of Satisfaction

I'd like to clarify the relationship between two important concepts: a team's Definition of Done ...

Jun 28, 2022 • 38 Comments [Read](#)

3 Ways to Help Agile Teams Plan Despite Uncertainty

We might not like ambiguity, but it's a fact of life. Find out how to plan with uncertainty in mind.

Jun 21, 2022 [Read](#)

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

by
Mike Tagged:
Cohn estimating, planning poker, agile planning, planning meeting
Comments

Most agile teams estimate relatively. This means they estimate items in comparison to other items. For example, I don't know how long it will take to develop this user story, but I think it will take twice as long as that other user story. At home, I think pruning the trees in my yard will take twice as long as mowing the lawn.

Suppose item B in the figure below is estimated to be twice the effort of item A. And item C is then estimated to be twice that of B. This means, of course, that C is four times larger than A.

The team should consider that ratio, asking themselves, "Does C seem like it will take four times the effort of A?"

It would be ideal for a team to compare each item being estimated to all previous items. But doing so would mean that a team's 100th item would be compared against all 99 previously estimated items.

This is too inefficient to be seriously considered.

Triangulating

A more efficient strategy is to compare an item being estimated to two previous estimates. This technique is known as *triangulating*.

The two items used in the comparisons should not be randomly selected. Instead, team members should pick one estimate that is slightly smaller than what is being considered and one that is slightly larger.

For example, if a team is thinking of estimating an item as 5 story points, team members should compare it to items previously estimated as 3 and 8, if using the Fibonacci scale. It's also a good idea to occasionally compare against an item thought to be of the same size rather than one lower or higher. So when considering a 5-point estimate, team members might compare against items estimated as 3 and 5 or 5 and 8.

When I do this on, let's say, a 5-point item, I ask team members, "So, if we estimate this as 5 points, we're saying it is around 50% bigger than this three-point item. Does that seem right?"

If it does, I'll ask a similar question comparing the item to one with a larger estimate, an eight in this case. I'd say, "And we're saying it's about half the size of this 8-point item. Does everyone agree?"

Build a Reference List of Good Comparisons

It's important to select good backlog items for the comparisons. If your team feels like a particular item was incorrectly estimated, you should not use that item in future triangulations. Doing so would lead to incorrect comparisons that will lead to bad new estimates.

What I've found most useful is compiling a list of good items to use for comparisons. I can then refer to that list any time I need to find a good comparison item of any size.

While compiling this list of good backlog items to compare against, I find two criteria important:

Team members still agree with the estimate after the item has been developed

Most team members understand the item

The first point eliminates anything that was poorly estimated, such as a 5-point item that team members now think should have been estimated at 13 points.

The second point excludes esoteric items that were understood by only a small subset of the team. For example, a story may have been well estimated (the first criterion) but if it was only worked on and understood by two team members, it's not a good item to add to the list.

The Right Number of Comparison Items

A good list of favorite comparisons will contain 15–30 estimated product backlog items. Fewer than that and you'll struggle to find appropriate comparisons and will have to reuse the same items over and over again.

That's dangerous because if one of the selected items wasn't well estimated, any small error is compounded by doing frequent comparisons against it.

You can't have too many items on your comparison list. But most people fall back on using their few favorites over and over again. When manually selecting items for comparison you'll most likely choose from the same 15–30.

Age Old Items Off the List

It's useful to remove items from the list as they age. A three-year-old item might have been well estimated and understood by the majority of team members back then. But by the time it's three years old, the team may have many new members with no understanding of the old item. Even those who stayed with the team won't remember the item with sufficient clarity for it to still be a good comparison item.

So remove old items from your comparison list as they age.

Introducing Auto-Triangulation in Planning Poker™

I'm thrilled to let you know we just made triangulating your estimates easier with our beta auto-triangulate feature inside our Agile Mentors Community Planning Poker tool. And through January 11, 2022, we're making Planning Poker and this new feature completely free for anyone to use.

How Auto-Triangulate Works

Here's how it works: When a team is estimating, ANYONE on the team can click the auto-triangulate button. Normally this will be done by a Scrum Master, coach, or whoever is facilitating the Planning Poker session.

Whoever clicks auto-triangulate will be asked the number they want to triangulate around. After entering a value, the system will automatically and randomly select items with the next lower and higher values. So if, as in our example above, I want to auto-triangulate around 5, the system will display items that were

previously estimated at 3 and 8 points.

Your Favorites in Planning Poker

The items that are shown as comparisons are randomly chosen from a set of items that have been previously marked by a player as "favorites."

In Agile Mentors Planning Poker, you can easily identify an item as one you'd like to use for future comparisons. To do that, simply click the star that is shown on the top left of any current item being estimated, as shown in the following image. A yellow star indicates the item has been marked as a *favorite*. Items not selected as favorites are shown with an empty outline of a star.

You can see a list of all marked favorites by clicking the Items button in the top right of the window. That will display the Items window with a Favorites tab showing selected favorites as can be seen in the following image. From this page you can remove any item you no longer want favorited for use in auto-triangulating.

How Comparisons Are Selected

When someone clicks auto-triangulate, Planning Poker prompts them for the value they want to triangulate around. Planning Poker will then select the closest item below that value and the closest item above it. If multiple items have the same value, one is randomly selected. To see how this works, suppose you have favorited the following items:

ITEM	ESTIMATE
A	1
B	5
C	5
D	8
E	13

If you tell the system to auto-triangulate around 8 points, Planning Poker will randomly choose to display one of the two items estimated at 5 points. And it will show E, the one favorite that was estimated as 13.

If instead you request an auto-triangulation around 5 points, estimators will be shown the 1-point story and the 8-point story. This team was presumably using the Fibonacci sequence but no items estimated as 2 or 3

points had been favored. Planning Poker, therefore, selects the next lowest number, which is 1 in this case.

Your favorites persist between sessions, of course, so you can compile a list of great backlog items to compare against.

Try it for Free as a Gift From us This Black Friday

I'd love you to try out this new beta auto-triangulate feature, and through January 11, 2022 we're making Planning Poker free to try.

In fact, to celebrate Black Friday, we're offering you a free trial of the entire Agile Mentors Community.

Sign up for a free trial account before Friday December 3, 9pm Pacific, and you'll get full membership access including my resource library, my entire archive of weekly tips, a thriving forum where you can ask questions and network, and much more until January 11, 2022.

To find out more about the free trial, and to register, [click here](#).

[Get my chapters now!](#)

Posted: November 23, 2021

Tagged: estimating, planning poker, agile planning, planning meeting

About the Author

Mike is CEO of Mountain Goat Software, and one of the industry's most well-respected Certified Scrum Trainers. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile*. He is a

co-founder and former board member of the Scrum Alliance, and a co-founder of the non-profit Agile Alliance, home of the Agile Manifesto.

With 20+ years of agile training experience, Mike has honed a talent for explaining agile concepts with clear illustrations and real-life examples. Participants enjoy his passion for teaching the agile methodologies in a relatable and digestible way.

Scrum Certifications include: CSM, CST, CSPO, A-CSM, CSP-SM, CSP-PO

You may also be interested in:

What Are Agile Story Points?

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

Don't Equate Story Points to Hours

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • 39 Comments [Read](#)

For Better Agile Planning, Be Collaborative

The best plans are created by developers and stakeholders working together.

Jul 12, 2022 [Read](#)

Four Reasons Agile Teams Estimate Product Backlog Items

Estimating product backlog items provides benefits beyond predicting when a project will be finished.

May 17, 2022 [Read](#)

Estimating with Story Points Course Available for One Week

Registrations are now open to Estimating with Story Points.

Nov 17, 2021 [Read](#)

Are People Problems Creating Estimating Problems?

To create great estimates, you need to know that estimating is a human task, and humans can be complicated.

Nov 16, 2021 [Read](#)

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

by
Mike Tagged:
Cohn estimating, story points, video training
[Comments](#)

This past week on the blog, I've shared free online video training on how to use story points to create great estimates.

If you've enjoyed the free videos, why not sign up for the full Estimating with Story Points course?

It will be available to new members for one week, until **November 24 at 9 p.m. Pacific**, and then we'll close the doors to registrations.

The course includes more than 4 hours of video training, worksheets, audio, and closed captions in English, German, Spanish (Spain), Spanish (Latin America), Hindi, Dutch, Polish, and Portuguese.

It also comes with some great bonus materials to make estimating with story points even easier.

These include:

- BONUS #1 - An entire module to help you explain story points to stakeholders
- BONUS #2 - A practical module that shows you how to turn your estimates into a workable plan
- BONUS #3 - Free 12-month membership to the Agile Mentors Community (Value \$299)
- BONUS #4 - Free Planning Poker® Tool that you can use with unlimited players

This offer is only available until **Wednesday, November 24 at 9 p.m. Pacific**

[Get my chapters now!](#)

Posted: November 17, 2021

Tagged: estimating, story points, video training

About the Author

Mike is CEO of Mountain Goat Software, and one of the industry's most well-respected Certified Scrum Trainers. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile*. He is a co-founder and former board member of the Scrum Alliance, and a co-founder of the non-profit Agile Alliance, home of the Agile Manifesto.

With 20+ years of agile training experience, Mike has honed a talent for explaining agile concepts with clear illustrations and real-life examples. Participants enjoy his passion for teaching the agile methodologies in a relatable and digestible way.

Scrum Certifications include: CSM, CST, CSPO, A-CSM, CSP-SM, CSP-PO

You may also be interested in:

What Are Agile Story Points?

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

Don't Equate Story Points to Hours

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • 39 Comments [Read](#)

For Better Agile Planning, Be Collaborative

The best plans are created by developers and stakeholders working together.

Jul 12, 2022 [Read](#)

Four Reasons Agile Teams Estimate Product Backlog Items

Estimating product backlog items provides benefits beyond predicting when a project will be finished.

May 17, 2022 [Read](#)

Automatically Triangulating Estimates in Planning Poker

Comparing estimates during Planning Poker just got easier.

Nov 23, 2021 [Read](#)

Are People Problems Creating Estimating Problems?

To create great estimates, you need to know that estimating is a human task, and humans can be complicated.

Nov 16, 2021 [Read](#)

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

by
Mike Cohn **Tagged:**
estimating, story points, video training
0 Comments

This post is part of a free series of training videos covering challenges teams face when estimating with story points. We aren't charging for the training and it will be available until Wednesday, November 24 at 9 p.m. Pacific.

To watch all three videos now, [sign up here](#).

In this series so far, we've looked at solving the following problems:

Video #1: Equating Points to Hours

Video #2: Getting Hung Up on Perfect Estimates

Now, I want to share with you my final advice for solving a key estimating problem.

How People Create Estimating Problems

Even if you and your team understand the theory behind estimating and story points, if you don't account for human nature and personality dynamics, you can still encounter problems.

Someone refuses to change their estimate and discussions stall. How can you break the deadlock?

A tester won't estimate on development work because they're unfamiliar with that skill. Can you still include them?

New recruits with less experience are making the estimating process painfully slow. Should they be excluded?

These are just some of the problems we will teach you to handle in this final free video.

We've had some great responses about the videos. Remember, you can leave a comment and join in the discussion on the video page. Will I see you there?

"Thank you, Mike! I really love all your videos, they're so simple on such a difficult topic. I've sent it to my SMs to watch and then discuss with me and their teams. Great job, thank you. Looking forward for the next videos " Anastasia Butova-Nikishina

This Series is Available Only Until November 24

These videos are free to watch, but only available until **Wednesday, November 24 at 9 pm. Pacific.**

[Get my chapters now!](#)

Posted: November 16, 2021

Tagged: estimating , story points , video training

About the Author

Mike is CEO of Mountain Goat Software, and one of the industry's most well-respected Certified Scrum Trainers. He is the author of *User Stories Applied*

for Agile Software Development, Agile Estimating and Planning, and Succeeding with Agile. He is a co-founder and former board member of the Scrum Alliance, and a co-founder of the non-profit Agile Alliance, home of the Agile Manifesto.

With 20+ years of agile training experience, Mike has honed a talent for explaining agile concepts with clear illustrations and real-life examples. Participants enjoy his passion for teaching the agile methodologies in a relatable and digestible way.

Scrum Certifications include: CSM, CST, CSPO, A-CSM, CSP-SM, CSP-PO

You may also be interested in:

[What Are Agile Story Points?](#)

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments

[Read](#)

Don't Equate Story Points to Hours

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • [39 Comments](#)

[Read](#)

For Better Agile Planning, Be Collaborative

The best plans are created by developers and stakeholders working together.

Jul 12, 2022

[Read](#)

[Four Reasons Agile Teams Estimate Product Backlog Items](#)

Estimating product backlog items provides benefits beyond predicting when a project will be finished.

May 17, 2022

[Read](#)

[Automatically Triangulating Estimates in Planning Poker](#)

Comparing estimates during Planning Poker just got easier.

Nov 23, 2021

[Read](#)

Estimating with Story Points Course Available for One Week

Registrations are now open to Estimating with Story Points.

Nov 17, 2021

[Read](#)

Learn About Agile

New to Agile and Scrum?

Scrum

User Stories

Planning Poker

Agile Software Development

Agile Project Management

Agile Presentations

Mountain Goat on YouTube

Agile Mentors Podcast

Elements of Agile

Assessment

Agile & Scrum Training

Online Certifications

Video Courses

In-Person Certifications

On-Site Courses

Training Schedule

Compare Courses

PDUs and SEUs

More...

About Us

Contact Us

Our Students

Blog

Books by Mike Cohn

Agile Tools

by
Mike Cohn Tagged:
estimating, story points, video training
0 Comments

*This post is part of a free series of training videos covering challenges teams face when estimating with story points. We aren't charging for this training and it will be available until **Wednesday, November 24 at 9 p.m. Pacific.***

To watch videos one and two, and to find out when the next one is available, [sign up here](#).

Today's video looks at the problem of perfect estimates.

If a team believes that their estimates will be treated as promises, they can feel under pressure to get them exactly right.

Managers need to know when something will be delivered, but teams may worry that if they miss that deadline, they'll be punished for it.

What happens next? Exacting high pressure can result in teams refusing to estimate, or padding the estimate until they feel one hundred percent confident they can meet that deadline.

Both of these results are bad for the team, and for stakeholders.

This video provides **four coaching techniques** to resolve the problem of perfect estimates, including how to

make sure teams and stakeholders are aligned regarding the type of estimate being produced, and helping stakeholders understand that estimates are **not** guarantees.

More than 13,500 people have registered for this training, and you can watch it for free now:

Have You Experienced Problems with Perfect Estimates?

Do you feel that your estimates need to be perfect? Have stakeholders expected them to be a guarantee?

Watch the videos and let's discuss it there, or leave a comment below.

[Get my chapters now!](#)

Posted: November 11, 2021

Tagged: estimating, story points, video training

About the Author

Mike is CEO of Mountain Goat Software, and one of the industry's most well-respected Certified Scrum Trainers. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile*. He is a co-founder and former board member of the Scrum Alliance, and a co-founder of the non-profit Agile Alliance, home of the Agile Manifesto.

With 20+ years of agile training experience, Mike has honed a talent for explaining agile concepts

with clear illustrations and real-life examples.
Participants enjoy his passion for teaching the agile methodologies in a relatable and digestible way.

Scrum Certifications include: CSM, CST, CSPO, A-CSM, CSP-SM, CSP-PO

You may also be interested in:

[What Are Agile Story Points?](#)

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments

[Read](#)

[Don't Equate Story Points to Hours](#)

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • [39 Comments](#)

[Read](#)

[For Better Agile Planning, Be Collaborative](#)

The best plans are created by developers and stakeholders working together.

Jul 12, 2022

[Read](#)

[Four Reasons Agile Teams Estimate Product Backlog Items](#)

Estimating product backlog items provides benefits beyond predicting when a project will be finished.

May 17, 2022

[Read](#)

[Automatically Triangulating Estimates in Planning Poker](#)

Comparing estimates during Planning Poker just got easier.

Nov 23, 2021

[Read](#)

Estimating with Story Points Course Available for One Week

Registrations are now open to Estimating with Story Points.

Nov 17, 2021

[Read](#)

Learn About Agile

New to Agile and Scrum?

Scrum

User Stories

Planning Poker

Agile Software Development

Agile Project Management

Agile Presentations

Mountain Goat on YouTube

Agile Mentors Podcast

Elements of Agile Assessment

Agile & Scrum Training

Online Certifications

Video Courses

In-Person Certifications

On-Site Courses

Training Schedule

Compare Courses

PDUs and SEUs

More...

About Us

Contact Us

Our Students

Blog

Books by Mike Cohn

Agile Tools

Books by Mike Cohn

Agile Tools

Books by Mike Cohn

Agile Tools

Books by Mike Cohn

by
Mike **Tagged:**
Cohn estimating, story points
[Comments](#)

Suppose you have a flight to catch later today. How far ahead of your flight should you leave for the airport?

This, of course, depends on many factors. For example:

How far away from the airport you currently are

How busy do you anticipate the airport will be? This, in turn, will be impacted by the time of day your flight is.

The day of the week

Whether today is a holiday

When my wife and I were newlyweds, the question of how far ahead to leave for a flight was our most frequent source of disagreements.

Why Our Estimates Were So Different

I don't like the stress of travel. And rather than making it more stressful, I like to arrive well before a flight's departure time. It's not like the time is wasted. I can read a book or there's always email to answer.

My wife, on the other hand, likes to arrive at the gate just as the agent is ready to close the door. While growing up, her dad was an airline pilot. Whenever her family would travel, they'd fly standby, meaning they

might or might not get on the plane. And if they didn't, no worries, another plane would take off sometime.

So, my wife was accustomed to arriving as late as possible. I wanted to arrive early enough there would be no practical chance I'd miss a flight.

When we were estimating how far ahead to leave for the airport we were estimating entirely different things.

My wife was planning based on an optimistic estimate: The time that would let her arrive at the gate just before the door closed.

I was providing a risk-averse estimate.

It took us perhaps a year and a few trips as a newly married couple to figure this out and find an appropriate compromise.

I see many teams struggle with this exact situation—some developers are providing optimistic estimates, others are providing risk-averse estimates. With such differing views on what they are estimating, team members will, of course, find it difficult (or impossible) to agree on a single value as their estimate.

The Five Possible Estimates

When asked to provide an estimate for a product backlog item, there are five possible values you could provide.

Optimistic and Worst-Case Estimates

First, you could provide an *optimistic estimate*. This is an estimate of everything going well. How often does *everything* go well?

Alternatively, a team could provide a *worst-case estimate*. This is an estimate of the most effort that may be needed to deliver something.

I sometimes must rein teams in from including everything and making the worst case ridiculous. A worst-case estimate does not need to account for meteor strikes, earthquakes, hurricanes, the source repository server catching on fire, all Amazon AWS servers going down for a week, and so on.

Stop short of providing a ridiculous worst-case estimate.

A Risk-Averse Estimate

Instead of the worst-case estimate, teams could provide a *risk-averse estimate*. A team providing this wants to be safe but knows they can't include meteor strikes in their estimates. This is the estimate I was using when leaving early for the airport with my wife.

Median and Most-Likely Estimates

A fourth option would be to provide an estimate of the median amount of effort. This is the midpoint. Half the time it will take more effort than this, half the time it will take less.

Finally, a team could provide an estimate of the most likely effort. If you could somehow do the work a hundred times, this is the amount of effort it would most frequently take.

Graphing the Types of Estimates

Before deciding which of these estimates a team should provide, let's put them on a graph. Let's start with the shape of the line.

The chart below shows the effort or amount of time something will take across the horizontal axis.



On the vertical, the likelihood of it taking that long. This line is a lognormal distribution and it is how most time-based tasks are distributed.

To see why, think about going grocery shopping. There is essentially zero likelihood that going grocery shopping will take no time at all.

Most of your trips to the market take about the same amount of time. That's the big hump in the chart. And some of your trips take longer. Some, a lot longer.

I recently discovered Cosmic Crisp apples. They're just a newer type of apple—like Fuji, Pink Lady, or Honeycrisp. I love them, so when I ran out, I wanted some more.

My usual store didn't have any. Neither did the second store I went to. I finally found them at the third. That trip to do my shopping would be way out on the right.

Effort is distributed like this rather than symmetrically because there's no realistic chance that doing something takes no amount of time.

And there's a slight chance that doing it takes a really long amount of time. That creates this long tail to the right.

Adding the Estimate Types

Let's put the five types of estimates on this chart and discuss which is best for a team to provide.

The Worst-Case Estimate

Our worst-case estimate goes way to the right. When team members give what they consider a worst-case estimate, it's probably 95–99% safe. That means 95–99% of the area under the line is to the left of this point

What about the space on the line that continues further out to the right beyond what I've labeled the worst case? That's for what I called earlier the ridiculous worst case. All the extra time along the horizontal line represents the change from maybe 98 to to 99.9%. As before, there's no value in having a team come up with a ridiculous, worst-case estimate.

The Risk-Averse Estimate

Not as far out to the right is the risk-averse estimate. Think of a risk-averse estimate as being about 90% safe. Things could go wrong, and it could take longer. Risk averse is a good, safe estimate. But it isn't the worst case.

The Optimistic Estimate

On the far left of the graph is the optimistic estimate. This is everything going well. When developers give this type of estimate, most are thinking it's something they'll make perhaps 10% of the time.

Note the balance between optimistic and risk-averse: each lops off about 10% of the total area under the curve.

The Most-Likely Estimate

The most-likely estimate will always be the highest point on the chart. In a lognormal distribution, the most likely estimate will always be somewhere to the left of the median estimate. This is because of the long, gradually declining right tail of a lognormal distribution.

The Median Estimate

Finally, the median estimate is at the point where half the area under the curve is to the left and half is to the right. This indicates that half the time the actual effort will take less than the median and half the time the actual will take more.

Which Estimate Is Best?

So, which of these estimates should team members provide? Let's consider the options.

An optimistic estimate isn't very useful. Remember an optimistic estimate is one the team will make only about 10% of the time. That's just too infrequent to use as the basis of your product backlog estimates.

The same is true for risk averse and worst case. Those are just too pessimistic. You don't want to build a plan based on estimates that pessimistic. It's unrealistic and no one would ever approve the project.

A good choice we haven't considered would be the mean estimate—the average of all possible completion times. With a lognormal distribution like product backlog item completion times will be, the mean will be to the right of the median. That's because the mean accounts for all those really far right data points. Those pull the mean to the right of the median.

Although using mean values would be nice, they're too hard to estimate directly.

My recommendation is to estimate the most likely or the median value. I personally prefer median but have worked with plenty of successful teams that estimate the most-likely.

Discuss This with the Team

The most important thing is to discuss with all team members the type of estimate they should provide. If you've never discussed the type of estimate with team members, you almost certainly have people providing different types of estimates.

This is supported by [research](#) conducted by [Magne Jørgensen](#), who is the chief research scientist and professor at the Simula Research Lab in Oslo, Norway. His research has greatly influenced my views.

In one study he asked developers what type of estimate they had provided when not told in advance to provide a certain type.

Most common was an optimistic estimate (which he called an ideal estimate), provided by 37% of survey respondents. But 27% had provided a most-likely estimate. And a full 22% weren't aware of what type of estimate they had provided.

If a team has not had this discussion, team members are unknowingly providing different types of estimates. No wonder it can be hard to gain agreement on how many story points one item should have when compared to others, with one team member thinking in terms of an optimistic estimate and another thinking in terms of a risk-averse estimate of the same product backlog item.

Once you have this discussion and get everyone to agree to think in terms of either a most-likely or median estimate when deciding on story points, you'll find gaining agreement on an estimate much easier to achieve.

What Do You Think?

Have you discussed with your team which type of estimate each should provide? Which did you choose? Did agreeing on which type of estimate to create improve your estimating meetings? Please share your thoughts in the comments below.

Want more help creating estimates? Free video training available today

Over the next week, I'm opening the doors once again to my free series of training videos that make it easy to understand and explain story points.

Video one is now available, and you can register to watch it [here](#).

This video looks at the problem of **equating points to time** and what you can do to stop this happening with your team.

To get access to this video and find out when the next video is available, [sign up here](#).

The training will be available until **Wednesday, November 24 at 9pm Pacific**.

[Get my chapters now!](#)

Posted: November 9, 2021

Tagged: estimating, story points

About the Author

Mike is CEO of Mountain Goat Software, and one of the industry's most well-respected Certified Scrum Trainers. He is the author of User Stories Applied for Agile Software Development, Agile Estimating and Planning, and Succeeding with Agile. He is a co-founder and former board member of the Scrum Alliance, and a co-founder of the non-profit Agile Alliance, home of the Agile Manifesto.

With 20+ years of agile training experience, Mike has honed a talent for explaining agile concepts with clear illustrations and real-life examples. Participants enjoy his passion for teaching the agile methodologies in a relatable and digestible way.

Scrum Certifications include: CSM, CST, CSPO, A-CSM, CSP-SM, CSP-PO

You may also be interested in:

[What Are Agile Story Points?](#)

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

[Don't Equate Story Points to Hours](#)

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • 39 Comments [Read](#)

[For Better Agile Planning, Be Collaborative](#)

The best plans are created by developers and stakeholders working together.

Jul 12, 2022 [Read](#)

[Four Reasons Agile Teams Estimate Product Backlog Items](#)

Estimating product backlog items provides benefits beyond predicting when a project will be finished.

May 17, 2022 [Read](#)

[Automatically Triangulating Estimates in Planning Poker](#)

Comparing estimates during Planning Poker just got easier.

Nov 23, 2021 [Read](#)

[Estimating with Story Points Course Available for One Week](#)

Registrations are now open to Estimating with Story Points.

Nov 17, 2021 [Read](#)

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

by
Mike **Tagged:**
Cohn estimating, story points, productivity, measure
[Comments](#)

After working with story points for a bit, teams often realize they can calculate the team's cost per point. The team's product owner can use this cost per point to make decisions. For example, suppose a team's cost per point is \$2,000. Their product owner could then estimate that a 10-point product backlog item will cost about \$20,000 and be able to decide if the item is worth that investment.

Shortly after many teams realize such calculations are possible, though, they'll often ask me if they're dangerous.

Before considering whether calculating a team's cost per point is dangerous, let's first look at how it can be calculated and how it can be used.

Calculating the Cost per Point

To calculate a team's cost to deliver a story point, start by selecting an appropriate time frame over which to do the calculation. I recommend using at least three sprints' worth of data. I'd actually prefer three months of data, which will likely be more sprints, depending of course on sprint length.

For the period you've selected, calculate the total number of story points delivered. To do that, simply sum the velocities for each sprint. Let's use the data in the following velocity chart. During the seven sprints

shown (14 weeks), this team delivered 167 story points.

Next, determine how much the team was paid during the same period. The personnel department is unlikely to share individual salaries with you. But they'll probably give you the sum, and you don't need individual salaries. All you need is the total paid to the team over the period.

You'll need to think who you want to include in this calculation. Maybe your team consults with an architect occasionally. Or they work with a DevOps Engineer who isn't officially on the team but helps out a few times at least each week. It's up to you whether to include partial salaries for these folks. Usually part-timers won't materially affect any decisions based on a team's cost per point.

Additionally, you'll need to decide between using salary data and fully burdened (also called fully loaded) labor costs. Fully loaded labor costs include an allocation for office space rent, benefits, paid time off, etc. All this means is that a team member who earns \$40 per hour costs the company more than that. You can usually get the complete figure from a controller or someone in the finance group.

I recommend not bothering with loaded labor costs because, as with part-time team members, it usually won't lead to different decisions being made.

An Example

Let's suppose we have a team of eight people who were paid \$160,000 total over the 14 weeks being analyzed.

From the velocity chart above we determined this team delivered 167 story points. To determine the cost per point, divide the total cost (\$160,000) by the number of points (167) and get \$958 per point.

Using the Cost per Point

There are a number of ways a cost per point can be useful. I'll describe two. First, the cost per point can be extremely helpful to a product owner who needs to decide whether a feature is worth developing.

Prioritizing a Feature

Imagine you are a product owner whose team has just estimated a feature to be 40 points. Knowing your cost per point is \$958, the question in your mind then becomes whether the new feature is worth spending \$38,320 ($\958×40).

Actually, I'd hope you'd just round that off and think about whether the feature is worth \$40,000.

Even better would be for a product owner to understand that 40 points was the team's estimate. A decision about building a new feature should include some margin of safety in the decision. I'd recommend the product owner add at least 50% and consider whether the feature would be worth it at that cost.

Earlier today I bought an old issue of Rolling Stone magazine that I want to frame and hang on my office wall. It was \$200. Since I only have the eBay seller's word and photo to vouch for the quality of the 45-year-old issue, I had to ask myself if I'd still pay \$200 if the magazine isn't quite as pristine as I'm hoping for.

It's the same with the product owner making prioritization decisions. I don't want a product owner to have the team build something that is estimated at 40 points that the product owner would not want built at 41 points.

Estimating Development Cost

A related use of a team's cost per point is to estimate the development cost of a product, project, or set of features.

This arises frequently in contract development. Imagine a company approaches a contract development shop to have a new product built. The team estimates it at 200 points. Knowing their cost is \$958 per point, they can do the math ($200 \text{ points} \times \958) and determine the product will cost just under \$200,000 to develop.

Note that they should not then go bid \$200,000 to the client. This was an unloaded labor cost and no allowances have been made for uncertainty, project management, client management, or profit. But the \$200,000 can be used as a starting point and these additional items added to it.

The Dangers of Using a Cost per Point

Note that in each of these appropriate uses of a team's cost per point, the value was used in aggregate. It was used as a team cost rather than the cost of one person doing the work. And it was used as the average cost over a large number of points (40 and 200, in the two examples).

Danger: Using Cost per Point Too Narrowly

This is an appropriate use of cost-per-point data. Things can go wrong if the cost-per-point is applied too narrowly. A mistake I commonly see involves someone looking at the cost per point at an individual level, for the individual velocity of each member, which is a very bad idea.

Individual velocity should be impossible to calculate because a team shouldn't have backlog items that can be delivered by one person. A typical backlog item might require a programmer, a tester, a DBA, an analyst, a user interface designer, and more. With so many disciplines involved it would be impossible to allocate points to each role. So velocity can be calculated only at the team level.

Using cost per point too narrowly can occur when looking at small product backlog items, too. Think about a two-point product backlog item and suppose, for simplicity, those two points are split evenly between coding and testing.

The team has only one tester but has two coders, each equally likely to pick up the item. One programmer earns twice the salary but is actually three times faster at coding. On one small product backlog item, the cost per point could be affected by which team members do the work.

But if this same team is using their cost per point to make decisions about 100 points of work, it would be impractical to think all that work could be done by one particular developer. No—with the larger amount of work, things will average out.

You want to be careful of using cost per point at too small of a level.

Danger: Using Cost per Point to Compare Teams

The biggest danger in calculating a cost per point is the risk that someone outside the team will use it as the basis for comparison against other agile teams. This is inappropriate because the velocities of two teams are not comparable unless the teams have taken great effort to [establish a common baseline for story points](#) and to maintain it. And, even then, it [may not be a good idea to have a common baseline](#).

The desire to compare teams based on cost per point follows from the mistaken thinking that velocity can be used as a productivity measure. It can't. It is entirely possible for a team with a lower velocity to be more productive than a team with a much higher velocity.

Cost per Point Can Aid Decision Making

I believe that knowing a team's cost per point can help people choose. The metric is easy to calculate and easy to use in ad hoc situations; working with round numbers and rough estimates is enough to help a product owner decide if a feature warrants the estimated effort.

The primary drawback of it being used to compare teams is not a new foe for most Scrum Masters or coaches. Stakeholders often attempt to compare teams—but they'll attempt that with velocity alone, so knowing each team's cost per point doesn't worsen the situation.

What Do You Think?

What has your experience been with calculating a team's cost per point? Have you found it useful? Have you used it in ways other than I described? And what problems, if any, have you run into? Please share your thoughts in the comments below.

[Get my chapters now!](#)

Posted: September 28, 2021

Tagged: estimating, story points, productivity, measure

About the Author

Mike is CEO of Mountain Goat Software, and one of the industry's most well-respected Certified Scrum Trainers. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile*. He is a co-founder and former board member of the Scrum Alliance, and a co-founder of the non-profit Agile Alliance, home of the Agile Manifesto.

With 20+ years of agile training experience, Mike has honed a talent for explaining agile concepts with clear illustrations and real-life examples. Participants enjoy his passion for teaching the agile methodologies in a relatable and digestible way.

Scrum Certifications include: CSM, CST, CSPO, A-CSM, CSP-SM, CSP-PO

You may also be interested in:

What Are Agile Story Points?

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

Don't Equate Story Points to Hours

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • 39 Comments [Read](#)

For Better Agile Planning, Be Collaborative

The best plans are created by developers and stakeholders working together.

Jul 12, 2022 [Read](#)

Four Reasons Agile Teams Estimate Product Backlog Items

Estimating product backlog items provides benefits beyond predicting when a project will be finished.

May 17, 2022 [Read](#)

How Implementation Intentions Help My Sprints

Planning an exact time to do something within a sprint helps make sure you get the important stuff ...

Mar 22, 2022 [Read](#)

Automatically Triangulating Estimates in Planning Poker

Comparing estimates during Planning Poker just got easier.

Nov 23, 2021 [Read](#)

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

by
Mike Tagged:
Cohn estimating, sprinting
[Comments](#)

As well-intentioned as teams are, it's really hard to finish absolutely everything by the end of a sprint. A team may have grabbed eight product backlog items (typically user stories), but only finish six or seven of them. The other items are often close to done, but this isn't horseshoes so close doesn't count.

But what about the estimates on those unfinished product backlog items? Should they be re-estimated? And should the team get partial credit for the work they did complete? Let's consider each of these questions, starting with partial credit.

Should a Team Earn Partial Credit?

When a team finishes part of a story, team members often ask to receive partial credit toward their velocity. For example, a team that claims to be half done with an eight-point product backlog item may expect to count half of that—four points—toward their velocity.

This is a bad idea. A team should earn credit only for work that is truly done. In this example, nothing yet meets the team's definition of done. Work is only half done. Taking credit for partially done work would be like me inviting you over for dinner and serving you half-cooked chicken. It might taste OK now, but you're going to regret it later.

When a team takes credit for partially complete work, they often overestimate how much is done. There is a joke in the software development profession that says software development projects are 90% done for 90% of their schedule.

This is because developers think they see the full scope of what's needed and they are truly 90% done with that. But as they work to finish the last 10%, they realize the solution is bigger than they thought—and even after more work they are still just 90% done with the bigger scope.

So a team that says they're 50% done is perhaps only 40% or 48% or 35% done. Knowing the percentage done is very hard and most of us overestimate how done we are.

Because teams tend to think they are further along than they truly are, velocity is slightly inflated. An inflated velocity feels good at the moment—like my half-baked chicken—a team can tell its stakeholders a nice, big, juicy velocity value. But that inflated velocity will stop feeling good if anyone ever uses that artificially high velocity to predict when the next project will be done.

No Partial Credit

I advise teams to take a stance of *no partial credit*. No matter how close team members are to finishing a backlog item, they earn no points for it until the entire item is done. This is analogous to scoring a touchdown in American football. In American football, a team needs to move the ball 100 yards down the field, getting into their opponent's end zone. Doing so earns a team six points (and the opportunity to earn one or two additional points).

Moving the ball 99 yards earns the team...zero points. No partial credit.

The Benefits of a No-Partial-Credit Stance

There are two big benefits to taking a stance of no partial credit. First, team members will seek to work on smaller product backlog items in the future.

A team doesn't care if their Scrum Master refuses them partial credit on a one-point story. They do care when they can't take partial credit on an eight-pointer. Most teams will respond to not getting partial credit by splitting items into smaller discrete pieces of work.

Second, when not allowed to take partial credit, many teams resolve to try harder to complete the items they bring into a sprint. That's great—we want that.

When I refrain from allowing a team to claim partial credit, it's a double win: the team brings smaller items into their sprints and they focus more on finishing them. What's not to like?

Having resolved that a team should not earn partial credit for an incomplete product backlog item, let's turn our attention to whether that unfinished item should be re-estimated.

Re-Estimating Unfinished Work

When a product backlog item is unfinished at the end of an iteration, the team often wants to know if the item should be re-estimated. This is independent from the issue of wanting partial credit. After all, team members reason, at least some work has been done and so the estimate of remaining work may have changed.

In fact, the size of the product backlog item may have increased such that the estimate of what remains actually goes up and is higher than the initial estimate. For example, a team feels like they've made two points of progress on a five-point story but also have discovered that there are eight points remaining. In re-estimating partially finished work, it is entirely possible for the new estimate to exceed the original.

But should a team re-estimate at all?

Reasons to Re-Estimate

There are two arguments in favor of re-estimating:

1. It's more accurate since some part of the work has been performed
2. It's necessary to plan the next sprint

It's hard to argue with the first point, so I'll accept that. The second point is where things get interesting.

Team members will say they can't decide how much work to bring into a sprint if some of the backlog items have outdated estimates because they are partially complete.

For example, suppose a team has room for five more points in a sprint. An eight-point story won't fit. But what if the team thinks approximately half of the eight-point story was finished in the just-completed sprint? It sure seems like half the item could fit in the next sprint. Should it then be re-estimated at four or five points to indicate that it fits?

My opinion is that no, it should not. That is: leave the original estimate alone and bring the item into the sprint if the team thinks it fits.

The argument for resizing the backlog item assumes the team is doing [velocity-driven sprint planning](#). In that approach a sprint is planned by selecting product backlog items whose estimates sum to the team's average velocity. So if a team has a velocity of 30, they bring in 30 points of work. If they've already brought in 25 and the next item is a partially finished item estimated at eight points, that item won't fit.

But for a [variety of reasons](#), I don't think velocity-driven planning is the best approach to sprint planning. I far prefer [capacity-driven sprint planning](#), which involves the team simply selecting items one at a time and seeing if the item can be done in the sprint. If they can, they include it and consider another until the sprint is full.

Capacity-driven sprint planning uses velocity but is not driven by it. Teams using this approach aren't constrained by pulling in exactly what their velocity says they should. And so a team doing capacity-driven sprint planning will take into account the finished portion of that eight-point story when they decide whether the story will fit in their sprint.

Focus on Average Rather than Actual Velocity

In general, teams will do better if they pay a bit less attention to each sprint's velocity and more to the

average velocity of recent sprints. In the approach I'm advocating, a team can feel cheated out of a few points of velocity in one sprint since they aren't able to claim partial credit. But, since the item is not re-estimated, they're getting full credit for the work in the following sprint. It all comes out in the end.

In the example of the eight-point story that is half done, the team claims zero points of velocity credit in the first sprint despite doing perhaps three or four points of work. But they get all eight points in the next sprint. This means the average velocity for those two sprints will be exactly as one would expect.

The Sole Time to Re-Estimate Partially Finished Items

Re-estimating partially finished product backlog items is unnecessary work. There is one situation, however, in which it's advisable. A team should re-estimate a backlog item that is being put back on the product backlog and will not be finished in the next iteration or two. In that scenario it makes sense to determine a reasonable split of the item, give the team some credit for the work done, and return the item to the backlog with a good estimate of the work remaining.

In all other cases, though, a team will do well to not take partial credit and not bother re-estimating, especially if the team is doing capacity-driven sprint planning.

What Does Your Team Do?

What does your team do with unfinished work? Do you re-estimate it? Do you take partial credit? Do you never, ever, ever have anything unfinished? Please share your thoughts in the comments below.

Get Your Free Scrum Cheat Sheet

Sign up below to receive this FREE reference.

First Name

Email Address

[Privacy Policy](#)

Posted: June 15, 2021

Tagged: estimating, sprinting

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of User Stories Applied for Agile Software Development, Agile Estimating and Planning, and Succeeding with Agile as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

by
Mike Cohn
Tagged:
estimating, story points, video training
[Comments](#)

Over the past week I've shared some [free online video training](#) to help solve common problems teams face when estimating with story points.

I'm now opening the door to new registrations for the full video course: [Estimating With Story Points](#).

Estimating with Story Points helps you:

1. Communicate confidently about story points in way that overcomes resistance
2. Establish a process for estimating that brings diverse people together no matter their skill, experience, or personality
3. Know what to look out for when things go wrong and know how to fix them, so that your team can actually improve their estimates and be trusted

There are more than 4 hours of video training, plus worksheets, audio, and closed captions in English, German, Spanish (Spain), Spanish (Latin America), Hindi, Dutch, Polish, and Portuguese.

We're opening doors for only one week, but when you register today, you get a stack of bonuses as a thank-you for signing up before the deadline.

[Click here to read about the course and to register](#)

"Estimating with story points is something I find many teams struggle with. Every module in this course is packed with beautifully presented and carefully planned lessons that can change that. Mike's style and real-world examples make learning fun; while quizzes, worksheets and ongoing access to materials help us retain and practice what we've learned. It's an essential part of my toolset, and I think that you would benefit from adding it to yours too. "

David Cassidy

BONUS #1 - How to Explain Story Points to Stakeholders Module

Even if your team understands story points, it takes a different conversation to explain them to stakeholders.

This module provides additional materials to help you understand:

- What stakeholders need to know about points
- Why they can't just use person-days
- Predicting date and costs using story points
- Why you can't measure productivity or compare teams with points

BONUS #2 - Planning with Story Points Module

This was a popular bonus last time we offered the course. Now that you understand story points and can explain them to teams and stakeholders—what do you need to know about planning before you use them?

In this extra module you'll get 6 brand-new lessons that explain:

1. The components of a plan
2. Estimating what you don't know
3. Estimating velocity as a range
4. Estimating velocity without data
5. An introduction to fixed-date plans
6. An introduction to fixed-scope plans

"It's presented by Mike Cohn, one of the pioneers in Agile Estimation. Production values of the videos and materials are high. It comes with exercises and materials you can share with your team. There's lifetime access. Usually purchasing a course comes with a few months of Agile Mentors membership, which gives you access to many hours of recorded training sessions plus monthly live Q&A sessions with Mike."

Mark Nolte

BONUS #3 - Free 12 month membership to the Agile Mentors

Community (value \$299)

When you sign up today you'll also get a FREE 12-month membership to my Agile Mentors community.

This is the only place you can access:

- Monthly Q&A with myself and other agile experts
- A friendly forum with some of the sharpest agile mentors
- A library of resources with 40+ Scrum Education Units®
- Archive of all my Weekly Tips

"Take the course, absorb the material, then share it with your team. Mike has some good points, suggestions, and examples that everyone on your team can relate to. As a professional who has gone through literally hundreds of estimating ceremonies, Mike has the experience and expertise to guide you through the process and address the challenges you face. Plus, if you don't find the course useful, he'll guarantee your money back...no questions asked. You have nothing to lose, but everything to gain."

Brian Silverio

BONUS #4 - The Coolest Planning Poker Tool Around?

We like to build tools that are useful and fun to use, and our Planning Poker® tool is no exception. It allows you to play multiple games with any number of users. Your guests don't need to own the course. Create a new game, add your stories, invite users, and start estimating. You can mark items as favorites and use them to estimate new items. While estimating you can view an item, see its acceptance criteria and notes, plus edit and add backlog items.

Remember this offer is only available until **Wednesday, April 21 at 9 p.m. Pacific.**

It would be great to see you as a member.

[Get my chapters now!](#)

Posted: April 14, 2021

Tagged: estimating, story points, video training

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the *Better User Stories* video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

What Are Agile Story Points?

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

Don't Equate Story Points to Hours

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • 39 Comments [Read](#)

For Better Agile Planning, Be Collaborative

The best plans are created by developers and stakeholders working together.

Jul 12, 2022 [Read](#)

Four Reasons Agile Teams Estimate Product Backlog Items

Estimating product backlog items provides benefits beyond predicting when a project will be finished.

May 17, 2022 [Read](#)

Automatically Triangulating Estimates in Planning Poker

Comparing estimates during Planning Poker just got easier.

Nov 23, 2021 [Read](#)

Estimating with Story Points Course Available for One Week

Registrations are now open to Estimating with Story Points.

Nov 17, 2021 [Read](#)

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

by
Mike Tagged:
Cohn estimating, story points, video training
[Comments](#)

Today's post is the final installment in a [free series of training videos](#) tackling common problems teams face when estimating with story points. The training will be available until Wednesday, April 21 at 9 p.m. Pacific.

So far in this series of training videos we've looked at:

[Video #1: Equating Points to Hours](#)

[Video #2: Getting Hung Up on Perfect Estimates](#)

And in this final video I look at one of the biggest sources of estimating problems: people.

[Video #3 Overcoming People Problems to Get Good Estimates](#)

Estimating is a human task, and humans can be complicated. No matter how clearly you define and explain the process, estimates still get influenced by bias, background, and individual perspectives.

We've all been on teams where someone won't budge on an estimate, or new recruits are uncertain about the process and make it painfully slow for others. And some people just flat-out refuse to estimate anything that isn't in their skillset.

If that sounds like your team, don't give up.

There are still plenty of practical coaching solutions you can use to manage and even eliminate some of these common problems.

Jan shared her experience of estimating by skillset:

Jan Archer

Thank you for this. I've definitely encountered the 'discipline only' estimation. I think it's partly that often non-developers are intimidated by the idea of having an opinion about how long a software development might take. I guess it is down to the coach to build up their confidence and teach them that their team experience is as valuable as everyone else's, even though as you say the developer's estimate has more weight at the end of the day. More pertinent for me was the quest for the perfect estimate. Like the hypothetical VP in your example, often commercial staff want to know what to quote an external customer, and will base that quote on an estimate from the team. If the estimate is too small the job runs at a loss, if it's too big the customer feels cheated. I have in the past used the margin of error idea, and it makes a lot of sense, but if it is visible to the commercial people they will be tempted to shave it off to win business. Finally, thank you for reinforcing the point that we all need to be giving the same estimate. I will be using that curve you demonstrated in the future!

[+ Reply](#) [Share](#)

In this video I show you how to overcome some common people problems, including:

How to handle people who feel they can't estimate outside of their skillset

Making sure everyone knows what stage of completion they're estimating to

How to create a good baseline for people to estimate against

How to prevent new members from slowing down or frustrating the rest of the team

Last Chance to Watch

Remember, this free training will only be available until April 21. I've really enjoyed sharing this series and hearing how people are going to implement the coaching techniques:

Bev

Great series of videos, Mike! I really appreciate how you break down the recommendations for overcoming these misconceptions about estimating and story points into analogies. Also, the summaries at the end help drive home your main points. In the first video I learned about triangulating. My team usually does relative sizing but only considering the new stories we are refining and then estimating. I'm going to see if they'd like to instead compare the new stories to an "old" story they completed and felt good about, as far as accuracy of the estimate. I took a lot of notes from the third video as well. I'm now going to watch the second one. Thanks again for sharing these (FREE!) videos!

[+ Reply](#) [Share](#)

Lauren P

Hey Mike, this was a great video series. You addressed some of the common problems I faced with estimating using story points and you offered solutions. I'm getting ready to launch estimating again with my teams and I will have more confidence because I'm prepared for objections and problems that will arise. Thanks so much!!!

[+ Reply](#) [Share](#)

Don't miss this chance to watch all three videos, and let me know what you think.

[Get my chapters now!](#)

Posted: April 13, 2021

Tagged: [estimating](#), [story points](#), [video training](#)

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

What Are Agile Story Points?

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

Don't Equate Story Points to Hours

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • 39 Comments [Read](#)

For Better Agile Planning, Be Collaborative

The best plans are created by developers and stakeholders working together.

Jul 12, 2022 [Read](#)

Four Reasons Agile Teams Estimate Product Backlog Items

Estimating product backlog items provides benefits beyond predicting when a project will be finished.

May 17, 2022 [Read](#)

Automatically Triangulating Estimates in Planning Poker

Comparing estimates during Planning Poker just got easier.

Nov 23, 2021 [Read](#)

Estimating with Story Points Course Available for One Week

Registrations are now open to Estimating with Story Points.

Nov 17, 2021 [Read](#)

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

by
Mike **Tagged:**
Cohn estimating, story points, video training
[Comments](#)

*Today's post introduces the second installment in a series of [free training videos](#) tackling common problems teams face when estimating with story points. The training will be available until **Wednesday, April 21 9 p.m. Pacific**. To watch this video and find out when the next video is released, [sign up here](#).*

In [video one](#), I shared a sample of feedback we received about our coaching advice, designed to help teams resist the temptation to equate points to time.

But even when teams **do** understand this, they can run headlong into another problem:

The problem of perfection.

Video #2 Getting Hung Up Trying to Create Perfect Estimates

One problem teams face when trying to create 'perfect' estimates (other than the fact that perfect estimates don't exist) is that teams and stakeholders aren't all in agreement about what an estimate is, and how it should be used.

Management needs to prioritize work, and they also love metrics. So they tend to push for estimates that will tell them when something will be done.

But most teams can tell you: managers can take an estimate as a promise and become angry if it's missed. Uncomfortable situation.

This can lead to teams padding their estimates to avoid punishment, but it's a terrible idea to do this (watch the video to see why).

Elaine could relate to this. Her team was being forced to give estimates that were used for burndown. Video 2 gave her some new ideas to try:

Elaine O' Grady

I could relate so much to Video 2. Unfortunately for me I am in that 'rare' space where my teams have to provide estimates that will be used to burndown towards a product launch deliverable a year away. Estimates are viewed as absolutes by some management. The teams are stressed. And I am struggling to get through to the management level who are pushing them. This video will certainly help. Thank you very much for sharing.

[» Reply](#) [» Share](#)

Chris Exley was also facing this same challenge—and liked the ideas he learned in the training video on how to explain it to stakeholders and management.

Brian E. Jones

Very good series about story point estimating, Mike. I really appreciate your insights and tips. I have been fortunate to follow in the footsteps of others who trained our early adopters of agile well in this regard. Our teams do well in relative estimating but you brought out some extra points where we could definitely improve. Your discussion about best-case/worst-case scenarios and getting individuals and teams estimating the same thing was very helpful. I will also be using your graph to help management and stakeholders understand where our teams are estimating and what that exactly means. Simple but very useful tools to add to the old agile tool belt! Thanks for sharing!

[» Reply](#) [» Share](#)

Watch video 2 to learn:

What leads teams to pad estimates, and why it might seem like a good idea—but isn't (and learn why padding often doesn't even prevent late deliveries)

Why it's important to discuss with your team **what type** of estimate you want to create

How to deal with management demands for guarantees instead of estimates

Problems with Perfection?

Perfect estimates don't exist, but that doesn't stop teams trying to create them. Have you experienced this?

Have you struggled to explain to stakeholders that estimates are **not** a guarantee? Register to watch the videos and join in the discussion there, or leave a comment below.

[Get my chapters now!](#)

Posted: April 8, 2021

Tagged: estimating, story points, video training

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of User Stories Applied for Agile Software Development, Agile Estimating and Planning, and Succeeding with Agile as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

What Are Agile Story Points?

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

Don't Equate Story Points to Hours

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • 39 Comments [Read](#)

For Better Agile Planning, Be Collaborative

The best plans are created by developers and stakeholders working together.

Jul 12, 2022 [Read](#)

Four Reasons Agile Teams Estimate Product Backlog Items

Estimating product backlog items provides benefits beyond predicting when a project will be finished.

May 17, 2022 [Read](#)

Automatically Triangulating Estimates in Planning Poker

Comparing estimates during Planning Poker just got easier.

Nov 23, 2021 [Read](#)

Estimating with Story Points Course Available for One Week

Registrations are now open to Estimating with Story Points.

Nov 17, 2021 [Read](#)

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

by
Mike Cohn **Tagged:**
estimating, story points, video training
0 Comments

Today's post introduces the first installment in a [free series of training videos](#) to help teams use story points to create estimates. The training will be available until Wednesday, April 21 at 9 p.m. Pacific.

To watch the first video and find out when video #2 is released, sign up [here](#).

Last year I released a free series of training videos about estimating with story points. More than 5,900 people took the training, and we had hundreds of comments in the discussion area.

I've decided to release it again, and we're now offering closed captions in multiple languages. So if you'd like to watch the videos with closed captions, you can now choose from German, Spanish, Hindi, Dutch, Polish, and Portuguese!

What People Loved About Video #1: Equating Points to Hours

One universal problem that really resonates with people is teams that insist on equating points to hours. We're so used to time-based methods of estimating that getting a team to really understand what we mean by *relative*, *abstract*, and *effort* can be hard work.

As Leise commented, setting expectations from the start about story points definitely makes it easier:

Leise Astrid Passer Jensen

Hi Mike, this is a great initiative, Thx a lot. My takeaway from video #1 was that I was right in the past when I always have said and experienced this: once a team had started with comparing a SP with a certain number of hours, it is almost impossible to get them to think differently. Thx for confirming that - it stresses how important it is that we educate people in *relative* estimation from the beginning. Looking forward to your next video.

[+ Reply](#) [+ Share](#)

Starting Out or Trying to Reset Bad Habits? This Video Will Help

If you're introducing points to your team for the first time, this video is perfect for getting everyone on the same page.

But what if you don't have the luxury of working with a brand-new team? Well, if the best time to teach a team about story points was 20 years ago, the second-best time is today—if you do it in the right way.

It helps to start with the theory, of course, but to make it really sink in you have to go beyond that and use multiple coaching examples and illustrations.

As Marageret saw, even the language you use to describe and coach teams about story points can make a difference:

Margaret Clark

Hi Mike, I really enjoyed this first video and was able to relate as I have been in many teams and SM'd many teams where the struggle is real in defining what a story point is. My biggest takeaway from this is not only around resetting the definition of story points being relative and abstract measurements of effort but also with my own language as a SM when I am coaching my team in their journey. This video series is great. Thank you.

[+ Reply](#) [+ Share](#)

But what if people are so resistant to the idea of story points that team members just don't want to listen to your theory or your illustrations? I have some practical tips in this video to help:

Jennifer Shah

This is fantastic and easily understandable. I just went through the painful phase of reminding myself why the team can't just equate points to hours and we've just begun resetting their understanding of points overall. Just finished sprint planning using our new pointing model and there's a lot of resistance because it feels like it's taking too long. But the Triangulation method should make that run much more smoothly. Thanks for this! Looking forward to sharing it with my teams.

[+ Reply](#) [+ Share](#)

Watch now to learn:

How to overcome the temptation to give in to a team's desire to equate points to hours—specific ways to explain it confidently

The problems you'll encounter using a points-to-time approach

An overview of why points are abstract, relative, and effort, with coaching illustrations you can use to explain this

The tell-tale signs that signal your team is still thinking in terms of time (and why sometimes this is okay)

This video will help you **start** to weed out those deep-rooted problems and bad habits teams have picked up from equating points to time. It clears the path to save time, so you can estimate rather than argue, and have better discussions about the effort required to deliver your work.

Join the Conversation

We're going to be covering the topic of estimating with story points over the next week, so sign up to watch the videos and add your voice to the conversation, or leave a comment below.

[Get my chapters now!](#)

Posted: April 6, 2021

Tagged: estimating, story points, video training

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com.

If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

[What Are Agile Story Points?](#)

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments

[Read](#)

[Don't Equate Story Points to Hours](#)

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • [39 Comments](#)

[Read](#)

[For Better Agile Planning, Be Collaborative](#)

The best plans are created by developers and stakeholders working together.

Jul 12, 2022

[Read](#)

[Four Reasons Agile Teams Estimate Product Backlog Items](#)

Estimating product backlog items provides benefits beyond predicting when a project will be finished.

May 17, 2022

[Read](#)

[Automatically Triangulating Estimates in Planning Poker](#)

Comparing estimates during Planning Poker just got easier.

Nov 23, 2021

[Read](#)

Estimating with Story Points Course Available for One Week

Registrations are now open to Estimating with Story Points.

Nov 17, 2021

[Read](#)

Learn About Agile

New to Agile and Scrum?

Scrum

User Stories

Planning Poker

Agile Software Development

Agile Project Management

Agile Presentations

Mountain Goat on YouTube

Agile Mentors Podcast

Elements of Agile Assessment

Agile & Scrum Training

Online Certifications

Video Courses

In-Person Certifications

On-Site Courses

Training Schedule

Compare Courses

PDUs and SEUs

More...

About Us

Contact Us

Our Students

Blog

Books by Mike Cohn

Agile Tools

by
Mike **Tagged:**
Cohn estimating, sprint planning
[Comments](#)

When I was young, my mom taught me to tie my shoes. I no longer remember the exact steps, but it had to do with bunny ears.

Since then I've learned there are many different ways to tie my shoes, many of which are better. Most are better for specific circumstances: wide feet, narrow feet, high arches, and so on.

Just as I've learned a lot since tying my first shoes, I've also learned a lot since working on my first agile teams.

Sprint Planning on Early Teams

On my early agile projects, team members signed up for tasks during the sprint planning meeting. We would leave a planning meeting with someone's name associated with each task. Our sprint backlog, which we called a sprint plan back then, looked like this.

There were some obvious advantages to this.

- It was incredibly clear who would be responsible for each task.
- Team members felt greater responsibility for their tasks.
- It helped the team see if anyone was overcommitted for the sprint.

The biggest drawback to the approach was that team members were making decisions about who would work on things when the least was known about that work. Many of my early teams ran four-week sprints. This meant they were deciding who would work on tasks that might start 20 days in the future. Obviously, quite a lot could change during those 20 days.

Moving Away from Signing up During Sprint Planning

I persisted with this approach for quite a while. I even advocated it in my [User Stories Applied](#) book, which was published in 2004.

But, just as with tying shoelaces, I learned there was a better way. Rather than having team members sign up for every task during sprint planning, I learned it was better to have team members choose their next tasks as the team progressed through the sprint.

Usually this would happen during [daily standup meetings](#). As team members reported they had finished one task, they'd select their next task.

This works really well. Decisions about what to work on next can consider the full state of the iteration. Questions such as these could be considered:

- Is the team behind on a certain product backlog item?
- Is the sprint goal at risk?
- Is one type of work (e.g., testing) falling behind?

There Was a Problem

This real-time allocation of work really helped. Teams that used this approach were better able to adjust as things changed during an iteration.

There was just one problem: During sprint planning, how could a team know if anyone was taking on too much work?

This had been easy to assess when all tasks were allocated to individuals during sprint planning. Each team member could look at the set of tasks they were taking on and decide whether it was neither too much nor too little but just right.

But when tasks weren't allocated during sprint planning, that assessment was impossible.

Finding a Path Through the Work

Without a name next to every identified task, teams found it harder to decide whether they were bringing the right amount of work into the sprint.

To better assess the amount of work being brought into the sprint, I began asking teams to "find a path through the work." Think of this as team members each pointing to the set of tasks they thought they'd work on during the sprint. For example, you point to four tasks you anticipate doing. I point to a different four. And our remaining team member points to the other three.

If every task has someone saying "I'll do it" and if no team member feels overloaded with work, the selected work is most likely achievable.

When team members "found a path through the work," it was just one of multiple possible paths. Team members were not expected to do exactly the tasks they'd indicated. Rather, there simply needed to be at least one possible path through the work for everyone to feel the sprint was well planned.

To promote the transient nature of this tentative allocation of work, some teams referred to this as "signing up in pencil." In pointing at a set of tasks and saying, "I intend to do these," team members were signing up, but in pencil that was metaphorically erased at the end of the meeting, leaving each task without being officially allocated to anyone.

Estimating Without Knowing Who Would Do Each Task

An additional problem these teams faced was estimating how long a task would take without a specific person allocated to the task. What estimate should be put on a task that will take either 5 or 10 hours depending on which team member does it?

The idea of signing up in pencil helped here, too. The estimate put on a task could be the estimate for how long the work would take for the person stating an intent to perform the work.

So, your estimates go on the tasks you think you'll do and my estimates go on those I think I'll do.

Then, during the iteration, when someone started a task they would change the estimate at the moment they grabbed that task as their next. In that way I might grab a task and immediately change its estimate from 5 to

8 hours if I'm slower at that work than the person who had originally planned to do the work.

This approach works because it usually represents just a swap of hours between team members. You grab a task I'd intended to work on and change my estimate. And I grab a task you'd planned to do and change yours.

And even if the new estimates are higher, the current allocation of work is always expected to represent the best, currently knowable path through the work. In other words, team members are seeking the optimal path through all that needs to be done in an iteration.

What I Currently Recommend

I'm a big fan of delaying the decision about who will work on something until closer to when the work begins. And so I prefer not allocating all tasks during sprint planning. Members of teams I coach will usually each sign up for just their first task at the end of sprint planning. That allows everyone to leave a sprint planning meeting knowing exactly what work to begin.

While I prefer not allocating all tasks during sprint planning, I think there are two cases in which a team can benefit from signing up for every task during sprint planning:

- Teams that are new to agile
- Teams that struggle with individual accountability

Individual and Team Accountability

As noted earlier, when team members sign up for tasks they feel a greater sense of individual accountability toward those tasks with their names on them. Seeing my name next to a particular task will make it more likely I complete the work on that task.

But this comes at the expense of *team accountability*. I feel more accountable for tasks with my name on them but less accountable for those with your name.

A team will be at its best with a strong sense of team accountability (everyone is accountable for all the work). But it's hard to create team accountability without having individual accountability first.

So if there is a low sense of individual accountability on your current team, you may want to allocate all the tasks in sprint planning for a while. Once individual accountability has increased, consider not allocating every task during sprint planning and shifting toward feelings of team accountability.

Teams that Are New to Agile

Teams that are new to agile may also benefit from allocating all tasks during sprint planning for a while. When the close collaboration of agile is new to a team, it can be hard to imagine how a team can commit to finishing a set of work without first divvying up all work among team members.

An Easy, Hybrid Way to Get Started

For any team that wishes to move away from assigning all tasks during sprint planning, there is fortunately an easy way of making the shift. Do so by gradually reducing the number of tasks that are allocated to specific people during the planning meeting.

If your team allocates all work today, try allocating only 50% of the tasks next time. Team members would then walk out of the next planning meeting with, say, 5 tasks instead of 10. The remaining tasks—about half the work of the sprint—are not yet allocated to team members. Those tasks will be allocated as the sprint progresses.

I feel comfortable almost guaranteeing that this will feel no better or no worse to the team. But it's a good step in the direction of minimal allocating. So have the team allocate half of the work for a couple of sprints.

Do it until the team becomes comfortable with the idea. And then dial it up. Instead of allocating 50% of the tasks during the planning meeting, allocate only 25%. Do that a few times until everyone is again comfortable with the approach.

Next, go further. Allocate only about 10% of tasks during the planning meeting. And finally consider going all the way. Even then, most teams will have each team member sign up for the one or two tasks they do immediately after the planning meeting.

What Do You Do?

As we gain experience, we find new and better ways of working. Just as I outgrew tying my shoes by thinking about bunny ears, teams I coach shift away from allocating tasks during sprint planning as those teams gain experience.

What does your team do? When do you determine who will work on the tasks of your sprint backlog? Please share your thoughts in the comments below.

Posted: March 30, 2021

Tagged: estimating, sprint planning

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the *Better User Stories* video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

Why I Don't Emphasize Sprint
Goals

What Are Agile Story Points?
Story points are perhaps the most

Don't Equate Story Points to
Hours

For Better Agile Planning, Be
Collaborative

3 Ways to Help Agile Teams
Plan Despite Uncertainty

Four Reasons Agile Teams
Estimate Product Backlog Item

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile Assessment		

by Tagged:
Mike estimating, story points, news
Cohn

Estimating With Story Points is currently closed for new registrations. You can [register here](#) to find out when doors open again.

This past week I've shared free [online video training](#) to help tackle some of the most common problems teams face when estimating with story points.

If you've enjoyed the training, I'd like to now introduce you to my [new](#) video course:

Introducing: Estimating with Story Points

"As an experienced scrum master, I'd

unknowingly picked up some of the bad estimating habits that are prevalent across many teams in my current organization. Not only did the course make me aware of this, it has shown me how I can help others to rediscover the power of using story points in the **right** way. I've started using this knowledge with my team, and I'm seeing improvements already."

Jane Johnson

Estimating with Story Points helps you master the estimating process with your team, whether you are starting out on your agile journey or a seasoned professional.

Highlights of the course include:

- 7 modules and almost **4 hours of training** on creating estimates with story points
- 1 BONUS module to help you introduce story points to stakeholders
- 40 lessons with transcripts, worksheets, audio files, and captions for every video
- BONUS Planning Poker™ tool for unlimited users
- BONUS Free 12 month membership to my AgileMentors community (value \$708)

Because of the high level of interest we've already had, we're only opening the doors for one week to make sure all the new members get settled.

Registration for the course will close Wednesday, October 28 at 9pm US Pacific

[Click here to read more about the course and to register.](#)

What People Are Saying about Estimating with Story Points

We recently finished a beta launch in which a number of agilists worked through the course, providing feedback along the way. This let us tweak, polish, and finish it so it's even more practical and valuable.

Here's what people had to say:

I have been a fan of Mike's teaching for years and the latest class is among his best. If you have a team or organization that has fallen into some bad habits with story points or if you have a team who has yet to experiment with them, then this class will empower you with the knowledge and tools to make a change for the better. *Dana Baldwin*

Take the course, absorb the material, then share it with your team. Mike has some good points, suggestions, and examples that everyone on your team can relate to. As a professional who has gone through literally hundreds of estimating ceremonies, Mike has the experience and expertise to guide you through the process and address the challenges you face. Plus, if you don't find the course useful, he'll guarantee your money back...no questions asked. You have nothing to lose, but everything to gain.*Brian Silverio*

"As a reader of Mike's great book, "Agile Estimating and Planning," I was pretty confident that I was on the right path, but I was also aware of a lot of problems I didn't know how to overcome and I can't think of anybody better than Mike to get the knowledge from. This is like a fishing vest. I had my main pockets filled with very valuable knowledge, but I had a lot of other, valuable pockets empty. Thanks to Mike, now I have these pockets filled with new tools, advice, and tips & tricks, so now I'm confident I can greatly improve my estimation skills."
Mariano de la Fuente

[Get my chapters now!](#)

Posted: October 21, 2020

Tagged: estimating, story points, news

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

[What Are Agile Story Points?](#)

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments

[Read](#)

[Don't Equate Story Points to Hours](#)

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • [39 Comments](#)

[Read](#)

[For Better Agile Planning, Be Collaborative](#)

The best plans are created by developers and stakeholders working together.

Jul 12, 2022

[Read](#)

Announcing the New Agile Mentors Podcast

Don't miss this new podcast from the Scrum and agile experts at Mountain Goat Software.

May 31, 2022

[Read](#)

Four Reasons Agile Teams Estimate Product Backlog Items

Estimating product backlog items provides benefits beyond predicting when a project will be finished.

May 17, 2022

[Read](#)

My Most Popular Posts of 2021

Here are my top ten blog posts from last year based on comments and views.

Jan 25, 2022

[Read](#)

Comments on this post are closed.

Learn About Agile

New to Agile and Scrum? [Online Certifications](#)

Scrum [Video Courses](#)

User Stories [In-Person Certifications](#)

Planning Poker [On-Site Courses](#)

Agile Software Development [Training Schedule](#)

Agile Project Management [Compare Courses](#)

Agile Presentations [Books by Mike Cohn](#)

Mountain Goat on YouTube [Agile Tools](#)

Agile Mentors Podcast [PDU and SEUs](#)

Elements of Agile [PDUs and SEUs](#)

Assessment [Books by Mike Cohn](#)

Agile & Scrum Training

About Us [About Us](#)

Contact Us [Contact Us](#)

Our Students [Our Students](#)

Blog [Blog](#)

Books by Mike Cohn [Books by Mike Cohn](#)

Agile Tools [Agile Tools](#)

More...

Exciting New Course on Estimating with Story Points Now Open

by **Tagged:**
Mike estimating, story points, estimates
Cohn

This post was part of a free series of training videos to help people understand and explain story points. It's not currently available but you can [register here](#) to find out when it's next released.

*Today's post introduces the final installment in a [free series of training videos](#) tackling common problems teams face when estimating with story points. The training will be available until **Wednesday October 28, at 9 pm Pacific US.***

In this series so far we've given practical coaching tips to explain story points to teams and stakeholders, and we've seen that an estimate doesn't need to perfect—in fact, shouldn't be perfect—if you want it to have value.

We've had a tremendous response to the videos so far, and we're now offering closed captions in multiple languages. So if you'd like to watch the videos with closed captions you can now choose from: German, Spanish, Hindi, Dutch, Polish and Portuguese!

If you've used some of the coaching techniques in these videos, you may have already seen an improvement in the understanding of estimates.

But there's one more problem I want to cover in this series and that is dealing with people problems when you estimate.

When I say *people problems*, I'm not for one minute saying your team members are problem people; I'm sure they're awesome. It's just that people can be complicated.

And these complications can definitely affect the value of your estimates. For example:

People with a bad experience of estimating may go along with the process, but not really put the effort in.

New recruits may stall an estimating session or be intimidated by more dominant voices in the room.

Some members may feel very strongly that they should only estimate stories that fall into their specific skill or area of expertise.

If you can't wrangle your individual team members with hidden biases and opinions—not to mention varying areas of skill and experience—you'll struggle to work as a cohesive team and produce a valuable estimate.

Fortunately, you don't need to be a qualified psychologist to take care of these human quirks and be successful estimating with story points.

In this video I'm going to share—and help you solve—five common problems that can derail your estimating process, including:

Programmers and testers wanting to estimate their own type of points

People making different assumptions about how *done* the work needs to be

Team members not choosing good old items to compare against

New team members not being accustomed to your estimating scale

People getting burned out through too much estimating

This is the final video in this short training series and will be available only until **Wednesday, October 28**. So sign up to watch all three videos now.

Care to Share YOUR People Problems?

I'd like to know how you've managed people during estimating sessions to get everyone to work together. Or has it been a challenge to overcome resistance, or the desire to estimate based on skill? Don't hold back, share your people problems with us—although I recommend you keep names anonymous to protect your colleagues!

[Get my chapters now!](#)

Posted: October 19, 2020

Tagged: estimating, story points, estimates

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

What Are Agile Story Points?

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

Don't Equate Story Points to Hours

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • 39 Comments [Read](#)

For Better Agile Planning, Be Collaborative

The best plans are created by developers and stakeholders working together.

Jul 12, 2022 [Read](#)

Four Reasons Agile Teams Estimate Product Backlog Items

Estimating product backlog items provides benefits beyond predicting when a project will be finished.

May 17, 2022 [Read](#)

Automatically Triangulating Estimates in Planning Poker

Comparing estimates during Planning Poker just got easier.

Nov 23, 2021 [Read](#)

Estimating with Story Points Course Available for One Week

Registrations are now open to Estimating with Story Points.

Nov 17, 2021 [Read](#)

Comments on this post are closed.

Learn About Agile **Agile & Scrum Training** **More...**

New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

by Mike Cohn
Tagged:
estimating, story points

This post was part of a free series of training videos to help people understand and explain story points. It's not currently available but you can [register here](#) to find out when it's next released.

Today's post introduces the first installment in a [free series of training videos](#) to help teams use story points to create estimates. The training will be available until Wednesday October 28 at 9pm Pacific.

To watch the first video and find out when the next video is available, [sign up here](#).

Over the next week I'm going to release some new (and free) training videos I created to tackle common problems teams face when estimating with story points.

Earlier this year I posted a survey to discover what challenges people had, and got more than 2,400 responses from Scrum Masters, agile coaches, product owners, and managers, who highlighted the following issues:

- Team members equating points to time
- People refusing to estimate or demanding to know every detail about a story before estimating
- Tension with stakeholders who treat estimates as guarantees
- People with different skills and experience unable to reach an agreement
- A lot of wasted time and frustration

If you're struggling with the same issues, I think you'll enjoy these videos. It's free to register and you'll have the chance to comment and discuss the lessons shared in each video.

[Get instant access to Video #1](#)

At the end of the training, there will also be an option to unlock more in-depth training (details about that coming soon).

Video #1: Equating Points to Hours

It's no surprise that after years of using time-based methods to estimate work, teams can struggle with the abstract nature of story points. Some people find it difficult to detach from the idea that a story point should equal some number of hours.

But unless you change your perspective, not only will your estimates have no value, you'll experience endless arguments between team members who simply cannot agree how long something will take. When you equate points to hours, a common situation is a junior developer saying to a senior developer:

"Sure it's 5 points if you do it, but if I do it, it's 8 points."

And they're right...if they only think in time.

Now, some teams think that this won't be a problem if they know in advance who will do the work, but in my experience that's not the case. At some point, multiple team members will work on one story. Sure, one person may take the lead on programming a feature, but it will still need testing, designing, and so on.

As a result, everyone needs to estimate together...and agree. And you can't do that if people equate points to hours.

Watch Video One now to discover:

Why it doesn't suffice to simply tell team members not to equate points to time

Why you shouldn't give into a team's desire to equate points to hours, even if it seems the path of least resistance

The tell-tale signs your team is thinking in time

A simple overview of why points are abstract, relative, and about effort. This is a great starting point for introducing story points if your team is new to the concept

Two practical coaching techniques you can use to encourage relative estimation

This video will help you **start** to conquer those deep-rooted problems and bad habits teams have picked up from equating points to time.

Your new techniques mean you can save time—estimating rather than arguing—and have more peaceful and productive discussions about the effort required to deliver your work

Does your team struggle with equating points to time?

Please weigh in on this one in the comments, since we're going to be talking all things to do with story points and estimating over the next week. What signs do you see of teams struggling with this problem? Let me know in the comments.

[Get my chapters now!](#)

Posted: October 13, 2020

Tagged: estimating, story points

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

[What Are Agile Story Points?](#)

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

[Don't Equate Story Points to Hours](#)

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • 39 Comments [Read](#)

[For Better Agile Planning, Be Collaborative](#)

The best plans are created by developers and stakeholders working together.

Jul 12, 2022 [Read](#)

[Four Reasons Agile Teams Estimate Product Backlog Items](#)

Estimating product backlog items provides benefits beyond predicting when a project will be finished.

May 17, 2022 [Read](#)

[Automatically Triangulating Estimates in Planning Poker](#)

Comparing estimates during Planning Poker just got easier.

Nov 23, 2021 [Read](#)

[Estimating with Story Points Course Available for One Week](#)

Registrations are now open to Estimating with Story Points.

Nov 17, 2021 [Read](#)

Comments on this post are closed.

Learn About Agile **Agile & Scrum Training** **More...**

New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

by
Mike Cohn
31 Comments
Tagged:
estimating, story points

Are We Really Bad at Estimating?

The topic of estimating can be a contentious one. Teams may be reluctant to estimate for fear of creating something stakeholders will use against them. Clients and stakeholders will always want to know what will be delivered and when, but how do you predict this if the work is new or unfamiliar? And if you work in an environment where estimates are usually wrong, should you skip estimating altogether and just get on with building something?

I believe a lot of the frustration and issues surrounding estimates stems from the belief that, as humans, we're just bad at estimating.

But I don't think that's true.

We're Actually Pretty Good at Estimating (Some Things)

Don't get me wrong—we're definitely bad at estimating some things. But others we're quite adept at.

For example, later today I plan to write another blog post. I estimate it will take two hours to complete the first

draft of that. I'm pretty sure it won't take exactly two hours but it will probably take between one-and-a-half and three hours. For the purpose of planning my afternoon, that's a good estimate.

Back when I was teaching in-person [Certified ScrumMaster® courses](#), I would set up the room the day before. I would put a lot of supplies out for each person in the class. I had to hang some posters on the wall. And so on. From experience, I'd estimate that a typical room set-up would take 45 minutes. I've set up for so many Certified ScrumMaster courses that I feel fairly confident in that estimate.

There are probably a myriad of similar tasks that you find yourself estimating (successfully) most days—whether it's fixing dinner, driving to a friend's house, or going grocery shopping.

We're pretty good at estimating these things because we have a certain level of familiarity with them. We're not as good at estimating things we aren't familiar with.

Data supports my claim that we're not really that bad at estimating. In a [review of the existing research on estimates](#), University of Oslo professor and Chief Scientist at the Simula Research Laboratory Magne Jørgensen found most estimates to be within 20 to 30% of actuals. And on software projects, he did not find an overall tendency for estimates to be too low:

The large number of time prediction failures throughout history may give the impression that our time prediction ability is very poor and that failures are much more common than the few successes that come to mind. This is, we think, an unfair evaluation. The human ability to predict time usage is generally highly impressive. It has enabled us to succeed with a variety of important goals, from controlling complex construction work to coordinating family parties. There is no doubt that the human capacity for time prediction is amazingly good and extremely useful. Unfortunately, it sometimes fails us. —Magne Jørgensen

But if we're not too bad at estimating, why is there a common perception that we are?

One Answer Lies in the Projects We Never Start

Imagine a boss who describes a new product to a team. The boss wants an estimate before approving or rejecting work on the project. Let's suppose the project, if played out, would actually take 1,000 hours. Of course, we don't know that yet, since the team is just now being asked to provide an estimate.

For this example, let's imagine the team estimates the project will take 500 hours.

The boss is happy with this and approves the project.

But...in the end it takes 1,000 hours of work to complete. It comes in late and everyone involved is left with a vivid memory of how late it was.

Let us now imagine another scenario playing out in a parallel universe. The boss approaches the team for an estimate of the same project. The team estimates it will take 1,500 hours.

(Remember, you and I know this project is actually going to take 1,000 hours but the team doesn't know that yet.)

So what happens?

Does the team deliver early and celebrate?

No. Because when the boss hears that the project is going to take 1,500 hours, she decides not to do it. This project never sees the light of day and no one ever knows that the team overestimated.

A project that is underestimated is much more likely to be approved than a project that is overestimated. This leads to a perception that development teams are always late, but it just *looks* that way because teams didn't get to run the projects they had likely overestimated.

Our Overconfidence Often Leads to Incorrect Estimates

As I mentioned, we're not necessarily bad at estimating, but we can definitely get it wrong. In my experience, this usually stems from an overconfidence in our ability to estimate.

I've done an exercise previously to help people see this: I ask a series of ten questions in class and ask people to provide me with a range that they are 90% confident will contain the answer.

For example, I might ask you to estimate when the singer Elvis Presley was born. And I ask you to give me a range of years that you are 90% certain will contain the correct answer. If you're a huge Elvis fan, you'll know the year he was born. You might even know the exact date, and as a result, you wouldn't likely need to give a range of years

But let's say you're less of a fan, and your range is going to be wider.

You might start by thinking, "Didn't he have some hit records in the fifties? Or was it the sixties?"

Remember, I want you to give me a range of years that you are 90% confident will contain the correct year. You might then think that if Elvis was recording at the age of 20, and had hits in the fifties, an early year of his birth could be 1930.

And at the upper range, if he was recording in the sixties, perhaps he wasn't born until 1940.

So your range might be: 1930–1940.

And in this case, you'd be correct, since Elvis was born in 1935.

But Elvis was an iconic figure and many people will be familiar with when he was alive.

So for my other questions I deliberately ask for answers that might be more difficult to narrow down. For example I might ask how many iPads were sold in 2019, or how many athletes competed in the 2016 Olympics, or for the length of the Seine River.

Now, for each question, the parameters remain the same:

Provide a range of numbers (units/athletes/miles, etc.) that you think contains the **correct answer**.

Be 90% confident that the correct answer is within that range.

What happens is surprising! Even though I tell them to give me ranges that are 90% certain of, **most people get most questions wrong.**

The ranges they provide are usually much smaller than they should be when considering their unfamiliarity with the subject.

For example, let's imagine that I have no idea how many iPads were sold in 2019. If I want to be 90% certain the range I give contains the correct answer, I should provide a huge range, say zero to one billion. That would be widely exaggerated, but I'd be confident in my answer.

In my experience, the ranges people pick are narrower, suggesting that we overestimate our own ability to estimate accurately.

Obviously this is a simplified illustration, but there are [a number of studies that suggest we are overconfident when it comes to forecasting](#).

Can We Get Better? The Data Suggests It's Possible (with Feedback)

When people are presented with evidence of their overconfidence, there's data to show that individual estimators do improve.

[In one study of software development work](#), programmers gave correct estimates 64% of the time on the first ten items they estimated.

When provided with that feedback, they improved to 70% correct on the second set of ten items. And then to 81% on the third set, after additional feedback on their accuracy.

Getting estimators to realize that excessive confidence in their own estimating abilities is misplaced is helpful to encourage a collaborative approach to estimating.

If someone is absolutely convinced of the infallibility of their own estimates, that person won't engage constructively in debates about the right estimate to provide.

Want More Help With Story Points? (Coming Soon)

Very soon I'll be releasing some free video training to help solve some common story points problems. If you want to be the first to find out when it's available, register now to join the wait list.

What Do You Think?

Do you think the perception that we're bad at estimating is unfair? Have you and your team been able to improve estimates as you work together? How did you do it? Let me know in the comments.

[Get my chapters now!](#)

Posted: October 6, 2020

Tagged: [estimating](#), [story points](#)

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

[What Are Agile Story Points?](#)

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments

[Read](#)

[Don't Equate Story Points to Hours](#)

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • [39 Comments](#)

[Read](#)

[For Better Agile Planning, Be Collaborative](#)

The best plans are created by developers and stakeholders working together.

Jul 12, 2022

[Read](#)

[Four Reasons Agile Teams Estimate Product Backlog Items](#)

Estimating product backlog items provides benefits beyond predicting when a project will be finished.

May 17, 2022

[Read](#)

[Automatically Triangulating Estimates in Planning Poker](#)

Comparing estimates during Planning Poker just got easier.

Nov 23, 2021

[Read](#)

Estimating with Story Points Course Available for One Week

Registrations are now open to Estimating with Story Points.

Nov 17, 2021

[Read](#)

Learn About Agile

New to Agile and Scrum?

Scrum

User Stories

Planning Poker

Agile Software Development

Agile Project Management

Agile Presentations

Mountain Goat on YouTube

Agile Mentors Podcast

Elements of Agile Assessment

Agile & Scrum Training

Online Certifications

Video Courses

In-Person Certifications

On-Site Courses

Training Schedule

Compare Courses

PDUs and SEUs

More...

About Us

Contact Us

Our Students

Blog

Books by Mike Cohn

Agile Tools

PDUs and SEUs

by
Mike Cohn
58 Comments
Tagged:
estimating, transitioning to agile, resistance

Are Estimates Ever Helpful to Developers?

We know that stakeholders and clients want estimates.

They benefit from being able to plan, prioritize, and predict the dates their product will be delivered.

But for a developer, the benefits are less clear. In fact, it can feel like there is no benefit at all if estimates are interpreted as a promise and then constantly used against you.

So it's no surprise that developers are often reluctant to estimate. If doing something always seems to cause you trouble, why keep doing it? Isn't that the definition of insanity?

I understand and sympathize with this way of thinking, but the truth is that estimates can benefit teams. In fact, over time they can elevate your status in the eyes of stakeholders and give you more bargaining power than you might realize.

But how?

My Experience of Being Held Accountable to Estimates

My background is very much as a hardcore programmer, first in C, then C++, then Java. When I began programming, being asked to estimate made me nervous because I knew every estimate I ever gave could possibly be used against me.

Not only that, but we were rarely celebrated for achieving an estimate. There were no rewards or bonuses if I finished coding something within my estimate. I'd merely done what I'd said I'd do, fulfilled a basic expectation. No one really noticed.

But if I finished late? People noticed, and noticed enough so that sometimes I'd get yelled at.

This early experience stayed with me as I progressed into a management role. If managers were supposed to get mad at developers who finished late, that's what I'd do. This was the only management style I'd seen at the time. It didn't feel right, but it was all I knew.

I quickly learned that holding team members accountable for bad estimates didn't work.

The developers working for me were—just as I had been—chronically optimistic, and would tend to underestimate the amount of time it would take to complete something.

As a new manager, I thought that if I held the team to their overly optimistic estimates, they might not make them—but they'd be close. We'd finish projects faster than my boss and stakeholders expected. And I'd look good as a new manager.

Predictably, that didn't work out. When I pushed the team to reach these estimates, they soon realized there were consequences of estimating too low—they'd have to work overtime or cut corners to meet the deadline. That wasn't a fun way to work and so they learned to pad their estimates as a way to make the pace more sustainable.

It was tense, dishonesty was becoming entrenched, and I knew then I needed a better approach.

A Valuable Incentive for Teams to Estimate: Bargaining Power

As I mentioned at the start of this post, the benefits of estimating seem to be stacked in favor of stakeholders. I wanted to encourage my team to **want** to estimate instead of seeing it as an arbitrary exercise that could only get them into trouble.

By then I was managing multiple teams. Naturally some teams were better at estimating than others.

And I noticed something very interesting: The teams who had given better estimates on their past few projects had better relationships with their stakeholders. Stakeholders listened when those teams said that meeting a deadline was impossible.

What Happens When a Team Is Bad at Estimating (or Refuses to Try)

Imagine a team that refuses to estimate—they have no history of delivering on their estimates. Or imagine a team that is terrible at estimating—they have a poor track record of delivering on what they say is possible.

Their boss comes to the team, outlines a set of functionality and asks the team how long it will take, hinting that it really needs to be delivered in six months if possible.

The team estimates the work and decides that it's going to take much longer than six months. Team members explain they need more resources, more time, or can deliver in six months with fewer features. But all that functionality in six months? Just not possible.

The boss's likely response in a situation like this? "Let's just do it in six months."

Now, before we decide this boss is a jerk, remember this team either has **no history** of estimating, or a **history of poor** estimating.

The team hasn't earned the right to be heeded when they say the deadline is impossible.

The boss in this case is likely to view the team's resistance as something to overcome rather than an informed opinion based on facts.

A Good (Not Perfect) Estimating History Elevates Your Status

Now let's contrast this situation with one in which the team estimates and estimates well. I don't want to cheat and say the team estimates perfectly. They're good, not perfect.

For example, on one past project they told the boss they could deliver in April or May, and they finished just before the end of May.

On another project, team members said it would take three months. And they delivered right on the deadline but did have to postpone a couple of fairly small product backlog items.

So, they aren't perfect estimators, but they're good.

And now the boss asks this team if they can deliver a certain amount of functionality in six months. This team estimates the work, considers it, and team members ultimately tell the boss that it's too much. It can't be done and realistically they'd need eight or nine months.

Does the boss tell this team to do it anyway as with the first team?

No. This team has provided credible estimates in the past, so the boss pays attention when they say it can't be done. The boss will even treat the team as equal partners in solving their shared problem of what to do, given that the desired deadline is impossible.

The boss and team will likely discuss ways to solve their shared problem. Perhaps more people can be added to the team. Or an entire second team could be added. Perhaps one or two requirements have an outsized impact on the schedule. Perhaps it's not a problem to release the full functionality a month late as long as certain features are delivered on time.

A Healthy, Respectful Relationship

This team's history of providing good estimates leads to a healthy relationship with the boss that is based on mutual respect and trust. This allows the boss and team to explore options as equals.

It might be hard to imagine the benefits that are waiting for you if you can build a solid history of successful estimating, but they are there. Don't get me wrong: this type of relationship isn't built overnight, but I can guarantee it won't be built at all if a team refuses to estimate.

What Other Benefits Are There for Developers?

What do you think? Are there more benefits for developers if they build a credible estimating track record? Or are stakeholders the only ones who benefit? Have you tried a new approach to overcome team resistance to estimating? Let me know in the comments.

[Get my chapters now!](#)

Posted: September 22, 2020

Tagged: estimating, transitioning to agile, resistance

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the

Better User Stories video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

What Are Agile Story Points?

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

Don't Equate Story Points to Hours

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • 39 Comments [Read](#)

8 Reasons Scrum Is Hard to Learn (but Worth It)

Transitioning to Scrum is worth it, but some aspects are challenging.

Oct 11, 2022 [Read](#)

Elements of Agile: An Agile Assessment Tool for Iterative Improvements

New to agile, or needing an agile re-boot? We've got a new no-cost assessment and actionable ...

Sep 13, 2022 [Read](#)

From Project Manager to Scrum Master -3 Tips for Making the Transition

What does a good project manager need to do to become a great Scrum Master?

Aug 23, 2022 [Read](#)

For Better Agile Planning, Be Collaborative

The best plans are created by developer and stakeholders working together.

Jul 12, 2022 [Read](#)

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

by
Mike Cohn **Tagged:**
estimating, quality, productivity
22 Comments

Suppose you're working on a project that has a major deadline a year from now. You're agile, so you might release every couple of weeks if that's appropriate for your customers and the product.

But the big deadline—the one everyone is focused on—is a year away.

And, let's suppose you're on schedule. It's not going to be easy, but the general feeling is that if everyone stays focused and avoids major changes in scope then the team will deliver on time.

But the Boss Wants It Immediately

Now, what if I told you I just got an email from your boss. She said there's a massive bonus in it for you and the rest of the team if you can shorten that schedule. Finish in one month instead of twelve and there's a huge bonus.

Would the promise of a massive bonus entice you to deliver in one month what you expect to take twelve months?

Probably not. No amount of overtime or extra focus is going to help a team cut more than 90 percent from a realistic schedule.

The Boss Wants It a Little Sooner

Instead, what if your boss offered that massive bonus to reduce the twelve-month schedule down to eleven months.

For the right incentive, most of us could be enticed to shorten our lunch breaks, stay a bit more focused during the day, keep meetings to a minimum and fully productive, stay a little late occasionally, and do whatever else was needed to cut a month from a one-year schedule.

In fact, many of us work that way when we need to without the promise of a massive bonus, but simply because our organizations and leaders have tapped into our intrinsic motivation. And we want our products and organizations to succeed.

The U-Shaped Relationship Between Productivity and Pressure

What this means is that there is a U-shaped relationship between productivity and pressure. [Research](#) by professors Ning Nan and Donald Harter bears this out.

As shown in the following figure, they found that reducing estimates given by teams by up to 20% leads to a reduction in delivery time for projects.

A U-shaped relationship between pressure and schedule shows that modest deadline pressure can reduce schedule. But excessive pressure prolongs the schedule.

This research fits with my experience.

Excessive time pressure often reduces productivity and extends schedules. One reason for this is that excessive pressure is often accompanied by team members working excessive overtime. Too much schedule pressure also often leads teams to taking (or being forced to take) actions that appear to increase productivity but really worsen it. Writing "quick and dirty code" or skipping some testing are examples of

things teams are sometimes told to do to speed up development that have the opposite effect.

The first parts of the chart above also match my experience. Teams with no deadline pressure at all seem to take longer to deliver than teams with slight pressure. It benefits a team to know there's urgency to whatever they're working on.

In the face of urgency, teams usually respond by working a little faster, usually through greater focus or diligence.

But does that additional productivity from slight pressure come at the expense of quality?

The Effect of Time Pressure on Quality

The question of how time pressure affects quality was investigated by Magne Jørgensen and Dag Sjøberg. In their research, they found a significant increase in defects introduced when programmers were given a schedule that was estimated incorrectly: 40% below what it should have been based on their prior work.

This led them to theorize a relationship between pressure and quality as shown in the following image.

Estimates that are too low lead to lower quality.

On the left in this graph are estimates that are too low—the actual amount of work exceeds the estimate. This leads to reduced quality as team members rush to deliver the product. Accurate estimates lead to improved quality. An estimate that is too high, or “padded,” doesn’t lead to increased quality over that achievable with a realistic, accurate schedule.

Implications for Agile Teams

So, what do these relationships between time pressure, productivity, and quality mean for agile teams?

At a minimum, I think they illustrate one of the benefits of iterations. The short iterations or sprints of agile

mean that a team's next deadline is usually no more than a month away. I suspect that puts teams in the sweet spots shown by this research and in the two graphs.

Should Leaders Add Time Pressure?

Does this mean product owners, Scrum masters, or managers should seek to put a little artificial pressure on teams?

No, I don't think they should.

The gains in productivity and faster delivery created by time pressure are real. But they're minor. And if too much schedule pressure is applied, quality could drop dramatically, as shown in the second chart above.

To put it another way, there are minor benefits to a little schedule pressure but there are major risks to too much schedule pressure. That means leaders should be cautious of applying any additional time pressure.

The inherent pressure on a team of needing to have something fully delivered every few weeks is generally all the pressure a team needs.

What Do You Think?

I'd love to know what you think. Have you been part of a team that was expected to deliver much more quickly than team members felt realistic? Or have you been part of a team with absolutely no pressure to deliver? How were productivity and quality impacted on those projects? Please share your thoughts in the comments below.

[Get my chapters now!](#)

Posted: May 5, 2020

Tagged: estimating, quality, productivity

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the *Better User Stories* video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

[What Are Agile Story Points?](#)

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

[Don't Equate Story Points to Hours](#)

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Jul 12, 2022 [Read](#)

[For Better Agile Planning, Be Collaborative](#)

The best plans are created by developers and stakeholders working together.

Jul 12, 2022 [Read](#)

[Four Reasons Agile Teams Estimate Product Backlog Items](#)

Estimating product backlog items provides benefits beyond predicting when a project will be finished.

Jul 12, 2022 [Read](#)

[How Implementation Intentions Help My Sprints](#)

Planning an exact time to do something within a sprint helps make sure you get the important stuff ...

[Automatically Triangulating Estimates in Planning Poker](#)

Comparing estimates during Planning Poker just got easier.

Nov 22, 2024 [Read](#)

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

by
Mike Cohn **Tagged:**
estimating, planning, scaling
26 Comments

When organizations scale agile, they're faced with the challenge of managing multiple teams. And if you have multiple teams working on the same project, that coordination becomes more complex, particularly when creating estimates.

The teams will need to estimate and plan, and then track progress against the plan so the product owner can prioritize work and communicate to stakeholders when something will be delivered.

But working with multiple teams adds some additional challenges, including:

- How do you deal with teams that have different levels of skill and experience?
- Can you accurately estimate work without involving everyone on every team?
- Is it possible to create estimates ahead of time if you don't know which team will do which work?

There is a solution that I'm going to share with you, but before that let's take a look at two key mistakes most organizations make when estimating for multiple teams (and based on a discussion inside my [Agile Mentors Community](#), these mistakes are still prevalent today):

Mistake #1: Creating estimates that don't reflect the abilities of all teams

The first mistake tends to happen when a company handpicks a set of individuals to estimate the entire backlog. This in itself is not a bad idea (in fact I'm going to advocate you do this a little later). It's certainly faster and more efficient than having everyone on a large team all estimate every item. And you might think that if you can just select the right people, the result will be an insightful estimate.

But that's exactly where things tend to go wrong, because if you don't have people who are familiar with the work to be done and truly represent a cross-section of abilities throughout all teams, you will create estimates from a very narrow (and inaccurate) perspective.

Unfortunately, this does happen in companies today: estimates are created by people who are out of touch with the work to be done, and out of touch with the teams who will be doing the work.

One of the results from this is having estimates that are far more ambitious than teams can possibly achieve (especially when these estimates are used to bid for fixed-price contracts). While a low estimate might initially delight customers and stakeholders, as the project is inevitably delayed, clients become upset, trust is eroded, and working relationships break down.

What's worse, rarely will people blame the estimate. Instead it's the teams that face growing pressure and criticism as they scramble to deliver on an impossible deadline.

Mistake #2: Equating story points to hours

The second mistake is when an organization tries to create consistency and predictability by insisting that a uniform definition of story points be adopted by all teams. A popular (but incorrect) way to do this is to force all teams to use a set amount of hours for one story point, for example 1 story point as 3 hours, or 3 story points for 8 hours.

Again, I can see the logic behind this: it seems like a very simple way for management to see at a glance how teams are performing.

But, it doesn't work.

Defining a story point as equal to a number of hours eliminates [the main benefit to story points](#), which is that the number of points given to something doesn't depend on who will do the work. A mile is a mile is a mile regardless of whether it will be run by a fast or a slow runner. So you should not [equate story points to hours](#).

Worse, when a story point is defined as some number of hours for all teams, management may assume that teams can be easily compared solely on their velocity, which isn't the case.

What Should Large Projects Do Instead?

So how do you create meaningful estimates for projects that involve multiple teams?

[To estimate at scale, establish a common baseline for all teams to estimate with and to use to track progress toward shared goals.](#)

But you need to do this in a way that is representative of **all** teams rather than simply forcing people to accept an estimate.

Here's how.

Get the Right People Together

The best way to establish a common baseline across multiple teams is to bring together one or more representatives from each team. How many people will vary depending on the size and number of teams on the project. If you only have two or three teams, consider having everyone participate, but for larger numbers of teams it's more likely to be two or three people per team.

Teams should select whomever they think can best estimate. Usually this will be more senior members, but that isn't always the case. Teams should also consider mixing up the skills their representatives have. For example, don't send three great JavaScript developers if only a portion of the work is JavaScript.

Estimate a Variety of Backlog Items

Those representatives assemble to play [Planning Poker](#), ideally in-person but perhaps virtually if some representatives are remote, as is likely. This group is not going to estimate the entire product backlog for what is likely a large project. No, they're going to estimate only 10–20 items.

That means that when this meeting is over, everyone will leave with a list of 10–20 items with estimates everyone has agreed on. And each team will have one or more people who participated in creating those estimates and can share any nuances or assumptions that were made.

It's important however that these people estimate a *variety* of backlog tasks. That will make it easier when teams are estimating using this baseline of estimates for the basis of their relative estimates. You don't want to estimate 20 very similar product backlog items because that won't help when other teams use this reference to create estimates for backlog items that are totally different.

Know that You Don't Need Everyone to Relate to Every Item

The approximately ten to twenty product backlog items I'm recommending be estimated do not each need to be understandable by every representative. For example, a hardcore scientific calculation may be representative of the overall work on this multi-team endeavor. But a couple of estimators in the room can't relate to that item.

That's fine. Not every estimator needs to be able to relate to every item. Instead the goal is that most items should be understandable by most teams.

This means selecting the approximately ten to twenty items is the responsibility of those participating in the meeting. They'll be in the best position to judge when the items they've chosen cover a representative breadth of the product and whether most estimators can participate in estimating most items.

Using a baseline created this way, with estimates agreed upon by representatives of each team, will enable

all teams to estimate in a consistent manner.

Have You Estimated With Multiple Teams? I Want to Hear from You

I know from the discussions inside the [Agile Mentors Community](#), that this is still a hot topic today so I'd love to know what you think. Have you estimated for multiple teams? What was your approach? What worked and what didn't? Share your experience in the comments below.

[Get my chapters now!](#)

Posted: March 3, 2020

Tagged: estimating, planning, scaling

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

[What Are Agile Story Points?](#)

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

[Don't Equate Story Points to Hours](#)

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • 39 Comments [Read](#)

[For Better Agile Planning, Be Collaborative](#)

The best plans are created by developers and stakeholders working together.

Jul 12, 2022 [Read](#)

[Four Reasons Agile Teams Estimate Product Backlog Items](#)

Estimating product backlog items provides benefits beyond predicting when a project will be finished.

May 17, 2022 [Read](#)

[How Implementation Intentions Help My Sprints](#)

Planning an exact time to do something within a sprint helps make sure you get the important stuff ...

Mar 22, 2022 [Read](#)

[Automatically Triangulating Estimates in Planning Poker](#)

Comparing estimates during Planning Poker just got easier.

Nov 23, 2021 [Read](#)

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

by
Mike Cohn
31 Comments
Tagged:
estimating, sprint planning, planning poker, fibonacci

If you've estimated with Planning Poker, you may very well have used cards with either the Fibonacci sequence, or a modified Fibonacci sequence.

The traditional Fibonacci sequence is 1, 2, 3, 5, 8, 13, 21 and so on, with each number the sum of the preceding numbers.

Years ago I began having teams estimate with a modified Fibonacci sequence of 1, 2, 3, 5, 8, 13, 20, 40 and 100.

Why?

It's because numbers that are too close to one another are impossible to distinguish as estimates.

Weber's Law

Imagine being handed two weights—one is one kilogram (2.2 pounds) and the other is two kilograms (4.4 pounds). With one in each hand but not able to see which is which, you can probably distinguish them. The two kg weight will feel noticeably heavier.

Imagine instead being handed a 20kg weight and a 21kg weight. They are the same one kg difference as the one and two kg weights. But you would have a much harder time identifying the heavier of the two weights.

This is due to Weber's Law. Weber's Law states that the difference we can identify between objects is given by a percentage.

Weighing the Difference Between Objects

The difference from one to two kilograms is 100%. You can probably distinguish the weight of items that differ by 100%.

The difference between 20 and 21kg, however, is only 5%. You probably can't tell the difference. (I know I can't.) And if you could, it would mean you should be able to distinguish between a 1.00 kg weight and a 1.05 kg weight, as that would also be 5%.

The values in the Fibonacci sequence work well because they roughly correspond to Weber's Law. After the two (which is 100% bigger than one), each number is about 60% larger than the preceding value.

According to Weber's Law, if we can distinguish a 60% difference in effort between two estimates, we can distinguish that same percentage difference between other estimates.

So, the Fibonacci values work well because they increase by about the same proportion each time.

Modifying the Fibonacci Sequence

Early agile teams I worked with made use of this and estimated with the real Fibonacci sequence. Ultimately, though, we learned that an estimate of 21 implied a precision we couldn't support. Stakeholders would look at the 21 and be impressed that we called it 21 rather than rounding it to 20 or even 25.

This led us to start using 20 rather than 21. Once we'd deviated from the Fibonacci sequence once, we felt free to do so further.

That led to experimentation in which we introduced 40 and 100. Those worked well because they represented 100% and 150% increases over the preceding numbers. This was much larger than the 62% of the Fibonacci sequence.

This wasn't a violation of Weber's Law because Weber's Law has been shown not to hold at extreme values. Estimates such as 40 and 100 could, perhaps, be thought of as extreme values.

Experimenting with Planning Poker Sequences

Up until 2007, teams I worked with experimented with both the modified Fibonacci sequence and a simple doubling of numbers—1, 2, 4, 8, 16, 32.

Each worked equally well. Most teams had a strong preference for one or the other, but I could find no clear evidence that either sequence was better than the other.

But in 2007, we began printing [Planning Poker Cards](#), which we sold at cost, distributed at various agile events, and that I use in some in-person courses.

To keep printing costs down, I had to choose between these two sequences. At the time, I just *slightly* favored the modified Fibonacci sequence. And so I made the call to go with that. Once that sequence began to appear on thousands of decks a year, the sequence gained popularity at the expense of simply doubling the values. My slight preference for 1, 2, 3, 5, 8, 13... over 1, 2, 4, 8, 16... came from being in a lot of meetings with the latter sequence in which discussions were always "Is this product backlog item double the size of the other one?"

All discussions were about whether something was double, quadruple, etc. That concerned me slightly because I didn't see it with teams using the modified Fibonacci sequence. Their discussions were not always "Is this one 60% more effort?" Those seemed like healthier discussions, even though I had no way of measuring that.

These days I remain fairly impartial between the two sets of values. I think teams can estimate successfully with either set of values as each exhibits attributes of Weber's Law.

What Do You Think?

What's your experience with different estimating sequences? Which values do you use and do they work for your team? Please share your thoughts in the comments section below.

[Get my chapters now!](#)

Posted: September 10, 2019

Tagged: estimating, sprint planning, planning poker, fibonacci

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

[Why I Don't Emphasize Sprint Goals](#)

Sprint goals are considered a mandatory part of Scrum. Here's why I disagree.

Mar 21, 2023

[What Are Agile Story Points?](#)

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments

[Read](#)

[Don't Equate Story Points to Hours](#)

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • 39 Comments

[Read](#)

[For Better Agile Planning, Be Collaborative](#)

The best plans are created by developers and stakeholders working together.

Jul 12, 2022

[Read](#)

[3 Ways to Help Agile Teams Plan Despite Uncertainty](#)

We might not like ambiguity, but it's a fact of life. Find out how to plan with uncertainty in mind.

Jun 21, 2022

[Read](#)

[Four Reasons Agile Teams Estimate Product Backlog Item](#)

Estimating product backlog items provides benefits beyond predicting when a project will be finished.

May 17, 2022

[Read](#)

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile Assessment		

by
Mike Cohn Tagged:
23 product backlog, estimating, story points
Comments

When estimating with story points, most teams use a predefined set of values that doesn't include every possible number. For example, teams commonly use powers of two (1, 2, 4, 8, 16,...) or a Fibonacci sequence (1, 2, 3, 5, 8, 13, ...)

By intentionally leaving some numbers out of the set of acceptable estimates, teams avoid bogging down in discussions of, for example, 15 versus 16. Estimating to that level of precision would be extremely difficult and time-consuming, and most likely not even possible.

One of the questions I often get about story points is what to do if you can't decide whether a particular product backlog item should be an 8 or a 13 (or a 3 and a 5). Part of the answer can be found by thinking about water buckets.

Suppose you have 10 liters of water you need to store. You also have an 8-liter bucket and a 13-liter bucket.

Which bucket would you store the water in?

The 13-liter bucket, right? Ten liters of water doesn't fit in an 8-liter bucket. The water would overflow and spill out.

Extrapolating further, you'd use the 13-liter bucket for all amounts of water from 9 liters through 13 liters.

Once you hit 14 liters, though, you'd once again need a bigger bucket.

And so it is with the values you choose to use when estimating stories. Think of each value as a bucket. A value bucket is used for all stories between that value and the next lower value.

For example, when playing [Planning Poker](#) many teams will use a modified Fibonacci sequence of 1, 2, 3, 5, 8, 13, 20, 40 and 100. The 13-point card should be used for any story the team estimates larger than 8 and no larger than 13.

That is, each story point value is implicitly a range--just like a bucket can hold a range of amounts of water. The trick is to choose the right bucket.

Choose the Right Bucket for Each Product Backlog Item

For accurate planning and estimating, teams need to understand that their job is not to put an estimate on a product backlog item. Their job is to put the story in the right bucket. Yes, I know that in both cases someone grabs a pen and writes a number on a story card. But the goal is to find a number (a bucket) just large enough to hold the whole story.

When estimating, ask the team to picture a set of buckets in front of them labeled 1, 2, 3, 5, 8, 13 and so on (or whatever sequence it uses). And then ask them to picture themselves throwing story cards into the buckets.

This helps team members move away from the feeling that estimates need to be perfect and precise. They don't. Product backlog items merely need to be "thrown" into the right bucket.

Why Rounding Up Some Estimates Is Good

You might be concerned that this rounding up of estimates could lead to inflated or padded schedules. Let me explain why that usually won't happen.

Consider a product backlog item that you believe should be 10 points. But you're using the modified Fibonacci sequence I listed above and are following the rounding up strategy I recommend. And so, the bucket you "throw" the product backlog item into will be 13.

Next, let's assume that at 13 points, the item is too big to bring into a sprint. And so the team splits it into smaller stories. Perhaps they split it into three smaller stories, which they estimate as 5, 5 and 2 points.

Notice what happened there with the numbers.

The three small stories add up to 12 points. That's one less than the 13-point estimate you actually put on the larger story. But it's two more than the 10 you really thought was the right estimate for the initial large story.

By forcing the estimate into the next larger bucket, you've improved the likelihood of accurately predicting the delivery date of this project.

If you had rounded down to the next lower estimate (8), the project would be 4 points late. If you had used your actual estimate (10), the project would be 2 points late.

Instead, by using the estimate values as buckets and rounding up, you've made it more likely you'll deliver this project on time.

But couldn't the 10-point story have turned out to really be 8 points? Sure. But it could also have been 14 or 15 or any other number.

Ranges & Rounding Up Are Good Tools for Estimating

Rounding up estimates doesn't lead to padding or inflation--a topic I cover in another blog post, "[How to Prevent Estimate Inflation](#)." Instead, rounding and ranges are a reflection of the uncertainty in estimates. Thinking of story points as a range of capacity-limited buckets reflects this uncertainty and will help create more accurate plans.

[Download my PDF](#)

Posted: July 16, 2019

Tagged: product backlog, estimating, story points

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding

member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

[What Are Agile Story Points?](#)

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

[Don't Equate Story Points to Hours](#)

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • 39 Comments [Read](#)

[Epics, Features and User Stories](#)

I've been getting more and more emails lately from people confused about the difference between ...

Nov 08, 2022 • 93 Comments [Read](#)

[For Better Agile Planning, Be Collaborative](#)

The best plans are created by developers and stakeholders working together.

Jul 12, 2022 [Read](#)

[Relationship between Definition of Done and Conditions of Satisfaction](#)

I'd like to clarify the relationship between two important concepts: a team's Definition of Done ...

Jun 28, 2022 • 38 Comments [Read](#)

[Four Reasons Agile Teams Estimate Product Backlog Item](#)

Estimating product backlog items provides benefits beyond predicting when a project will be finished.

May 17, 2022 [Read](#)

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

by
Mike Cohn
Tagged:
estimating, velocity, planning
22 Comments

Velocity can be a very useful predictor of how much work an agile team will complete in the future. It can be especially helpful when looking forward at least four or five iterations.

This is because over the short term, velocity can vary from iteration to iteration. But, velocity is more stable over the long-term as the law of large numbers kicks in and errors in individual product backlog items are balanced out.

This means velocity can work as a reliable predictor as it allows teams to state things like, “In the next 5 sprints we’ll probably finish between 100 and 120 story points.”

Such statements are based on a team knowing its velocity (perhaps as a single, averaged value but even better as a range).

But how should a team estimate its future velocity when team members are taking time off, whether personally or because of a company holiday?

There are a couple of different approaches to this. And each is appropriate in different contexts.

Approach 1: Ignore It

When adjusting velocity for holidays, vacation or personal time, a simple but valid strategy may apply: Ignore it.

Ignoring the impact of various forms of time off can be a valid approach when planning over a sufficiently long enough period. The idea is that *team members will probably take time off in the future at the same rate they took time off in the past.*

If that seems true for the period for which you are planning, you don't need to make any adjustments to velocity at all.

An Example

As a trivial example, suppose you are planning a project for all of next year. (I don't recommend you commit to a fixed date and scope that far out, but it makes a good example.)

In this case, a good starting point will be total velocity for all of this year. Any times of reduced productivity (such as Christmas or summer vacation time) will balance out over a long term.

Approach 2: Adjust Proportionately to the Full Team

In some cases, however, you do want to adjust for holidays and team member vacations. You should do this if you're in a situation such as:

the project is shorter (perhaps 3-6 months) and you don't think time off during that period will be consistent with the past

You know that one or more team members will be out for a longer than normal period such as a sabbatical, birth of a new child, or just an extended vacation.

One or more holidays with greater than normal impact occur during the period

The Simplest Approach

The simplest approach is to assume that that everyone on a team can do everything. This is probably not true but it can be a starting point in deciding how to adjust velocity.

When making this assumption, the formula for predicted velocity in a sprint is:

$$\text{Predicted Velocity} = \text{Average Velocity} \times (\text{Planned Working Days} \div \text{Average Working Days})$$

An Example

As an example, suppose a six-person team has an average velocity of 40. They are running two-week iterations. In the coming iteration, one team member will be out the full two weeks, and the rest of the team will be gone for one day for a national holiday.

This team normally has 60 working days in an iteration (6 people \times 10 days). In the coming iteration they will lose 15 days to time off. (One person for ten days; the other five for a day each.) This leaves them 45 working days in the coming iteration.

This means their predicted velocity will be 30. This is determined by taking their historical average

An Improvement

In this example, I have used 60 days as the team's average number of working days. For a six-person team running two-week iterations, 60 is very unlikely as it assumes no one is ever sick.

velocity (40) and multiplying it by the percentage of days planned to be worked ($45 \div 60 = 0.75$). So

$$40 \times (45 \div 60) = 30$$

You should begin tracking the actual number of working days per iteration and use that value rather than the theoretical maximum number of days.

Approach 3: Adjust Proportionately Based on Skill

When team members are not highly interchangeable, a better approach can be to reduce velocity by the same percentage that a person's absence represents within their primary skill.

For example, if one of two database administrators on a team will be gone for the entire sprint and no one else can do that work, reduce velocity by 50% since total DBA capacity is down that much.

For most roles this will be too dramatic of a reduction. The best approach I've found is to use somewhere between this amount and the amount given by assuming everyone is equal.

Combining Adjustments

Let's return to the case of one of two DBAs being gone for an entire iteration. And let's assume the team is made up of six individuals running two-week (ten-day) iterations.

In this case, one DBA being gone represents a 50% reduction if the amount of work the DBAs will finish ($1 \div 2$). It also represents 17% less total capacity in the sprint ($1 \div 6$).

So, one DBA being out will likely reduce velocity between 17% and 50%. Choose a specific amount for your project based on whether DBA work is on or near the critical path and how interchangeable others are at helping with the DBA's work.

What to Do When Multiple People Will Be Absent

If multiple people representing different skills will be absent during a coming sprint, reduce velocity only by the absence with the largest impact.

Think about it with this extreme example: Suppose 90% of your programming capacity will be gone if the coming sprint. And suppose 10% of your testing capacity will also be gone. The fact that you have slightly lower testing capacity won't matter in the sprint. The total work done in this case will be determined by the much larger reduction in programming capacity.

An Example

As an example, let's assume we have an 8-person team with two DBAs and three full-stack developers. Of these, one DBA and one developer will be absent for the entire sprint.

In this case, the DBAs will lose 50% of their normal capacity ($1 \div 2$) and the developers will lose 33% of theirs ($1 \div 3$).

Treat those as upper limits on likely loss and then reduce velocity only by the larger of the two.

Will You Help Me by Sharing Your Biggest Estimating Challenges?

Reliably planning an agile project is completely feasible. And for many teams it's necessary. Perhaps someday those teams can operate in a "you'll get it when you get it" environment.

But today many teams need to create and communicate plans with the expectation that the plans are sufficiently accurate for good decisions to be made.

Right now, I'm working on something that deals with the problems of creating estimates particularly for fixed-date, fixed-scope, fixed-price (or even fixed-everything!) projects.

Have you ever struggled with estimating in these situations?

If so, I'd love to hear from you.

Would you click the link below and answer some quick questions?

[Get my chapters now!](#)

Posted: November 27, 2018

Tagged: estimating , velocity , planning

[About the Author](#)

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

[What Are Agile Story Points?](#)

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments

[Read](#)

Don't Equate Story Points to Hours

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • [39 Comments](#)

[Read](#)

For Better Agile Planning, Be Collaborative

The best plans are created by developers and stakeholders working together.

Jul 12, 2022

[Read](#)

Four Reasons Agile Teams Estimate Product Backlog Items

Estimating product backlog items provides benefits beyond predicting when a project will be finished.

May 17, 2022

[Read](#)

How Implementation Intentions Help My Sprints

Planning an exact time to do something within a sprint helps make sure you get the important stuff ...

Mar 22, 2022

[Read](#)

Automatically Triangulating Estimates in Planning Poker

Comparing estimates during Planning Poker just got easier.

Nov 23, 2021

[Read](#)

Learn About Agile

New to Agile and Scrum?

Scrum

User Stories

Planning Poker

Agile Software Development

Agile Project Management

Agile Presentations

Mountain Goat on YouTube

Agile Mentors Podcast

Elements of Agile Assessment

Agile & Scrum Training

Online Certifications

Video Courses

In-Person Certifications

On-Site Courses

Training Schedule

Compare Courses

PDUs and SEUs

More...

About Us

Contact Us

Our Students

Blog

Books by Mike Cohn

Agile Tools

by

Mike **Tagged:**

Cohn product backlog, estimating, user stories, product owner,
14 transitioning to agile, planning, leadership, ceremonies

Comments

One of the biggest questions I get about agile is:

Where do I go from here?

Succeeding with agile isn't just about knowing where to start, it's about knowing where to go next—wherever you are on your agile journey.

Whether you're completely new, transitioning to agile, or moving into a leadership role there's a plethora of information on these topics. But it's not easy to wade through it all to find the resources most relevant to your unique agile journey.

If it were possible, I'd love to sit down with you, listen to where you are now, and then give you my personal recommendations about what steps to take next. But with all the traveling, training, and coaching I do, I can't do that right now.

So perhaps I can do the next best thing?

Your (Free) Personalized Agile Guide

Instead of you sifting through thousands of agile articles and resources, I've handpicked some recommendations just for you.

Simply take 60 seconds to complete a short series of assessment questions and you'll instantly get a customized content plan to help you wherever you are in your agile experience.

Simply rate your proficiency on the following subjects:

- Agile Ceremonies
- User Stories
- The Product Backlog
- Estimating
- Planning & Metrics
- The Effective Scrum Master
- The Effective Product Owner
- The Effective Agile Leader
- Agile Transformations

After that, you'll get your custom list of content—the agile articles and resources I believe will help you succeed with agile.

It's like a personal reading list from me to you.

Get your guide in seconds then track your progress

The Personalized Guide to Agile is designed to make life easier not harder, and the last thing we want to do is add to the deluge.

That's why you can store your personalized guide for free and track your progress. You can see at a glance which resources you've worked through and which ones you'd like to save for later.

What are your next steps?

Everyone is on a unique agile journey and I'd love to know more about where you are now. Let me know in the comments where you are today and what you have planned for your next steps in agile.

[Get your custom results now](#)

Posted: November 13, 2018

Tagged: product backlog, estimating, user stories, product owner, transitioning to agile, planning, leadership, ceremonies

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

User Stories: How to Create Story Maps

Story maps help to create a shared understanding of the product, visualize user needs, and elicit ...

 Mar 14, 2023 [Read](#)

How to Run a Successful User Story Writing Workshop

What you need to know to conduct a story-writing workshop with your team.

Feb 28, 2023 • 3 Comments [Read](#)

Be a Great Product Owner: Six Things Teams and Scrum Masters Need

Learn six ways effective product owners ensure their teams' success.

Jan 31, 2023 • 16 Comments [Read](#)

What Happens When During a Sprint

Succeeding with Scrum is easier when you know when and why to conduct each of the Scrum events ...

Jan 03, 2023 • 34 Comments [Read](#)

What Are Agile Story Points?

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

Don't Equate Story Points to Hours

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • 39 Comments [Re](#)

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

by
Mike Cohn **Tagged:**
50 estimating, planning
[Comments](#)

It is very common for agile teams, especially Scrum teams, to estimate both their product backlog and sprint backlogs.
In this article, I'll address:

- Why estimating both the product backlog and sprint backlog can be useful even though it seems redundant
- Why teams should estimate the two backlogs in different units
- When teams should estimate
- Whether all teams should estimate

If you're new to agile or need a quick reminder about what the product and sprint backlogs are, you can watch these two videos on the [product backlog](#) and [sprint backlog](#).

Why You Should Estimate both the Product Backlog and Sprint Backlog

It's useful to estimate both the product backlog and the sprint backlog because the estimates are used for different reasons.

Reasons to Estimate the Product Backlog

There are three main reasons to estimate a product backlog. First, it allows a team and its product owner to make longer term predictions about how much can be delivered by when. It allows teams to answer questions like:

- How much can you deliver in three months?
- When can a certain set of product backlog items be delivered?

Second, it aids product owners in making prioritization decisions. Priorities should be set based on the expected benefits and costs of the product backlog items. A product backlog item estimated at 3 story points, days, or whatever units you use, is typically a higher priority than it would be if it were estimated at 100.

To say this isn't the case would mean that you always order the best wine on the wine list or drive the best car manufactured regardless of price. Chateau Mouton-Rothschild 1945, anyone?

Most of us, including product owners on agile projects, cannot afford to make decisions that way. In prioritizing work, we consider the cost of developing product backlog items. For most items, the estimate of the effort involved is the biggest component of the cost.

A third reason to estimate items on the product backlog is that team members become more knowledgeable about the item by thinking about it enough to estimate it. This means there will be fewer surprises when the feature is being developed.

Reasons to Estimate the Sprint Backlog

Now let's look at why teams should also estimate the sprint backlog. There are two reasons to estimate the sprint backlog. First is that it helps the team determine how much work to bring into the sprint.

By splitting product backlog items into small, discrete tasks and then roughly estimating them during sprint planning, the team is better able to assess the workload. This increases the likelihood of the team finishing all they say they will.

Second, identifying tasks and estimating them during sprint planning helps team members better coordinate their work. For example, if sprint backlog items are not estimated, a team might not notice a critical path through the work or that the designer will be the busiest during the coming iteration.

Why Estimate In Different Units

Because the estimates on a Scrum team's two different backlogs serve different purposes, they should be made in different units.

For product backlog items, in particular, it is vital that the team can estimate quickly.

To see why, suppose a boss asks the team to estimate when forty product backlog items in the form of [user stories](#) can be delivered.

This could be an entirely valid request. Perhaps the boss wants to know whether to hire additional team members if the project will take too long. Or perhaps the boss only wants to start the project if it can be reasonably expected by a specific date.

If the team were to answer the boss by splitting each user story into tasks, as is commonly done in sprint planning, and estimating each of those, the time spent estimating would be huge.

If we assume an average of 15 minutes discussion and estimating per user story (as is commonly needed during sprint planning), estimating 40 user stories would take 600 minutes or 10 hours of whole-team effort.

If the team can instead use higher-level but equally accurate estimates on the product backlog items themselves, those estimates can usually be created much more quickly. I advise teams to target three to four minutes on average per product backlog item. In this case, estimating 40 user stories would take no more than 160 minutes, or about 2½ hours.

The best way to do this is for a team to estimate its product backlog items in [story points](#) and its sprint backlog tasks in hours.

This works well because story points are a more abstract measure that individuals with different strengths can agree on. Just as you and I can agree on the length of the measure “one foot,” even though our individual feet are most likely different lengths, so can agile team members of different skills agree that this user story will take twice as long to do as that user story.

Story points don’t work, though, at the sprint level. During sprint planning, remember the goal is for a team to determine how much work to bring into the sprint. That’s hard with abstract units like story points. It’s far easier with hours.

Estimating in hours is feasible on a sprint backlog because sprints typically contain fewer items than the entire product backlog, which means it won’t take as long. Plus, a typical sprint task will be performed by one person. And in many cases, it’s clear which person will do it. These factors make it feasible to estimate a sprint backlog in hours.

When to Estimate the Product Backlog

It’s clear that sprint backlog items should be estimated as part of a [sprint planning meeting](#) when the sprint backlog is created. But when should a team estimate its product backlog items?

I recommend estimating product backlog items at two different times. First, estimate a day or two after holding a story-writing workshop.

This is a meeting I recommend product owners conduct for their teams on a (roughly) quarterly basis. The goal is to identify the user stories needed to achieve some larger-than-a-sprint initiative. Identifying those product backlog items could take 2-4 hours (per quarter). Estimating them should then take an hour or two more.

The second time a team should estimate product backlog items is once per sprint, if new product backlog items have been added since the previous sprint. This can happen any time but it should be relatively late in the sprint to minimize the chance of new stories coming in afterwards. Most commonly this is done during a team’s product backlog refinement meeting or immediately following a daily scrum when everyone is already interrupted and present.

Why Not Estimate Product Backlog Items During Sprint Planning?

It may seem like a good idea to estimate product backlog items right at the start of the sprint planning meeting. However, there are two big problems with this.

First, it's too late for the product owner to consider the estimate when prioritizing.

Remember that one of the reasons why teams estimate their product backlog items at all is so that the product owner can prioritize. If the product owner isn't given estimates until the start of sprint planning, it's not realistic to assume the product owner will fully consider those when prioritizing.

Second, teams that estimate product backlog items at the start of their sprint planning meetings tend to spend much longer estimating.

I suspect this is because team members are about to perform more detailed sprint planning. With their minds on that, the need for more detail often creeps into the effort to estimate the product backlog, making it take longer than my target of 3-4 minutes per item.

For these reasons, try to estimate any new product backlog items that need to be estimated outside of sprint planning and also late enough in the sprint that most (if not all) new user stories have already been identified.

Should All Teams Estimate

I've established that there are good reasons to estimate both the product backlog and the sprint backlog. And I've argued that these estimates should be in different units (story points and hours) and should be estimated at different times.

But do these arguments apply to every agile team? Or are there some teams who don't need to estimate either or both backlogs?

I'll share my thoughts on that in my weekly tip on Thursday. If you haven't already, you can [sign up to receive a short tip](#) on succeeding with agile from me each Thursday.

What's Your Experience?

How does your team estimate its product and sprint backlogs? Do you use the same units? When do you estimate? Please share your experience in the comments below.

[Get your weekly tips!](#)

Posted: September 11, 2018

Tagged: estimating , planning

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of User Stories Applied for Agile Software Development, Agile Estimating and Planning, and Succeeding with Agile as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

[What Are Agile Story Points?](#)

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments

[Read](#)

[Don't Equate Story Points to Hours](#)

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • [39 Comments](#)

[Read](#)

[For Better Agile Planning, Be Collaborative](#)

The best plans are created by developers and stakeholders working together.

Jul 12, 2022

[Read](#)

Four Reasons Agile Teams Estimate Product Backlog Items

Estimating product backlog items provides benefits beyond predicting when a project will be finished.

May 17, 2022

[Read](#)

How Implementation Intentions Help My Sprints

Planning an exact time to do something within a sprint helps make sure you get the important stuff ...

Mar 22, 2022

[Read](#)

Automatically Triangulating Estimates in Planning Poker

Comparing estimates during Planning Poker just got easier.

Nov 23, 2021

[Read](#)

Learn About Agile

New to Agile and Scrum?

Scrum

User Stories

Planning Poker

Agile Software Development

Agile Project Management

Agile Presentations

Mountain Goat on YouTube

Agile Mentors Podcast

Elements of Agile Assessment

Agile & Scrum Training

Online Certifications

Video Courses

In-Person Certifications

On-Site Courses

Training Schedule

Compare Courses

PDUs and SEUs

More...

About Us

Contact Us

Our Students

Blog

Books by Mike Cohn

Agile Tools

by
Mike Cohn
Tagged:
estimating, product owner, meetings, daily scrum, scrum master
38 Comments

Before becoming a Scrum Master, I had worked as the technical lead on a number of teams. Part of my job on those teams was to make decisions. And I think I did it well. Being decisive and assertive is part of my personality.

But those personality traits didn't serve me as well once I became a Scrum Master. I realized that to succeed as a Scrum Master, I needed to shift from making assertions toward asking more questions. Because that wasn't my natural style--and it wasn't what had earned me any success I'd had through that point in my career--I struggled at first.

But with persistence I got better at asking questions. I'd like to share with you some of my favorite questions to ask. Most of these can be asked of a team whether you are a Scrum Master or a product owner.

Two Questions about Estimates

I often need a really rough estimate from a team. I'm not going to hold them to it. (I'm not asking for a commitment. [Estimating and committing](#) are not the same thing.) I really just want a rough estimate. What I find to work really well in this case is to ask:

I'm not looking for an estimate. But if I asked for an estimate, what unit pops into your minds: Hours,

days, weeks, months, or years?

Yes, I know those units overlap--many weeks can be more than a month. But getting an estimate from a team like, "Oh, weeks, a few weeks," is often good enough to make a decision, including perhaps the decision to ask the team to formally estimate the work in a more thorough way.

In situations where I do have a formal estimate from a team, another question I'll often ask is:

How confident are you in that estimate?

What you're looking for here is both the degree of confidence and whether team members agree. An estimate in which most people are 90% confident will be more likely to be accurate than one where confidence levels are all over the place and tend to be lower.

Disagreement among team members in their confidence in an estimate may also indicate the team rushed to create the estimate. That may be fine but consider the estimate to be less reliable.

Three Questions About Team Decisions

As a Scrum Master or a product owner, I sometimes want to get a sense of how thoroughly a team has thought through a decision. Here are three questions I often ask:

What are three other options you considered before making this decision?

What's the worst thing that could happen if we pursue this direction?

What has to go right for this to be the best decision?

You probably don't want to ask all three of these questions. And don't ask the same questions about every decision a team has made.

Also, you aren't asking these questions because you (as a Scrum Master or product owner) have the right to overrule the team's decision. But, you do have the right to understand how confident the team is in a decision and how aligned they are around the decision.

These questions are designed to uncover disagreement. If you ask "What has to go right for this to be the best decision?" and someone says, "Everything!" that may indicate a problem.

Two Questions about Meetings

I really dislike meetings. If I were dropped in a corridor with snakes at one end and a meeting at the other, I don't know which direction I'd run.

So I try to be diligent in keeping meetings--and the number of participants in a meeting--to a minimum. So there are two questions I often ask at the start of a meeting:

Do we need everyone who is here now?

Should anyone else be here?

The first question is designed to see if we can get by without one or two people. I often see agile teams go

too far in pursuit of teamwork and collaboration. Team members will feel like they need to be in every meeting, even ones that are completely irrelevant to them. This is the JavaScript developer who attends a meeting on whether the latest release from the database vendor is worth upgrading to.

If people on your team are being overzealous in their meeting attendance, thank them for their commitment to working collaboratively but assure them they don't need to attend every meeting. Establish a team norm that team members should not participate in a meeting if they won't add or receive sufficient value.

(And yes, this can be abused. You'll have to tell some people this does not mean they can opt of every meeting. Ultimately, the team as a whole should have the right to overrule one person's desire not to attend a meeting.)

And, along those lines, the second question above will help identify if someone absent from the meeting should be there. Yes, as much as I hate meetings (I'd probably run toward the snakes), sometimes we need to include more people.

Err or the side of too few meetings and too few people, but some meetings are worth having. And those meetings are made more valuable by having the right participants.

One Question to Ask When Wandering Around

Especially when working as a Scrum Master, I spend a lot of time dropping in on conversations. It's what is traditionally known as *management by wandering around*. For example, if I see a programmer and tester having what seems to be an important conversation, I might walk over and listen in case I can help with anything.

(Obviously don't do this every time they talk or if it looks like a private conversation.)

Sometimes the discussions I hear might be valuable to someone else. For example, perhaps I believe a tech writer should know about whatever the programmer and tester decided. So I'll ask:

Does anyone else need to know about this?

If so, I'll offer to be the one to go share the information if I can. If not, I'll offer to go get the person so they can be included right away.

One Question to Ask During Daily Standups

During a daily scrum, I'll often look at a team's [sprint burndown chart](#) and wonder how they're going to finish everything by the planned end of the sprint. But when I ask this same team if they're going to finish everything, the response is usually that they will.

If I don't agree that their prediction is realistic, I'll look back at the burndown and ask:

What do you know that I don't know?

I might get an answer that one team member hasn't updated their hours in the tool they're using. Or someone will explain that while they're currently behind, they learned a lot and things are just about to speed

up. (I find this belief to rarely hold true, but I hear it a lot.)

Asking a team what they know that I don't creates a great opportunity to synchronize assumptions. Perhaps they are assuming things will speed up and I'm not. It's a great question for uncovering different assumptions.

Asking Questions Reveals More than Making Statements

When I first became a Scrum Master and hadn't yet learned the power of questions, I often missed learning things about my teams and their work. Only after a while did I discover that the best way for me to learn things was to ask a question and then truly listen to the answer.

I hope you'll find these questions as useful as I have.

What Do You Think?

What are some of your favorite questions to ask teams? Please share your thoughts in the comments section below.

Get My Nine Questions PDF

Sign up to receive our free PDF summary of the Nine Questions Scrum Masters and Product Owners Should Be Asking

[Get My Nine Questions PDF Now](#)

Posted: August 29, 2017

Tagged: estimating, product owner, meetings, daily scrum, scrum master

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and

techniques to build extremely high-performance teams. He is the author of User Stories Applied for Agile Software Development, Agile Estimating and Planning, and Succeeding with Agile as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

Be a Great Product Owner: Six Things Teams and Scrum Masters Need

Learn six ways effective product owners ensure their teams' success.

Jan 31, 2023 • 16 Comments [Read](#)

Six Attributes of a Great ScrumMaster

Scrum Masters exhibit a shared set of traits. Discover 6 in-demand qualities, plus a bonus ...

Jan 17, 2023 • 32 Comments [Read](#)

What Happens When During a Sprint

Succeeding with Scrum is easier when you know when and why to conduct each of the Scrum events ...

Jan 03, 2023 • 34 Comments [Read](#)

What Are Agile Story Points?

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

Don't Equate Story Points to Hours

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • 39 Comments [Read](#)

Agile Mentors Podcast: Recap of Opening Series on Scrum

Recapping our agile podcast's initial launch series of topics

Sep 27, 2022 [Read](#)

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

by

Mike **Tagged:**

Cohn estimating, teams, teamwork, planning poker, relative estimating,
22 scrum team

Comments

A well-established best practice is that those who will do the work, [should estimate the work](#), rather than having an entirely separate group estimate the work.

But when an agile team estimates product backlog items, the team doesn't yet know who will work on each item. Teams will usually make that determination either during iteration (sprint) planning or in a more real-time manner in daily standups.

This means the whole team should take part in estimating every product backlog item. But how can someone with a skill not needed to deliver a product backlog item contribute to estimating it?

Before I can answer that, I need to briefly describe Planning Poker, which is the most common approach for estimating product backlog items. If you are already familiar with Planning Poker, you can [skip the next section](#).

Planning Poker

Planning Poker is a consensus-based, collaborative estimating approach. It starts when a product owner or key stakeholder reads to the team an item to be estimated. Team members are then encouraged to ask questions and discuss the item so they understand the work being estimated.

Each team member is holding a set of poker-style playing cards on which are written the valid estimates to be used by the team. Any values are possible, but it is generally advisable to avoid being too precise. For example, estimating one item as 99 and another as 100 seems extremely difficult as a 1% increase in effort seems impossible to distinguish. Commonly used values are 1, 2, 3, 5, 8, 13, 20, 40, and 100 (a modified Fibonacci sequence) and 1, 2, 4, 8, 16, and 32 (a simple doubling of each prior value).

Once the team members are satisfied they understand the item to be estimated, each estimator selects a card reflecting their estimate. All of the estimators then reveal their cards at the same time. If all the cards show the same value, that becomes the team's estimate of the work involved. If not, the estimators discuss their estimates with an emphasis on hearing from those with the highest and lowest values.

If you aren't familiar with estimating product backlog items this way, you may want to [read more about Planning Poker](#) before continuing.

How Can Someone Participate Without the Needed Skills

Equipped with a common understanding of Planning Poker, let's see how a team member can contribute to estimating work that they cannot possibly be involved in. As an example, consider a database engineer who is being asked to estimate a product backlog item that will include front-end JavaScript and some backend Ruby on Rails coding, and will then need to be tested.

How can this database engineer contribute to estimating this product backlog item?

There are three reasons why it's possible--and desirable.

1. Planning Poker Isn't Voting

When playing Planning Poker, participants are not *voting* on their preferred estimate. The team will *not* settle on the estimate that gets the most votes. Instead, each estimator is given the credibility they deserve. If one programmer wrote the original code that needs to be modified and happened to be in that same code a couple of days ago, the team should give more credence to that programmer's estimate than to the estimate of a programmer who has never been in this part of the system.

This means that each team member can estimate, but that the team will weigh more heavily the opinions of those more closely aligned with the work.

2. Estimates Are Relative and That's Easier

In Planning Poker, the estimates created should be *relative* rather than absolute estimates. That is, a team will say things like, "This item will take twice as long as the other item, but we can't estimate the actual number of hours for either item."

For example, this blog post contains one illustration. I provided my artist with a short description of what I had in mind for an image and he created the illustration. Most of my blog posts have one title illustration. Even though I have no artistic skill, I could estimate the work to create those illustrations as about equal each week.

Sure, some illustrations are more involved, and others can reuse a few elements from a past illustration. But most are close enough that I could estimate them as taking the same effort.

But some blog posts have two images. Even though I have no design skills at all, I'm willing to say that creating two images will take about twice as long as creating one image.

So a tester is not being asked to estimate how many hours it will take a programmer to code something. Instead the tester is estimating coding that thing relative to other things.

That can still be hard but relative estimates are easier than absolute estimates. And remember that because of point one above, the person whose skills may not be needed on the story will not be given as much credibility as someone whose skills will be used.

3. Everyone Contributes, Even If They Don't Estimate

I want everyone on the team to participate in an estimating meeting. But that does not mean everyone estimates every item.

Despite relative estimating being easier than absolute estimating, there will still be times when someone will not be able to estimate a particular product backlog item. This might be because the person's skills aren't needed on that item. But that person may still be able to *contribute* to the discussion.

Sometimes the person whose skills are not needed on a given product backlog item will be the most astute in asking questions about the item, uncovering overlooked assumptions, or in seeing work that others on the team have missed.

For example, a database developer whose skills are not needed to deliver a product backlog item may be the one who remembers that:

The team promised to clean up that code the next time they were in it

There is an impact on reports that no one has considered

That when the team did a similar story a year ago it took much longer than anyone anticipated

and so on. The database developer may know these things or ask these questions even if unable to personally estimate their impact on the work.

A Few Examples

To provide a few examples, let's return to role of a database engineer in estimating a product backlog item that has no database work. Here are some examples of things that team member might say that would add value to estimating that product backlog item:

"I'm holding up this high estimate because this sounded like a lot of effort to code. It sounded like about twice as much as this other backlog item."

In this case, the database engineer is making a relative assessment of effort. This will presumably be based on things said by coders. In some cases, the database engineer may be wrong in that assessment. But that doesn't mean the person's opinion is always without merit. The database engineer's opinion should be given the merit it deserves (which may be great or little).

"Are you sure it's that much work? I thought two sprints ago, you programmers were going to refactor that part of the system. If that happened, isn't this easier now?"

Here the database engineer is bringing up information that others may not have recalled or considered. It may or may not be of value. But sometimes it will be.

"Are you sure it's not more work than that? Have you considered the need to do this and that?"

In this case, the database engineer is pointing out work that the others may have overlooked. If that work is significant, it should be reflected in the estimate.

When Everyone Participates, It Increases Buy In

There's one final reason why I suggest the whole team participate when estimating product backlog items, especially with a technique such as Planning Poker: Doing so increases the buy in felt by all team members to the estimates.

When someone else estimates something for you or me, we don't feel invested in that estimate. It may or may not be a good estimate. But if it's not, that's not our fault. We will do much more to meet an estimate we gave than one handed to us.

We want everyone on a team to participate in estimating that team's work. You never know in advance who will ask the insightful questions about a product backlog item. Sometimes it's one of the team members who will work on that item. But other times those questions come from someone whose skills are not needed on that item.

So while not every team member needs to provide an estimate for each item, every team member does need to participate in the discussion surrounding every estimate. Teams do best when the whole team works together for the good of the product, from estimation through to delivery.

What Has Your Experience Been?

What has your experience been with involving the whole team in estimating product backlog items? Have you found it beneficial to have everyone participate even though not everyone has skills needed on each item?

[Take the Assessment](#)

Posted: July 18, 2017

Tagged: estimating, teams, teamwork, planning poker, relative estimating, scrum team

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of User Stories Applied for Agile Software Development, Agile Estimating and Planning, and Succeeding with Agile as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

[What Is Cross-Functional Collaboration in Agile?](#)

Perhaps the most prevalent and persistent myth in agile is that a cross-functional team is one on ...

Feb 14, 2023 • 49 Comments [Read](#)

[What Are Agile Story Points?](#)

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

[Don't Equate Story Points to Hours](#)

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • 39 Comments [Read](#)

[For Better Agile Planning, Be Collaborative](#)

The best plans are created by developers and stakeholders working together.

Jul 12, 2022 [Read](#)

[Relationship between Definition of Done and Conditions of Satisfaction](#)

I'd like to clarify the relationship between two important concepts: a team's Definition of Done ...

Jun 28, 2022 • 38 Comments [Read](#)

[Scrum, Remote Teams, & Success: Five Ways to Have A Three](#)

In a remote-first world, how does an agile team thrive working in a virtual, distributed way?

Jun 07, 2022 [Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile

New to Agile and Scrum?

Scrum

User Stories

Planning Poker

Agile Software Development

Agile Project Management

Agile Presentations

Mountain Goat on YouTube

Agile Mentors Podcast

Elements of Agile

Assessment

Agile & Scrum Training

Online Certifications

Video Courses

In-Person Certifications

On-Site Courses

Training Schedule

Compare Courses

PDUs and SEUs

More...

About Us

Contact Us

Our Students

Blog

Books by Mike Cohn

Agile Tools

by
Mike Cohn
28
Tagged:
estimating, story points, planning poker, relative estimating
[Comments](#)

Planning Poker relies on relative estimating, in which the item being estimated is compared to one or more previously estimated items. It is the ratio between items that is important. An item estimated as 10 units of work (generally, story points) is estimated to take twice as long to complete as an item estimated as five units of work.

An advantage to relative estimating is that it becomes easier to do as a team estimates more items.

Estimating a new item becomes a matter of looking at the previously estimated items and finding something requiring a similar amount of work. This is easier to do when the team has already estimated 100 items than when they've only estimated 10.

But, relative estimating like with Planning Poker suffers from a bootstrapping problem: How does a team select the initial estimates to which they'll compare?

My recommendation is that when a team first starts playing Planning Poker, team members identify two values that will establish their baseline. They do this without playing Planning Poker. They do it just through discussion. After the baseline is established, team members can use Planning Poker to estimate additional items.

Ideally, the team is able to identify both a two-point story and a five-point story. There is evidence that

humans estimate most reliably when sticking within one order of magnitude.

Identifying a two-point product backlog item and a five-point item does a good job of spanning this order of magnitude. Many other items can then be more reliably compared against the two and the five.

If finding a two and a five proves difficult, look instead for a two and an eight, or a three and an eight.

Anything that spans the one to 10 range where we're good estimators will work.

Avoid Starting with a One-Point Story

I like to avoid starting with a one-point story. It doesn't leave room for anything smaller without resorting to fractions, and those are harder to work with later.

Additionally, comparing all subsequent stories to a one-point story is difficult. Saying one product backlog item will take two or three times longer than another seems intuitively easier and more accurate than saying something will take 10 times longer.

I made this point in my 2005 "Agile Estimating and Planning" book (now also [a video course](#)). In 2013, it was confirmed by [Magne Jørgensen](#) of the Simula Research Lab. Jørgensen, a highly respected researcher, conducted [experiments involving 62 developers](#). He found that "using a small user story as the reference tends to make the stories to be estimated too small due to an assimilation effect."

Why Use Two Values for a Baseline?

Establishing a baseline of two values allows for even the first stories being estimated to be compared to two other items. This is known as [triangulating](#) and helps achieve more consistent estimates.

If a team has established a baseline with two- and five-point stories, team members can validate a three-point estimate by thinking whether it will take longer than the two and less time than the five.

Citing again the research of Jørgensen, there is evidence that the direction of comparison matters.

Comparing the item being estimated to one story that will take less time to develop and another that will take longer is likely to improve the estimate.

Don't Establish a New Baseline Every Project

Some teams establish a new baseline at the start of each project. Because this results in losing all historical velocity data, I don't recommend doing this as long as two things are true:

The team members developing the new system will be largely those involved in the prior system. The team doesn't need to stay entirely the same, but as long as about half the team remains the same, you're better off using the same baseline.

The team will be building a somewhat similar system. If a team is switching from developing a website to embedded firmware, for example, they should establish a new baseline. But if the systems being built are somewhat similar in either the domain or technologies used, don't establish a new baseline.

Whenever possible, retain the value of historical data by keeping a team's baseline consistent from sprint to

How Do Establish Your Baseline?

How do you estimate your baseline and initial estimates for a new team? Please share your thoughts in the comments below.

[Get my chapters now!](#)

Posted: May 31, 2016

Tagged: estimating, story points, planning poker, relative estimating

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of User Stories Applied for Agile Software Development, Agile Estimating and Planning, and Succeeding with Agile as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

What Are Agile Story Points?

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

Don't Equate Story Points to Hours

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • 39 Comments [Read](#)

For Better Agile Planning, Be Collaborative

The best plans are created by developers and stakeholders working together.

Jul 12, 2022 [Read](#)

Four Reasons Agile Teams Estimate Product Backlog Items

Estimating product backlog items provides benefits beyond predicting when a project will be finished.

May 17, 2022 [Read](#)

Automatically Triangulating Estimates in Planning Poker

Comparing estimates during Planning Poker just got easier.

Nov 23, 2021 [Read](#)

Estimating with Story Points Course Available for One Week

Registrations are now open to Estimating with Story Points.

Nov 17, 2021 [Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

by
Mike Cohn
Tagged:
estimating, story points, sprint planning, planning, tasks
70 Comments

Back when I was writing [Agile Estimating and Planning](#) (the 2005 book, but now it's also a video course), I had already been studying and experimenting with estimating approaches for five years. By conducting experiments with various teams, especially those that worked directly for me, I felt I had learned enough to write that book.

But there was quite a bit I didn't know (there still is, of course!), and so I sought out teams that were estimating or planning their projects in ways that differed from the advice I was giving. A few of these teams were using task points to estimate their sprint backlog items in conjunction with story points for the product backlog items.

This was intriguing. By then, I had already become a proponent of story points--I've since become even much more of a proponent because of the advantages that story points offer. But I couldn't see why a team would use task points. However, because I was such a fan of story points, I had to learn more about teams using task points.

To do that, I talked a couple of the teams using task points into letting me visit them so I could see what they were doing and learn more about it.

What I learned confirmed my suspicion that teams should not estimate sprint backlog tasks in task points.

The Main Benefit of Story Points

The [main benefit of story points](#) is that they allow individuals with different skills, experience levels and backgrounds to agree on a relative estimate for work.

As a silly example, suppose that an octopus and I each estimate the effort to wash my car.

The octopus has four times as many arms as I do. So, presumably, the octopus can wash my car four times faster than I can. (I told you it was a silly example. Go with it for a minute.)

But the octopus will also be four times faster than me at washing *any* car. So, the octopus and I can agree that perhaps my two-seat car can be estimated at five story points and a second, larger car can be estimated as 10 story points.

So, then, story points allow the octopus and I to agree on a number of story points, even though the octopus is presumably much faster than I am at washing cars.

Task Points

What I learned from the teams I visited was that a task point was nearly identical to a story point. All the teams I interviewed, though (or have talked to subsequently), wanted to use a more granular unit than their story points.

For example, a team that finished 10 story points per sprint might complete 80 task points per sprint.

Teams simply wanted a smaller, more granular unit to better track progress on things. It would be equivalent to a car's speedometer reporting speed in light years per hour. That would make it very hard for me to know if I'm exceeding the speed limit.

Introducing a more granular unit was a good idea for many teams. After all, it's really hard to gauge daily progress using story points for a team that finishes seven story points in two weeks, for example. For many points, story points are simply too large to be useful for this.

But We Already Have More a More Granular Unit

However, a more granular unit already exists. And it's one that every team member is already familiar with: hours.

When I looked at teams estimating in task points, I could not find an advantage over simply estimating sprint backlog tasks in hours.

There's a [fundamental difference](#) between product backlog items (most commonly, user stories) and sprint backlog tasks: a typical product backlog item will be worked on by three to five people, perhaps a programmer or two, a tester, and maybe a designer, architect, analyst or technical writer and so on. A task on the sprint backlog will usually be worked on by one person.

This fundamental difference means it is not vital for everyone on the team to agree on a task estimate as they should on a product backlog item estimate. Everyone on the team has the right to have input on a task estimate, but those who will do the work should ultimately be the ones who establish the estimate.

If there's only one person who is likely to do a specific task, put that person's estimate on the task. If two or three people might do the task, use a consensus estimate among those individuals or use the estimate of the person most likely to do the work.

If, during the sprint, the work is reallocated to someone else, the worst that will happen is that estimates will swap between tasks. A speedy team member picks up the work of a slower team member and changes an estimate from eight to four, and vice versa.

Why Not Task Points

This leaves task points without a compelling advantage over hours. Yet task points have all the drawbacks of story points:

- A foreign concept to many team members
- The need to establish baseline values against which relative estimating can begin
- A concern that estimates drift over time in comparison to the original relative values

My Advice

I finished my foray into learning about task points unconvinced of their value. I continue to advocate [capacity-driven sprint planning](#) as it's commonly known. But perhaps capacity-driven sprint planning is a better name. I find it to have [significant advantages](#) over [velocity-driven sprint planning](#).

Commitment- or capacity-driven sprint planning in hours works best for most teams. Some teams follow that general process, but throw away the estimates, using them only as a tool during the meeting to figure out what to bring into the sprint.

Other teams don't estimate sprint backlog tasks at all in any unit. Either of these approaches works well once a team has some experience with the approach.

I'm glad I undertook my project to learn more about task points and the teams using them.

In learning about alternative approaches to estimating, I'm always disappointed when a new approach doesn't turn out to be better than an existing one. But, many don't.

Yet, it's by looking at as many options as possible that we find the continuous improvements every agilist seeks.

What Do You Think?

Have you used task points? Did you find advantages to them that I've overlooked? Please share your thoughts in the comments below.

[Get my chapters now!](#)

Posted: May 17, 2016

Tagged: estimating, story points, sprint planning, planning, tasks

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of User Stories Applied for Agile Software Development, Agile Estimating and Planning, and Succeeding with Agile as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

Why I Don't Emphasize Sprint Goals

Sprint goals are considered a mandatory part of Scrum. Here's why I disagree.

Mar 21, 2023

[Read](#)

What Are Agile Story Points?

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

Don't Equate Story Points to Hours

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • 39 Comments [Read](#)

For Better Agile Planning, Be Collaborative

The best plans are created by developers and stakeholders working together.

Jul 12, 2022 [Read](#)

3 Ways to Help Agile Teams Plan Despite Uncertainty

We might not like ambiguity, but it's a fact of life. Find out how to plan with uncertainty in mind.

Jun 21, 2022

[Read](#)

Four Reasons Agile Teams Estimate Product Backlog Item

Estimating product backlog items provides benefits beyond predicting when a project will be finished.

May 17, 2022

[Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile **Agile & Scrum Training** **More...**

New to Agile and Scrum? Online Certifications About Us

Scrum Video Courses Contact Us

User Stories In-Person Certifications Our Students

Planning Poker On-Site Courses Blog

Agile Software Development Training Schedule Books by Mike Cohn

Development Compare Courses Agile Tools

Agile Project Management PDUs and SEUs

Agile Presentations

Mountain Goat on YouTube

Agile Mentors Podcast

Elements of Agile

Assessment

by
Mike Cohn 59
Tagged:
estimating, story points, planning poker, estimate inflation
[Comments](#)

I spoke with a Scrum Master recently who told me his team had nearly doubled their velocity in only two months. Rather than be happy about this, though, he was concerned.

He knew the team had not suddenly become twice as productive. In fact, he doubted they'd actually sped up at all. Yet their velocity showed they had.

The cause of this sudden and dramatic increase was story point inflation. Or, more generally, estimate inflation, because the problem can happen with estimating units other than just story points.

Estimate inflation is when the estimate assigned to a product backlog item (usually a user story) increases over time. For example, today the team estimates something will take five points but previously they would have called it three points.

Why Does Estimate Inflation Happen?

There are a few possible causes of estimate inflation. One of the most common, though, is excessive pressure on the team to improve or deliver more points per sprint. This often comes from bosses or possibly stakeholders outside the team who are pushing the team.

Velocity becomes a really tempting (but bad) metric in these cases and teams are pushed to demonstrate that they're going faster by increasing velocity.

When a team is under pressure to increase velocity, team members will often start to round estimates up during story point estimate meetings (often done with [Planning Poker](#)). For example, consider a team that is debating whether a particular story is three or five points. They're having a legitimate debate about this.

At some time during that discussion, one or more people will remember the team is under pressure to increase velocity. And some might shift in favor of calling the story five points instead of three.

I want to be clear this isn't lying. It's not blatant padding. The team was truly debating three versus five. And when someone remembers the team is under pressure, that person switches to favor five.

And so that story is called a five.

Now consider another story being estimated perhaps a week or two later. In considering the new story, someone compares it to the five-point story and thinks, "Well, this new one is a little bigger than that five," and proceeds to estimate it as perhaps an eight.

And this is how estimate inflation happens.

How to Prevent Estimate Inflation

I've found the best way to prevent estimate inflation from occurring is to always compare the item being estimated against two (or more) previously estimated product backlog items. In my "[Agile Estimating and Planning](#)" book, I referred to this as triangulation, borrowing the old nautical term for fixing a ship's location.

So, when a team thinks about estimating a story as five points, they would first compare that story to two other stories--ideally one smaller and one larger. In deciding if a story should be estimated as five points, they would compare the story to a three-point story and think, *Will the effort to do this new story be a little more than this three-pointer?*

They would next compare that story against an eight- or 13-point story. And they'd want to see if the story felt appropriately sized as five in comparison to one of those. These comparisons are shown in Figure 1.

Triangulating a 5-point story by comparing it with 3-point and 13-point stories.

When an item being estimated is compared to two or more previously estimated items, it helps ensure the internal consistency of the estimates.

Ideally we'd love to consider each estimate in comparison to *all* previous estimates. But that would be way too much work. Triangulating a story by comparing it to two others is generally sufficient.

If we think about the stories as nodes in a graph, triangulating can be visualized by drawing lines between each of the nodes the team explicitly compared while estimating. This can be seen in Figure 2.

Each story, A-F, has been compared with two or three other stories.

Here we see that product backlog items A and B have been compared to three other items each. Backlog items C through F have each been compared twice.

Triangulating Stops Estimate Inflation

Triangulating prevents estimate inflation because the use of two comparisons helps point out when estimates are beginning to inflate.

To see this, consider the team that is trying to decide between estimating a story as either three or five. Remembering they are under pressure to increase velocity, they decide to call it a five. And it may legitimately seem just a bit bigger than some other three-point stories.

But, when the team triangulates that story against another five or an eight, they'll most likely realize that the story is not really a five.

There's One More Good Way to Prevent Estimate Inflation

There's at least one more very good way to prevent estimate inflation. But, since this post is already long, I'll save that for the weekly tip I share each Thursday by email. If you're not already subscribed, consider [signing up now](#) if you're interested in learning more.

What Do You Think?

Has estimate inflation been a problem on your team? How do you handle it? Please share your thoughts in the comments below.

Posted: May 3, 2016

Tagged: estimating, story points, planning poker, estimate inflation

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the *Better User Stories* video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

[What Are Agile Story Points?](#)

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • [122 Comments](#)

[Read](#)

[Don't Equate Story Points to Hours](#)

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • [39 Comments](#)

[Read](#)

[For Better Agile Planning, Be Collaborative](#)

The best plans are created by developers and stakeholders working together.

Jul 12, 2022

[Read](#)

[Four Reasons Agile Teams Estimate Product Backlog Items](#)

Estimating product backlog items provides benefits beyond predicting when a project will be finished.

May 17, 2022

[Read](#)

[Automatically Triangulating Estimates in Planning Poker](#)

Comparing estimates during Planning Poker just got easier.

Nov 23, 2021

[Read](#)

Estimating with Story Points Course Available for One Week

Registrations are now open to Estimating with Story Points.

Nov 17, 2021

[Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

Should You Use Zero-Point Estimates on Your Product Backlog?

by

Mike Cohn

Tagged:

product backlog, estimating, backlog, planning poker, zero points

33 Comments

I'm occasionally asked about the merits (and whether there are any) of putting an estimate of zero on user stories. You may have noticed that our decks of [Planning Poker cards](#) include a zero card.

So, what's the logic behind a zero-point card?

Remember that story points are used to estimate the [effort](#) involved in delivering a product backlog item. A zero-point estimate simply indicates that delivering that item does not require "any" effort. That is usually an exaggeration. The product backlog item will take some effort but usually so little that the team will not want any velocity credit for it.

As an example, I looked in the recently completed items from the www.frontrowagile.com site and found a zero-point user story. The story had to do with a file that is generated each night and then automatically imported into our accounting system. The story involved changing the order of a few columns in the file and changing how dates are formatted. The story took the team about 15 minutes to code and test.

If the team had estimated this as a one, they would have had a higher velocity for doing that trivial amount of work. But in the next sprint, I might have expected them to maintain that velocity, which they might not have been able to do; a one-point story is normally a lot more than 15 minutes of work.

How Can Zero-Point Stories Deliver Value?

Sometimes I'm asked if product backlog items are supposed to deliver value, how can a zero-point product backlog item deliver value?

The value of a product backlog item is independent of the work to deliver the item. Think of the relationship between calories and the size of a food item. There really isn't any.

I remember when one of my daughters was young, she came home from school fascinated by the idea that celery has so few calories in it that we theoretically burn more calories chewing it than are in it. So, I could have a massive bowl of celery and consume very few calories. The same size bowl of ice cream would, unfortunately, have many more calories.

So it is with value, and the effort to deliver a product backlog item. A product backlog item with a zero-point estimate to develop it can still deliver value. Changing the sort order on the report in the example above saved our accountant a lot of manual time.

Zero-point stories are not something I recommend you use a lot of. But, used judiciously, they can still have their place on a product backlog.

[Download my PDF](#)

Posted: December 1, 2015

Tagged: product backlog, estimating, backlog, planning poker, zero points

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the *Better User Stories* video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

[What Are Agile Story Points?](#)

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

[Don't Equate Story Points to Hours](#)

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • 39 Comments [Read](#)

[Epics, Features and User Stories](#)

I've been getting more and more emails lately from people confused about the difference between ...

Nov 08, 2022 • 93 Comments [Read](#)

[For Better Agile Planning, Be Collaborative](#)

The best plans are created by developers and stakeholders working together.

Jul 12, 2022 [Read](#)

[Relationship between Definition of Done and Conditions of Satisfaction](#)

I'd like to clarify the relationship between two important concepts: a team's Definition of Done ...

Jun 28, 2022 • 38 Comments [Read](#)

[Four Reasons Agile Teams Estimate Product Backlog Item](#)

Estimating product backlog items provides benefits beyond predicting when a project will be finished.

May 17, 2022 [Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile

New to Agile and Scrum?

Scrum

User Stories

Planning Poker

Agile Software Development

Agile Project Management

Agile Presentations

Mountain Goat on YouTube

Agile Mentors Podcast

Elements of Agile

Assessment

Agile & Scrum Training

Online Certifications

Video Courses

In-Person Certifications

On-Site Courses

Training Schedule

Compare Courses

PDUs and SEUs

More...

About Us

Contact Us

Our Students

Blog

Books by Mike Cohn

Agile Tools

Change Isn't Free

by
Mike Cohn
38 **Tagged:** estimating, requirements, stakeholders, change, cost
[Comments](#)

Agile teams are told to "embrace change," which is the subtitle to Kent Beck's wonderful *Extreme Programming Explained* book. Although an agile team can embrace change, the stakeholders in an organization must understand that change is not always free.

Most agile teams seem to understand this. They get that requirements changes can crossover into requirements churn, to use the traditional term for the problem. Scrum tries to solve the problem by including as part of its framework a rule that change is not allowed into a sprint. Once a sprint has started, change is kept out of the sprint, leaving the team to focus on what was selected to be worked on in the sprint.

This can work extremely well, but keeping change out of the sprint is sometimes thrown back in the face of the team. A product owner or stakeholder who is told that a change cannot be introduced will respond with, "But I thought you're agile, and isn't the whole point of being agile that you welcome changing requirements?"

Well, yes and no. A team can welcome changing requirements but that doesn't mean those changes are free. Consider the following scenario:

You're out to dinner and settle on chicken as the main dish. As the waiter walks away, you call him back and tell him you've changed your mind. You'd prefer the fish. No problem, he says. There is no cost to this change. The restaurant is not going to charge you for changing your mind (changing the requirements) five seconds into your order (five seconds into the iteration).

But, let's make things a little more interesting.

Suppose the waiter takes your order for chicken and a salad to start. He brings you the salad, and you then change your order for the main dish from chicken to fish. My experience is that most restaurants will still not charge you for this. They'll bring you the fish even though they may have started cooking the chicken.

There may be a real cost to the restaurant. But I suspect they write it off as goodwill, hoping you enjoy your meal and the courtesy they showed you. And they hope you come back more often because of it.

And let's say you do come back. In fact, you come back every night. And every night, you jerk the waiter around by changing your order after he brings you the salad. The restaurant is incurring real costs because of you.

They may still be nice and not pass the costs of these changes onto you. But, if I were that restaurant, I'd

pass a different type of cost onto you: I'd wait to start cooking your main dish until after you ate the salad so that we were both sure of your order.

Let's consider one additional scenario. In this case, after eating the salad, you change your order of chicken to fish, as before. But before the fish comes, you call the waiter over and tell him you've re-considered and the steak sounds appealing.

Then, before the steak arrives, you call the waiter again and tell him you saw the linguine served to another diner and simply must have that. At some point, the restaurant has incurred real costs from all this switching and will charge you for it.

Change is not free.

You can change your order at the restaurant up to a point for free. You can change it after that and the restaurant will do all they can to minimize the impact of that change, but beyond some point there is a cost to changing.

The same is true on product development initiatives. We can and should embrace change. It is often why we want to be agile. But we can also educate our stakeholders so they understand that change introduced beyond a certain point in an iteration has a cost. And we can work together with them to minimize those costs.

[Get my chapters now!](#)

Posted: October 13, 2015

Tagged: estimating, requirements, stakeholders, change, cost

[About the Author](#)

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

[What Are Agile Story Points?](#)

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • [122 Comments](#)

[Read](#)

[Don't Equate Story Points to Hours](#)

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • [39 Comments](#)

[Read](#)

[7 Ways to Get and Improve Fast Feedback](#)

Here's how to get the rapid feedback you need to ensure you build the right product.

Jul 26, 2022

[Read](#)

For Better Agile Planning, Be Collaborative

The best plans are created by developers and stakeholders working together.

Jul 12, 2022

[Read](#)

Four Reasons Agile Teams Estimate Product Backlog Items

Estimating product backlog items provides benefits beyond predicting when a project will be finished.

May 17, 2022

[Read](#)

Top 7 Ways to Get Stakeholders to Attend Sprint Reviews

Poorly attended sprint reviews cause real problems. Fortunately there are easy fixes.

Mar 08, 2022

[Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

Budget When You Can't Estimate

by
Mike Cohn **Tagged:** product backlog, estimating, product owner, teams, budget
35 Comments

I've written before that we should only estimate if having the estimate will change someone's actions.

Normally, this means that the team should estimate work only if having the estimate will either:

- enable the product owner to better prioritize the product backlog, or

- let the product owner answer questions about when some amount of functionality can be available.

But, even if we only ask for estimates at these times, there will be some work that the team finds extremely hard or nearly

impossible to estimate. Let's look at the best approach when an agile team is asked to estimate work they don't think they can reasonably estimate. But let's start with some of the reasons why an agile team may not be able to estimate well.

Why a Team Might Not Be Able to Estimate Well

This might occur for a variety of reasons. It happens frequently when a team first begins working in a new domain or with new technology.

Take a set of highly experienced team members who have worked together for years building, let's say, healthcare applications for the Web. And then move them to building banking applications for mobile devices. They can come up with some wild guesses about that new mobile banking app. But that's about all that estimate would be--a wild guess.

Similarly, team members can have a hard time estimating when they aren't really much of a team yet. If seven people are thrown together today for the first time and immediately asked to estimate a product backlog, their estimates will suck. Individuals won't know who has which skills (as opposed to who says they have those skills). They won't know how well they collaborate, and so on. After such a team works together for perhaps a few months, or maybe even just a few weeks, the quality of their estimates should improve dramatically.

There can also be fundamental reasons why an estimate is hard to produce. Some work is hard to estimate. For example, how long until cancer is cured? Or, how long will it take to design (or architect) this new system? Teams hate being asked to estimate this kind of work. In some cases, it's done when it's done, as in the cancer example. In other cases, it's done when time runs out, as in the design or architecture example.

Budgeting Instead of Estimating

In cases like these, the best thing is to approach the desire for an estimate from a different direction. Instead of asking a team, "How long will this take?" the business thinks, "How much time is this worth?" In effect, what this does is create a budget for the feature, rather than an estimate. Creating a budget for a feature is essentially applying the agile practice of timeboxing. The business says, "This feature is worth four iterations to us." In a healthy, mature agile organization, the team should be given a chance to say if they think they can do a reasonable job in that amount of time. If the team does not think it can build something the business will value in the budgeted amount of time, a discussion should ensue. What if the budget were expanded by another sprint or two? Can portions of the feature be considered optional so that the mandatory features are delivered within the budget? Establishing a budget frames this type of discussion.

What Do You Think?

What do you think of this approach? Are there times when you'd find it more useful to put a budget or timebox on a feature rather than estimate it? Have you done this in the past? I'd love to know your thoughts in the comments below.

[Take the Assessment](#)

Posted: September 1, 2015

Tagged: product backlog, estimating, product owner, teams, budget

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

What Is Cross-Functional Collaboration in Agile?

Perhaps the most prevalent and persistent myth in agile is that a cross-functional team is one on ...

Feb 14, 2023 • 49 Comments [Read](#)

Be a Great Product Owner: Six Things Teams and Scrum Masters Need

Learn six ways effective product owners ensure their teams' success.

Jan 31, 2023 • 16 Comments [Read](#)

What Are Agile Story Points?

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

Don't Equate Story Points to Hours

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • 39 Comments [Read](#)

Epics, Features and User Stories

I've been getting more and more emails lately from people confused about the difference between ...

Nov 08, 2022 • 93 Comments [Read](#)

For Better Agile Planning, Be Collaborative

The best plans are created by developer and stakeholders working together.

Jul 12, 2022 [Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile **Agile & Scrum Training** **More...**

New to Agile and Scrum? Online Certifications About Us

Scrum Video Courses Contact Us

User Stories In-Person Certifications Our Students

Planning Poker On-Site Courses Blog

Agile Software Development Training Schedule Books by Mike Cohn

Development Compare Courses Agile Tools

Agile Project Management PDUs and SEUs

Agile Presentations

Mountain Goat on YouTube

Agile Mentors Podcast

Elements of Agile

Assessment

Estimates on Split Stories Do Not Need to Equal the Original

by
Mike Cohn Tagged:
12 estimating, user stories, story points, planning poker
[Comments](#)

It is good practice to first write large user stories (commonly known as epics) and then to split them into smaller pieces, a process known as [product backlog refinement or grooming](#). When product backlog items are split, they are often re-estimated.

I'm often asked if the sum of the estimates on the smaller stories must equal the estimate on the original, larger story.

No.

Part of the reason for splitting the stories is to understand them better. Team members discuss the story with the product owner. As a product owner clarifies a user story, the team will know more about the work they are to do.

That improved knowledge should be reflected in any estimates they provide. If those estimates don't sum to the same value as the original story, so be it.

But What About the Burndown?

But, I hear you asking, what about the release burndown chart? A boss, client or customer was told that a story was equal to 20 points. Now that the team split it apart, it's become bigger.

Well, first, and I always feel compelled to say this: We should always stress to our bosses, clients and customers that [estimates are estimates and not commitments](#).

When we told them the story would be 20 points, that meant perhaps 20, perhaps 15, perhaps 25. Perhaps even 10 or 40 if things went particularly well or poorly.

OK, you've probably delivered that message, and it may have gone in one ear and out the other of your boss, client or customer. So here's something else you should be doing that can protect you against a story becoming larger when split and its parts are re-estimated.

I've always written and trained that the numbers in [Planning Poker](#) are best thought of as buckets of water.

You have, for example an 8 and a 13 but not a 10 card. If you have a story that you think is a 10, you need to estimate it as a 13. This slight rounding up (which only occurs on medium to large numbers) will mitigate the effect of stories becoming larger when split.

Consider the example of a story a team thinks is a 15. If they play Planning Poker the way I recommend, they will call that large story a 20.

Later, they split it into multiple smaller stories. Let's say they split it into stories they estimate as 8, 8 and 5. That's 21. That's significantly larger than the 15 they really thought it was, but not much larger at all than the 20 they put on the story.

In practice, I've found this slight pessimistic bias to work well to counter the natural tendency I believe many developers have to underestimate, and to provide a balance against those who will be overly shocked when any actual overruns its estimate.

[Get my chapters now!](#)

Posted: July 28, 2015

Tagged: estimating, user stories, story points, planning poker

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

User Stories: How to Create Story Maps

Story maps help to create a shared understanding of the product, visualize user needs, and elicit ...

Mar 14, 2023

[Read](#)

[How to Run a Successful User Story Writing Workshop](#)

What you need to know to conduct a story-writing workshop with your team.

Feb 28, 2023 • [3 Comments](#)

[Read](#)

[What Are Agile Story Points?](#)

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • [122 Comments](#)

[Read](#)

[Don't Equate Story Points to Hours](#)

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • [39 Comments](#)

[Read](#)

[Epics, Features and User Stories](#)

I've been getting more and more emails lately from people confused about the difference between ...

Nov 08, 2022 • [93 Comments](#)

[Read](#)

Advantages of User Stories over Requirements and Use Cases

At the surface, user stories appear to have much in common with use cases and traditional ...

Oct 05, 2022 • [41 Comments](#)

[Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Estimates on Split Stories Do Not Need to Equal the Original

Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

by

Mike **Tagged:**

Cohn estimating, product owner, sprint planning, commitment,
77 commitment driven

Comments

There are two general ways to plan a sprint: [velocity-driven sprint planning](#) and capacity-driven sprint planning. In last week's post, I described velocity-driven planning; so in this week's, we turn our attention to capacity-driven sprint planning.

A capacity-driven sprint planning meeting involves the product owner, Scrum Master and all development team members. The product owner brings the top-priority product backlog items into the meeting and describes them to the team, usually starting with an overview of the set of high-priority items.

Select a Product Backlog Item

Following that, team members select a first item to bring into the sprint. This will almost always be the product owner's top-priority item, but it is possible that the product owner's top priority has too many open issues.

Ideally, a team should be able to still bring that item into the sprint and resolve the issues early enough in the sprint to complete the item. But, it's possible that there are so many issues, that the issues are so significant, or that resolving the issues would take so much time (for example, the need to convene a meeting with 25 user representatives) that the product owner's top priority is skipped.

Discuss that Product Backlog

Having selected a high-priority item, team members discuss the work involved, and identify the tasks that will be necessary to deliver the product backlog item.

Most teams will also *roughly* estimate the hours to do each task. These estimates are rough because the estimates will be used only to influence how many and which product backlog items are brought into the sprint. To do that, estimates do not need to be precise.

Do not ask or expect a team to think of every task that will be done during the sprint. Not only is that impossible, it is also unnecessary.

Teams should think of enough of the tasks that they feel they have thought through the work—but it is important to realize that thinking through the work is the real goal of this meeting. Identifying tasks and hours is secondary. For most teams, though, it's the best tool to know how many product backlog items to bring into a sprint.

Decide if the Item Can Be Completed Within the Sprint

After they've identified tasks and roughly estimated the hours for that one product backlog item, the team members ask themselves, "Can we commit to this?"

I find it very important that the team members ask this collectively of themselves rather than having a ScrumMaster ask, "Can you commit to this?" When team members ask, "Can we commit?" they are committing to each other rather than to the ScrumMaster.

I don't know about you, but my early, pre-Scrum career is littered with broken "commitments" to bosses who asked if I could deliver something while making it clear my answer better be yes.

The ScrumMaster isn't a boss and shouldn't create that type of feeling among team members, but the person is called "master"—and it's better not to risk being perceived as a boss insisting on a commitment.

Coach a team to ask, "Can we commit?" and it's clear that they are committing to one another, which will likely be a stronger commitment.

Further, by having the team ask themselves, "Can we commit?" it is clear that the answer should be, "Yes we can" or "No we can't." When a ScrumMaster asks, "Can you commit?" some team members will properly answer with "we" but others will answer with "I."

Scrum demands a full-team commitment: If you're behind, I'll help, and I know you'll do the same for me. It's not "these are my tasks" and "those are yours."

Repeat with More Product Backlog Items

If the team agrees they can deliver a product backlog item, they select another item and repeat the process. And they continue repeating it until someone says they cannot commit to the selected product backlog item.

If someone cannot complete the item within the sprint, team members will generally discuss the situation and see if someone else is available to help—perhaps a DBA with rudimentary JavaScript skills can help an overwhelmed JavaScript developer.

If not, perhaps that item can be put back on the product backlog but a smaller item can be brought in, or an item that needs less of the skills possessed by the person who could not complete the work.

What About Story Points and Velocity?

You may have noticed that in the process so far, there has been no role for story points or velocity. Although I still recommend that product backlog items be given quick, high-level estimates in story points, neither story points nor velocity play a role in capacity-driven sprint planning as described so far.

They do, however, play an important role in the final step of a sprint planning meeting.

Sanity Checking the Commitment

Once team members have filled their available time in the sprint, the Scrum Master can look at the selected product backlog items, sum the story points assigned to each, and share that sum with the team. Team members can then compare it to the average or recent velocity.

Suppose a team with an average velocity of 20 conducts a capacity-driven sprint planning meeting and selects 19 points of work. They've done this without knowing the story point values on any of the selected product backlog items.

When their Scrum Master tells them they've just selected 19 points of work and have an average velocity of 20, that team should feel very confident they've selected an appropriate amount of work for the sprint.

Suppose instead, though, that the ScrumMaster for this team announces they've selected only 11 points of work. They might in that case ask themselves why they were making the work so hard during sprint planning as compared to when they'd earlier estimated the same items in story points.

For example, this may reveal that during sprint planning they'd identified work they'd earlier not thought about, or perhaps had explicitly assumed would not be part of a given story. Or they may discover that the story really is harder than they'd thought when assigning points to it.

Either way, knowing they'd selected 11 yet averaged 20 will help the team know they've selected an appropriate amount of work or perhaps make a change to bring more.

Similarly, if the ScrumMaster announces that the team has selected 30 points, 10 more than their average velocity, team members may wonder what they are forgetting to consider. "Why," they would discuss, "does this work seem so much easier after sprint planning than it did while estimating story points?"

So: story points and velocity do not play a role during the main portion of a capacity-driven sprint planning meeting. But they play the vital role of acting as a sanity check and confirmation of the plan.

It's a Commitment, Not a Guarantee

It is important that the team's commitment not be viewed as a guarantee. As Clint Eastwood said in one of his movies, "If you want a guarantee, buy a toaster."

The team's commitment is to do its best. I'd like to see them make their commitment perhaps 80 percent of the time. It should be something they take seriously and should make most of that time. That's needed for the business to gain confidence in what a team says it can deliver.

However, finishing everything they say they will 100 percent of the time should not be the goal. A team forced to finish everything every time will do so—but by reducing what they commit to.

Read the Rest of This Series on Sprint Planning

This post was the second in a three-part series on sprint planning. Part one covered [velocity-driven sprint planning](#), an entirely different approach. In part three, I'll share my [preference between velocity-driven and capacity-driven sprint planning](#).

Note

This post was originally titled "Commitment-Driven Sprint Planning." I've re-titled to the more appropriate term "Capacity-Driven Sprint Planning" since the original term led to frequent misunderstandings that a commitment was a guarantee. Minor edits for consistency were made to the rest of the post.

Get Your Free Scrum Cheat Sheet

Sign up below to receive this FREE reference.

First Name

Email Address

We hate spam and promise to keep your email address safe. Unsubscribe at any time. [Privacy Policy](#)

Posted: October 28, 2014

Tagged: estimating, product owner, sprint planning, commitment, commitment driven

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile Assessment		

by

Mike **Tagged:**

Cohn product backlog, estimating, story points, effort, collaboration,

87 benefits

Comments

If story points are an estimate of the time (effort) involved in doing something, why not just estimate directly in hours or days? Why use points at all?

There are multiple good reasons to estimate product backlog items in story points, and I cover them fully in

the [Agile Estimating and Planning video course](#), but there is one compelling reason that on its own is enough to justify the use of points.

It has to do with King Henry I who reigned between 1100 and 1135. Prior to his reign, a “yard” was a unit of measure from a person’s nose to his outstretched thumb. Just imagine all the confusion this caused with that distance being different for each person.

King Henry eventually decided a yard would always be the distance between the king’s nose and outstretched thumb. Convenient for him, but also convenient for everyone else because there was now a standard unit of measure.

You might learn that for you, a yard (as defined by the king’s arm) was a little more or less than your arm. I’d learn the same about my arm. And we’d all have a common unit of measure.

Story points are much the same. Like English yards, they allow team members with different skill levels to communicate about and agree on an estimate. As an example, imagine you and I decide to go for a run. I like to run but am very slow. You, on the other hand, are a very fast runner. You point to a trail and say, “Let’s run that trail. It’ll take 30 minutes.”

I am familiar with that trail, but being a much slower runner than you, I know it takes me 60 minutes every time I run that trail. And I tell you I’ll run that trail with you but that will take 60 minutes.

And so we argue. “30.” “60.” “30.” “60.”

We’re getting nowhere. Perhaps we compromise and call it 45 minutes. But that is possibly the worst thing we could do. We now have an estimate that is no good for either of us.

So instead of compromising on 45, we continue arguing. “30.” “60.” “30.” “60.”

Eventually you say to me, “Mike, it’s a 5-mile trail. I can run it in 30 minutes.”

And I tell you, “I agree: it’s a 5-mile trail. That takes me 60 minutes.”

The problem is that we are both right. You really can run it in 30 minutes, and it really will take me 60. When we try to put a time estimate on running this trail, we find we can’t because we work (run) at different speeds.

But, when we use a more abstract measure—in this case, miles—we can agree. You and I agree the trail is 5 miles. We just differ in how long it will take each of us to run it.

Story points serve much the same purpose. They allow individuals with differing skill sets and speeds of working to agree. Instead of a fast and slow runner, consider two programmers of differing productivity.

Like the runners, these two programmers may agree that a given user story is 5 points (rather than 5 miles). The faster programmer may be thinking it’s easy and only a day of work. The slower programmer may be thinking it will take two days of work. But they can agree to call it 5 points, as the number of points assigned to the first story is fairly arbitrary.

What’s important is once they agree that the first story is 5 points, our two programmers can then agree on subsequent estimates. If the fast programmer thinks a new user story will take two days (twice his estimate

for the 5-point story), he will estimate the new story as 10 points. So will the second programmer if she thinks it will take four days (twice as long as her estimate for the 5-point story).

And so, like the distance from King Henry's nose to his thumb, story points allow agreement among individuals who perform at different rates.

[Get my chapters now!](#)

Posted: September 9, 2014

Tagged: product backlog, estimating, story points, effort, collaboration, benefits

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

[What Are Agile Story Points?](#)

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

[Don't Equate Story Points to Hours](#)

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • 39 Comments [Read](#)

[Epics, Features and User Stories](#)

I've been getting more and more emails lately from people confused about the difference between ...

Nov 08, 2022 • 93 Comments [Read](#)

[For Better Agile Planning, Be Collaborative](#)

The best plans are created by developers and stakeholders working together.

Jul 12, 2022 [Read](#)

[Relationship between Definition of Done and Conditions of Satisfaction](#)

I'd like to clarify the relationship between two important concepts: a team's Definition of Done ...

Jun 28, 2022 • 38 Comments [Read](#)

[Scrum, Remote Teams, & Success: Five Ways to Have A Three](#)

In a remote-first world, how does an agile team thrive working in a virtual, distributed way?

Jun 07, 2022 [Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile **Agile & Scrum Training** **More...**

New to Agile and Scrum? Online Certifications About Us

Scrum Video Courses Contact Us

User Stories In-Person Certifications Our Students

Planning Poker On-Site Courses Blog

Agile Software Development Training Schedule Books by Mike Cohn

Development Compare Courses Agile Tools

Agile Project Management PDUs and SEUs

Agile Presentations

Mountain Goat on YouTube

Agile Mentors Podcast

Elements of Agile

Assessment

by

Mike **Tagged:**

Cohn product backlog, estimating, story points, effort, time, risk,

106 complexity

Comments

Story points are about time. There, I've said it, and can't be more clear than that. I've written previously about why [story points are about effort, not complexity](#). But I want to revisit that topic here.

The primary reason for estimating product backlog items is so that predictions can be made about how much

functionality can be delivered by what date. If we want to estimate what can be delivered by when, we're talking about time. We need to estimate time. More specifically, we need to estimate effort, which is essentially the person-days (or hours) required to do something.

Estimating something other than effort may be helpful, but we can't use it to answer questions about when a project can be delivered. For example, suppose a team were to estimate for each product backlog item how many people would be involved in delivering that item.

One item might involve only a programmer and a tester, so it is given a "two." Another item might involve two programmers, a designer, a database engineer, and a tester. So it is given an estimate of "five."

It is entirely possible that the product backlog item involving only two people will take significantly longer than the one involving five people. This would be the case if the two people were involved intensely for days while the five were only involved for a few hours.

We may say that the number of people involved in delivering a product backlog item is a proxy for how long the feature will take to develop. In fact, I'd suspect that if we looked at a large number of product backlog items, we would see that those involving more people do, on average, take longer than those involving fewer people.

However, I'm equally sure we'd see lots of counter-examples, like that of the five and two people above. This means that the number of people involved is not a very good proxy for the effort involved in delivering the feature.

This is the problem with equating story points with complexity. Complexity is a factor in how long a product backlog item will take to develop. But complexity is not the only factor, and it is not sufficiently explanatory that we can get by with estimating just the complexity of each product backlog item.

Instead, story points should be an estimate of how long it will take to develop a user story. Story points represent time. This has to be so because time is what our bosses, clients and customers care about. They only care about complexity to the extent it influences the amount of time something will take.

So story points represent the effort involved to deliver a product backlog item. An estimate of the effort involved can be influenced by risk, uncertainty, and complexity.

Let's look at an example:

Suppose you and I are to walk to a building. We agree that it will take one walking point to get there. That doesn't mean one minute, one mile or even one kilometer. We just call it one walking point. We could have called it 2, 5, 10 or a million, but let's call it 1.

What's nice about calling this one walking point is that you and I can agree on that estimate, even though you are going to walk there while I hobble over there on crutches. Clearly you can get there much faster than I can; yet using walking points, we can agree to call it one point.

Next, we point to another building and agree that walking to it will take two points. That is, we both think it will take us twice as long to get to.

Let's add a third building. This building is physically the same distance as the two-point building. So we are

tempted to call it a two. However, separating us from that building is a narrow walkway across a deep chasm filled with boiling lava. The walkway is just wide enough that we can traverse it if we're extremely careful. But, one misstep, and we fall into the lava.

Even though this third building is the same physical distance as the building we previously estimated as two walking points, I want to put a higher estimate on this building because of the extra complexity in walking to it.

As long as I'm cautious, there's no real risk of falling into the lava, but I assure you I am going to walk more slowly and deliberately across that walkway. So slow, in fact, that I'm going to estimate that building as four walking points away.

Make sense? The extra complexity has influenced my estimate.

Complexity influences an estimate, but only to the extent the extra complexity affects the effort involved in doing the work. Walking to the one-point building while singing "Gangnam Style" is probably more complex than walking there without singing. But the extra complexity of singing won't affect the amount of time it takes me to walk there, so my estimate in this case would remain one.

Risk and uncertainty affect estimates similarly. Suppose a fourth building is also physically the same distance as the building we called a two. But in walking to that building we must cross some train tracks. And the train crosses at completely unpredictable times.

There is extra uncertainty in walking to that building—sometimes we get there in two points. Other times we get stuck waiting for the train to pass and it takes longer. On average, we might decide to estimate this building as a three.

So, story points are about time—the effort involved in doing something. Because our bosses, clients and customers want to know when something will be done, we need to estimate with something based on effort. Risk, uncertainty and complexity are factors that may influence the effort involved.

Let me know what you think in the comments below.

Get Your Free Scrum Cheat Sheet

Sign up below to receive this FREE reference.

First Name

Email Address

[Privacy Policy](#)

Posted: September 2, 2014

Tagged: product backlog, estimating, story points, effort, time, risk, complexity

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

[What Are Agile Story Points?](#)

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments

[Read](#)

[Don't Equate Story Points to Hours](#)

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • [39 Comments](#)

[Read](#)

[Epics, Features and User Stories](#)

I've been getting more and more emails lately from people confused about the difference between ...

Nov 08, 2022 • [93 Comments](#)

[Read](#)

For Better Agile Planning, Be Collaborative

The best plans are created by developers and stakeholders working together.

Jul 12, 2022

[Read](#)

Relationship between Definition of Done and Conditions of Satisfaction

I'd like to clarify the relationship between two important concepts: a team's Definition of Done ...

Jun 28, 2022 • [38 Comments](#)

[Read](#)

Four Reasons Agile Teams Estimate Product Backlog Items

Estimating product backlog items provides benefits beyond predicting when a project will be finished.

May 17, 2022

[Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

Paying the Cost for More Precise Estimates

by
Mike Cohn **Tagged:** product backlog, estimating, teams
22 Comments

I have no problem with a boss who asks a team to provide an estimate for how long a huge set of product backlog items will take to deliver. There could be hundreds of items – perhaps even thousands, and I have no issue with the boss asking for an estimate. I will be happy to provide a boss with that information.

So long as the boss understands that information has a cost.

I don't mean the boss has to slip the team a \$20 tip. What I'm referring to is the time the team will

spend creating that estimate.

If a boss wants a really quick, down-and-dirty estimate of when a set of features will be done, I can answer that instantly. My answer: in the future. It's certainly accurate.

But, it's probably not what the boss is looking for. (And don't blame me if you get fired for trying that answer!)

Perhaps a better estimate is needed. So the team puts more effort into estimating and thinks harder about what's being asked for. They may come up with something like, "Three to six months." That's certainly better than "in the future." And often, it's good enough for decisions like, "Should we start this project?" and, "Should we add more teams?"

But, many bosses will insist on something more precise—perhaps an answer like, "In August." Note that telling your boss a month is still framing your estimate as a range, which is vital to giving an accurate estimate. But, it's much more precise than saying, "Three to six months."

To add precision to an estimate requires time. And that is the cost I'm referring to.

As long as a boss is willing to pay that cost, a good team should be willing to provide an estimate at whatever level of detail and precision the boss wants. The problem is when the boss wants a highly precise estimate and isn't willing to pay the cost, for example, "I need to know by lunch time today exactly when this large project can be delivered."

When asked by a boss to provide an estimate you'd prefer not to provide, do not argue with the boss that the estimate is unnecessary. That will usually get you nowhere. Instead take the attitude that you're happy to provide the estimate (even if you aren't), but challenge the boss about how precise or detailed the estimate needs to be.

If providing the estimate will cause all other work to stop for a week, the boss needs to understand that. If the boss has a true need for the estimate, he or she may be willing to let all other work stop. If the boss just wants an estimate to feel good or to hold the team accountable, the boss will probably settle for an easier-to-provide but less precise estimate.

Information has a cost. If the person asking for that information is willing to pay that cost, you need to be willing to provide the information.

[Take the Assessment](#)

Posted: May 20, 2014

Tagged: product backlog, estimating, teams

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

What Is Cross-Functional Collaboration in Agile?

Perhaps the most prevalent and persistent myth in agile is that a cross-functional team is one on ...

Feb 14, 2023 • 49 Comments [Read](#)

What Are Agile Story Points?

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

Don't Equate Story Points to Hours

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • 39 Comments [Read](#)

Epics, Features and User Stories

I've been getting more and more emails lately from people confused about the difference between ...

Nov 08, 2022 • 93 Comments [Read](#)

For Better Agile Planning, Be Collaborative

The best plans are created by developers and stakeholders working together.

Jul 12, 2022 [Read](#)

Relationship between Definition of Done and Conditions of Satisfaction

I'd like to clarify the relationship between two important concepts: a team's Definition of Done ...

Jun 28, 2022 • 38 Comments [Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile **Agile & Scrum Training** **More...**

New to Agile and Scrum? Online Certifications About Us

Scrum Video Courses Contact Us

User Stories In-Person Certifications Our Students

Planning Poker On-Site Courses Blog

Agile Software Development Training Schedule Books by Mike Cohn

Development Compare Courses Agile Tools

Agile Project Management PDUs and SEUs

Agile Presentations

Mountain Goat on YouTube

Agile Mentors Podcast

Elements of Agile

Assessment

2 Times to Play Planning Poker and 1 Time Not To

by

Mike Tagged:

Cohn product backlog, estimating, user stories, product owner, sprint

22 planning, daily scrum, planning poker

Comments

This post continues my recent series about [Planning Poker](#) by focusing on when to estimate.

Because Planning Poker is a consensus-based estimating approach, it is most appropriate to use it on items that require consensus. That is, I recommend using Planning Poker on product backlog items rather than on the tasks that make up a sprint backlog.

Estimates on the user stories of a product backlog serve two purposes:

1. **They allow the product owner to prioritize the product backlog.** Consider two product backlog items that the product owner deems to be of equal value. The product owner will want to do first the item that has the lower cost. The estimates given to items are their costs, and so are used by the product owner to make prioritization decisions.
2. **They allow the company to make longer-term predictions.** A product owner cannot begin to answer questions like how much can be delivered by a certain date or when a set of features can be delivered without first knowing the size of the work to be performed.

When to Play Planning Poker

Knowing those two reasons for estimating product backlog items is important because it helps answer the question of when a team should estimate. A team should estimate early enough to fulfill these two needs but no earlier than necessary.

Practically, then, this means that a team should estimate with Planning Poker at two different times.

First, a team should play Planning Poker after a meeting like a story-writing workshop in which they have written a large number of product backlog items. I recommend doing such workshops about quarterly.

This allows the product owner and team to identify a larger goal than can be achieved in one sprint, and create the user stories needed to reach the goal.

A typical quarterly story-writing workshop might produce 20 to 50 product backlog items. At a target rate of estimating about 20 items per hour, this could lead to two or so hours spent estimating for each quarter.

The second time a team should play Planning Poker is once per sprint. Some teams will do this as part of a regular product backlog grooming meeting, and I think that's fine if the whole team participates in grooming.

If the whole team does not participate, another good time to play Planning Poker is following one of the daily scrums late in the sprint. If the team estimates too early in the sprint, there's a chance they'll need to repeat the process for any late-arriving stories.

But the team should avoid estimating so late in the sprint that the product owner cannot consider the newly estimated items when deciding what the team will work on in the next sprint.

A Time Not to Play Planning Poker

There's only one time when I think it's a mistake to play Planning Poker: at the start of the sprint planning meeting. The first problem with doing it then is that it's too late for the product owner to adjust priorities based on the new estimates.

Some product owners may be able to rejigger priorities on the fly, but even if they can, they'd probably do so better with a little more time to think.

A second problem with playing Planning Poker to start sprint planning is that it almost always causes the team to spend too much time estimating. I think this happens because team members walk into the sprint planning meeting ready to think in detail about their user stories.

But with Planning Poker, they are asked instead to think at a high-level and put a rougher, less precise estimates on the user stories. Many team members seem to have a hard time giving rough estimates in a meeting in which they will next be asked to give much more precise estimates of tasks.

This leads to more involved discussions than I recommend, and that leads to the team taking longer to estimate than when that is done separately. By following the guidelines here, you'll be able to help your team estimate at the two times they *should* use Planning Poker, and avoid estimating at the one time they shouldn't.

Question: When do *you* play Planning Poker?

[Get my chapters now!](#)

Posted: April 29, 2014

Tagged: product backlog, estimating, user stories, product owner, sprint planning, daily scrum, planning poker

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance

teams. He is the author of User Stories Applied for Agile Software Development, Agile Estimating and Planning, and Succeeding with Agile as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

Why I Don't Emphasize Sprint Goals

Sprint goals are considered a mandatory part of Scrum. Here's why I disagree.

Mar 21, 2023

[Read](#)

User Stories: How to Create Story Maps

Story maps help to create a shared understanding of the product, visualize user needs, and elicit ...

Mar 14, 2023

[Read](#)

How to Run a Successful User Story Writing Workshop

What you need to know to conduct a story-writing workshop with your team.

Feb 28, 2023 • 3 Comments [Read](#)

Be a Great Product Owner: Six Things Teams and Scrum Masters Need

Learn six ways effective product owners ensure their teams' success.

Jan 31, 2023 • 16 Comments [Read](#)

What Are Agile Story Points?

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

Don't Equate Story Points to Hours

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • 39 Comments [Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

Using Vertical Slicing and Estimation to Make Business Decisions at Adobe

by

Peter **Tagged:**

Green product backlog, estimating, product owner, teams, sprinting,

0 testing, customers, lean

Comments

As any regular reader of this blog knows, I advocate estimating product backlog items but only when actions will be made based on those estimates. A team shouldn't estimate just to make their boss feel better but rather when the estimates lead to actionable decisions. When I heard the following story from Peter Green of Adobe, I asked him if he'd share it with all of us. He kindly agreed. Here's his story. --Mike

I recently helped to facilitate a two-day planning session for an important initiative at Adobe that would require coordination across several component teams in multiple locations. Going in, my agile compromise-

o-meter was pretty far pegged since I felt like the teams were facing several challenges to their ability to deliver value to customers rapidly:

Vertical slicing would be difficult to accomplish within a single scrum team due to their cross-geography, component based team structure. For any readers unfamiliar with this term, see my blog post here: <http://blogs.adobe.com/agile/2013/09/27/splitting-stories-into-small-vertical-slices/>

The Product Owners had written a fairly detailed requirements document that each team had been studying and breaking down into something they were calling "engineering stories", essentially high-level component team tasks that would help achieve the end business value described in the requirements.

There was a separate integration testing team, rather than having the integration testing included in the scrum teams' definition of done.

There was no alignment in sprint boundaries or durations across the various teams, resulting in a lack of transparency when determining what features had met the definition of done at any given point.

There was no common Continuous Integration system across the technology stack, resulting in a slow feedback loop one whether changes from one team had broken the code of any other part of the system.

Putting those concerns aside and taking a pragmatic approach, we decided to focus on a few goals for the session:

Visualize the work and how it flowed across the various teams and where the dependencies existed.

Seek opportunities to deliver "done" vertical slices (even across teams) as early as possible so that the integration testing team could be engaged early and frequently.

Project some kind of timeline of delivery across all of the slices and teams.

We spent the first day and a half getting the work written on cards and up on the board in rough order of implementation, spacing it out to ensure that the order of development and dependencies were clear to all.

We then took markers and started circling sets of cards that delivered a vertical slice. We were able to identify several slices of development and cut quite a bit of scope from the original requirements documents in order to identify a Minimum Viable Product, and put the rest of the backlog items in a prioritized list for future releases.

The second afternoon we did a bunch of estimating to turn the ordered list into a projected timeline so that we could make some projections for the business. At the end of that session, we had a high level of consensus on a date to deliver those MVP slices, several months in the future. We used the "fist of five" technique to gauge individual confidence in the projected schedule (see

<http://blogs.adobe.com/agile/2013/12/17/using-the-fist-of-five-technique-to-gauge-confidence/> for more information on the Fist of Five technique). Everyone in the room was a 4 or 5 out of 5 with the exception of a few folks who were going to be (justifiably) uncomfortable with any projection of dates that far in the future prior to having started the work. I asked a few questions to test the hypothesis that the date was the earliest we could deliver that value to the business as we were currently structured, including things like "if a magic fairy bequeathed a bunch of new team members experienced in the technology upon us...". The answer kept coming back to the same date, maybe with higher confidence if we made those changes.

So I turned to the overall product owner, and said "well, with the current product backlog, it looks like Aug-Sep when we'll get the "MVP". We had a quick discussion about the fact that if new stories are discovered in the course of the work, or if anything takes longer than our guesses that we would need to push the date out further. She had a despondent look, one that she later told me was closer to nausea than despondence. She had already mentioned that she was not looking forward to the conversation she was going to have to have

with the VP about the longer-than-wished-for timeline.

The real problem was that the team's estimate project's goal was to open up a particularly important revenue stream that the business knew had customers waiting for them to deliver it. Those customers made purchases in a specific market window due to the nature of their industry, and the Product Owner said that the issue was that the current projected date for the MVP would miss that market window by a few months, making it nearly pointless to do the work in this fiscal year at all. Having participated actively in the breakdown and slicing for the two days, she understood why it would take that long; it just wasn't an acceptable business outcome. I asked her what she thought we should do about it.

At this point, we had an hour left in the two days before people started flying back to their home offices, etc. After a moment of silence, someone in the room stood up and asked: "what if we hired some contractors to manually do the work that would have encompassed slices one and two on the board, until we could get the software built to do it the right way?"

We all paused for a second, realizing we might have a path to satisfy the business, and a flurry of activity broke out. Everyone gathered around the board, started moving cards around, talking about how the new approach would work technically, validating that it was feasible. Having done the estimation and slicing, the teams were able to quickly re-plan and forecast that the "new MVP" plan could be delivered in time to meet the market window. The teams would build out the other slices to replace the manual portions directly after that, delivering in a similar time frame as the original MVP, but meeting the business needs and allowing the Product Owner to meet her revenue targets.

It occurred to me as a huge breakthrough, one that we would not have reached without the hard work of turning the requirements into vertical slices, visualizing all of the dependencies, and estimating the work.

There is often debate about the real value of estimation in software, given its inherent lack of predictability. There are arguments to be made that it is a non-value-add activity, what the lean community would call "Muda" waste. I think this experience highlights where estimation can be a value-add – when we need to make business decisions that are dependent on the work being feasible within a specific time frame to meet a market window.

I'm hopeful that such an approach will become more commonplace in the group, providing an opportunity for all of the right conversations to happen and breakthrough results to become the norm.

[Download Scrum Master Guide](#)

Get a free copy of Situational Scrum Mastering: Leading an Agile Team

[Get my free guide now!](#)

Posted: January 15, 2014

Tagged: product backlog, estimating, product owner, teams, sprinting, testing, customers, lean

About the Author

Peter Green is Adobe's Agile Transformation Leader, a Certified Scrum Trainer, and a professional musician. Peter has lead a large-scale, grass roots agile adoption at Adobe for several years, training thousands of developers and coaching hundreds of teams. Peter now leads a team of trainers and coaches that help shift Adobe's culture through focus on agility, innovation, and customer advocacy. Along with leading this transformation team, Peter's primary focus is on Leadership Agility, and is coaching directors and vice presidents in how to create an agile organization through developing their own leadership.

Follow Peter on Twitter: agilepeter@gmail.com.

For more of Peter's thoughts on agile, you can read his blog at <http://blogs.adobe.com/agile/>.

You may also be interested in:

[What Is Cross-Functional Collaboration in Agile?](#)

Perhaps the most prevalent and persistent myth in agile is that a cross-functional team is one on ...

Feb 14, 2023 • 49 Comments

[Read](#)

[Be a Great Product Owner: Six Things Teams and Scrum Masters Need](#)

Learn six ways effective product owners ensure their teams' success.

Jan 31, 2023 • [16 Comments](#)

[Read](#)

[What Happens When During a Sprint](#)

Succeeding with Scrum is easier when you know when and why to conduct each of the Scrum events ...

Jan 03, 2023 • [34 Comments](#)

[Read](#)

[What Are Agile Story Points?](#)

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • [122 Comments](#)

[Read](#)

[Don't Equate Story Points to Hours](#)

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • [39 Comments](#)

[Read](#)

Epics, Features and User Stories

I've been getting more and more emails lately from people confused about the difference between ...

Nov 08, 2022 • [93 Comments](#)

[Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

by

Mike **Tagged:**

Cohn product backlog, estimating, product owner, story points, sprint

57 planning, audio

Comments

This article has an audio version: [Download Audio Version](#)

As much as I value estimating the product backlog, not every team needs to do it. And those who do estimate the items on their product backlog need to understand why they do it, not just how to do it. There are two reasons to estimate product backlog items.

First, estimating product backlog items allows the product owner to prioritize the product backlog. An estimate represents the cost of an item. Without knowing the cost of an item, a product owner cannot successfully prioritize. If I don't know the costs of various cars, I'm putting Ferrari right up at the top of the list of cars I'd like to buy. And I'll follow it with Lamborghini and Tesla. But, whoa, those cars are expensive. Once I learn the cost of them, I adjust my priorities.

A product owner can know how much he or she wants a product backlog item without knowing the cost. But to correctly prioritize items, the product owner will eventually need to know the cost.

The second reason to estimate is so long-term projections can be made about the project. If a boss, client, or customer wants to know when a team will finish a set of product backlog items, those product backlog items need to be estimated. Similarly, if a boss, client, or customer wants to know how much can be delivered in three months, approximately three months worth of items need to be estimated.

Some product owners don't need to prioritize beyond a very simple sequencing of the work. Other product owners can take good enough guesses at the estimates that they can prioritize without bothering the team to do so. In some companies there may be no need to say where a team will be in some number of months or how long a stack of items will take to deliver. In these cases, it's not necessary to estimate the product backlog items.

There can, however, be ancillary benefits to estimating. For example, being asked to estimate forces a team to think about the work before they do and there can be value to that. (There is, though, of course also a cost to asking the team to do this.)

Knowing the reasons why to estimate product backlog items can also help fix a very common problem I'm seeing more often: Estimating the product backlog at the right time. The product backlog needs to be estimated early enough that the product owner can either prioritize or answer the above questions about when or how much will be delivered.

Putting story points on product backlog items during sprint planning is simply too late for those benefits to accrue. There are other problems with assigning story points during sprint planning but assigning them that late does not allow the product owner to take advantage of the new information in time for the sprint being planned. It's as though you wait until I'm on the Ferrari lot before telling me, "By the way, these cars are expensive." That may be early enough to stop me from making an expensive purchase. But it's not early enough for me to quickly figure out which cars I really should be looking at.

If your product owner doesn't need the estimates in order to prioritize well and if no one is asking for multi-sprint predictions of what will or can be delivered, don't bother putting story points on your product backlog items. But, if your project can benefit from these estimates, apply them earlier than during sprint planning so the product owner has time to think about and act on the new information.

[Download my PDF](#)

Posted: September 17, 2013

Tagged: product backlog, estimating, product owner, story points, sprint planning, audio

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the *Better User Stories* video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

[Why I Don't Emphasize Sprint Goals](#)
Sprint goals are considered a mandatory part of Scrum. Here's why I disagree.

[Be a Great Product Owner: Six Things Teams and Scrum Masters Need](#)
Sprint goals are considered a mandatory part of Scrum. Here's why I disagree.

[What Are Agile Story Points?](#)
Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

[Don't Equate Story Points to Hours](#)
I've been quite adamant lately that story points are about time, specifically effort.

[Epics, Features and User Stories](#)
I've been getting more and more emails lately from people confused about the

[For Better Agile Planning, Be Collaborative](#)
The best plans are created by developer and stakeholders working together.

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile **Agile & Scrum Training** **More...**

New to Agile and Scrum? Online Certifications About Us

Scrum Video Courses Contact Us

User Stories In-Person Certifications Our Students

Planning Poker On-Site Courses Blog

Agile Software Development Training Schedule Books by Mike Cohn

Development Compare Courses Agile Tools

Agile Project Management PDUs and SEUs

Agile Presentations

Mountain Goat on YouTube

Agile Mentors Podcast

Elements of Agile

Assessment

by Ilan Goldstein



Published Jul 2013

Publisher: Addison-Wesley Professional

Rather than a conventional review, here is the foreword I was asked to write for this book.

--Mike

In the spirit of this book, I'll take a shortcut and come right to the point: Buy this book. I assure you, you'll find the wisdom in this collection of shortcuts extremely helpful.

However, experience tells us to be leery of shortcuts. Very few work out. Horror movies begin when a group of teenagers take a shortcut through the woods on a dark night. The driver on a family trip opts for what looks like a shortcut and is reminded for years about how it turned out not to be. We're told "there are no shortcuts to success," and that success comes from a combination of perseverance and skill.

Yes, in life many shortcuts do not work out. The shortcuts in this book are different. They work.

I first met Ilan Goldstein online when a web search led me to his blog of Scrum shortcuts. He hadn't written

many shortcuts by then, but the few he had were tremendously helpful—and funny. Ilan's sense of humor shone through in every shortcut.

It didn't take a genius to see that Ilan was onto something with his shortcuts. And so I asked him if he'd consider writing a book of shortcuts. This book is the result. In it, Ilan offers thirty tips, covering the full gamut of a Scrum implementation. He offers tips on getting started, on requirements, on estimating and planning. There are tips about quality and metrics, about team members and roles, about managing bosses, and about continuous improvement. If you've struggled with it on a Scrum project, it's likely Ilan has a shortcut to help you.

Ilan has been there and done that. His tips come from his experience as a ScrumMaster and Certified Scrum Trainer. His shortcuts all come from routes he's traveled. They're practical, not theoretical. Further, I like that he's not afraid to take a stand. Too many books rely too often on the consultant's standard answer of "It depends." You won't find that here.

Whether you are a month, a year, or a decade into Scrum, you will find shortcuts here that will help you improve. I wish you well on your Scrum journey. I know you'll arrive more quickly by following the shortcuts in this book.

Tagged: estimating, metrics, continuous improvement, scrum master, book reviews

You may also be interested in:

[Six Attributes of a Great ScrumMaster](#)

Scrum Masters exhibit a shared set of traits. Discover 6 in-demand qualities, plus a bonus ...

Jan 17, 2023 • 32 Comments [Read](#)

[What Are Agile Story Points?](#)

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

[Don't Equate Story Points to Hours](#)

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • 39 Comments [Read](#)

[Elements of Agile: An Agile Assessment Tool for Iterative Improvements](#)

New to agile, or needing an agile re-boot? We've got a new no-cost assessment and actionable ...

Sep 13, 2022 [Read](#)

[From Project Manager to Scrum Master - 3 Tips for Making the Transition](#)

What does a good project manager need to do to become a great Scrum Master?

Aug 23, 2022 [Read](#)

[For Better Agile Planning, Be Collaborative](#)

The best plans are created by developer and stakeholders working together.

Jul 12, 2022 [Read](#)

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

Presentation by:
Mike Cohn

Tagged:
estimating, user stories, teams, story points, velocity,
management, planning poker, project management, presentations,
agile planning

In this session we will shatter the myth that agile teams can't plan. We'll start by looking at the benefits of the short cycles of iterative and incremental development. We'll then look at the six different levels of planning that occur in agile organizations. We'll see what user stories are and why they've become the preferred approach to planning and managing the work of an agile team. We'll look at how user stories are estimated with the popular Planning Poker technique.

To do that we'll discuss the merits of relative estimating and abstract approaches such as story points and what a point really means. With an initial plan created, we'll look at how agile teams use the concept of velocity to measure and predict progress. We'll also look at information radiators such as task boards and burndown charts. We'll conclude by looking at how these concepts can be applied to complex situations such as fixed-date and fixed-scope projects.

[View the Presentation](#)

[Download a PDF](#)

You may also be interested in:

User Stories: How to Create Story Maps

Story maps help to create a shared understanding of the product, visualize user needs, and elicit ...

Mar 14, 2023

[Read](#)

How to Run a Successful User Story Writing Workshop

What you need to know to conduct a story-writing workshop with your team.

Feb 28, 2023

• 3 Comments

[Read](#)

What Is Cross-Functional Collaboration in Agile?

Perhaps the most prevalent and persistent myth in agile is that a cross-functional team is one on ...

Feb 14, 2023

• 49 Comments

[Read](#)

What Are Agile Story Points?

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022

122 Comments

[Read](#)

Don't Equate Story Points to Hours

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022

39 Comments

[Read](#)

Epics, Features and User Stories

I've been getting more and more emails lately from people confused about the difference between ...

Nov 08, 2022

93 Comments

[Read](#)

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

How to Estimate Velocity As an Agile Consultant

by

Mike

Cohn

31

Tagged:

estimating, user stories, sprinting, story points, velocity

Comments

Many of you work in a dedicated in-house team, but some of you contract with companies for Scrum and agile consulting. One problem that sometimes arises as an agile contractor is when the prospective client wants an upfront commitment on the scope of the project, but the Scrum consultant feels there isn't enough info to estimate a backlog for the RFP phase. Add to that the fact that you are either estimating on behalf of other teams, or your own without prior history of the project, and it can be challenging.

So what then?

This challenge really comes down to being able to [estimate velocity](#) even in situations that aren't cut and dry. Chapter 16 in the [Agile Estimating and Planning](#) book talks about this.

To be clear, points are relative, and it's hard to compare teams, but if a team builds project A, then is about to start project B, we can use their A velocities to predict B as long as the points are consistent across projects. If a 5-point story on A is expected to take as long as a 5-point story on B, use the historical velocity.

The problems enter when a team has no velocity data.

[What to Do if You Already Have a Team Assembled for the Project](#)

Let's assume you have the team – they just don't have any velocity data. Have them [estimate the required stories](#) in story points the normal way, even if they have no idea what their velocity will be. They do know, however, that the sum of the stories is, say, 300 points.

Now, have them plan a sprint.

My preferred way to [plan a sprint](#) is to grab one story, split it into tasks, estimate the tasks in hours and ask the team if they can commit to finishing it. Then repeat with additional user stories until the sprint is full.

Suppose a team does this and commits to stories X, Y and Z. They couldn't have discussed velocity, as they have no data. But, they do have points on all stories including X, Y and Z.

So, look at the points on those three stories, add them up, and that's a starting point for velocity. If you can, get the team to do this again by planning another sprint that way and commit to M, N, O and P, which is probably a different number of points.

Use the two values as a range. Say 18 to 22 points. Maybe adjust those based on your intuition. If you think the team, like most teams, is about to overcommit in their early sprints, use estimates a few (or many) points lower based on your judgment, but based on what they came up with.

What to Do if You Don't Have a Team Ready to Go

Let's make the situation worse. Let's assume you don't even have the team yet. You've got plenty of employees, but you don't know who will do this project. Get some representative people together and have them pretend to be the team and do the steps above.

Then you can tell a boss or customer "a project with three senior programmers, a senior tester, a junior tester, a good designer and a great database person can do this project in X sprints."

Or make some adjustments based on differences in how you think you'll really staff it; change the three senior programmers to be two or more junior programmers and lower the velocity based on your expert opinion.

Using data from other teams can help. I like to make these types of predictions with velocity ranges. Suppose you have other agile teams. Calculate the relative standard deviation in velocity for those teams. (Relative StDev is just StDev but expressed as a percentage.)

Average those relative standard deviations over all teams. Say you have teams with 15 percent, 20 percent and 25 percent. That's an average of +/- 20 percent. Use that for the new team. If they estimate velocity as 20, you could add + and – 20 percent.

Often though, I'll take the estimated velocity as the high value and go down by 40 percent, just given the likelihood of teams overcommitting.

**Get 200 Real Life User Stories
Examples Written by Mike Cohn**

See user stories Mike Cohn wrote as part of several real product backlogs.

First Name

Email Address

[Privacy Policy](#)

Posted: July 25, 2013

Tagged: estimating, user stories, sprinting, story points, velocity

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

User Stories: How to Create Story Maps

Story maps help to create a shared understanding of the product, visualize user needs, and elicit ...



Mar 14, 2023

[Read](#)

How to Run a Successful User Story Writing Workshop

What you need to know to conduct a story-writing workshop with your team.

Feb 28, 2023 • 3 Comments [Read](#)

What Happens When During a Sprint

Succeeding with Scrum is easier when you know when and why to conduct each of the Scrum events ...

Jan 03, 2023 • 34 Comments [Read](#)

What Are Agile Story Points?

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

Don't Equate Story Points to Hours

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • 39 Comments [Read](#)

Epics, Features and User Stories

I've been getting more and more emails lately from people confused about the difference between ...

Nov 08, 2022 • 93 Comments [Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile

New to Agile and Scrum?

Scrum

User Stories

Planning Poker

Agile Software Development

Agile Project Management

Agile Presentations

Mountain Goat on YouTube

Agile Mentors Podcast

Elements of Agile

Assessment

Agile & Scrum Training

Online Certifications

Video Courses

In-Person Certifications

On-Site Courses

Training Schedule

Compare Courses

PDUs and SEUs

More...

About Us

Contact Us

Our Students

Blog

Books by Mike Cohn

Agile Tools

When You Miss the Point of Sprint Planning Meetings

by
Mike Cohn Tagged:
product backlog, estimating, sprint planning, audio, sprint backlog,
scrum master
[Comments](#)

This article has an audio version: [Download Audio Version](#)

In a recent interview for an upcoming agile book by Sondra Ashmore and Kristin Runyan, they asked me questions regarding several areas of agile development and Scrum. In the last two posts, I explored questions about the [product backlog](#) and [estimating](#). This time, I'd like to share my thoughts on sprint

planning meetings and the following question:

What are the most common mistakes in a sprint planning meeting?

Undoubtedly, the most common mistake in sprint planning is misunderstanding the purpose of the meeting.

There should be two things that come out of a sprint planning meeting:

1. A sprint goal
2. A [sprint backlog](#)

Because one of the artifacts of sprint planning is a sprint backlog, many teams obsess over getting the sprint backlog perfect. These teams try to identify every last task and put a perfectly precise estimate on each task.

Teams typically do this as a result of pressure from management, especially if they are new to Scrum. They want to perfectly estimate each task because they want to be able to get an accurate estimate to management.

But that's not the goal of the sprint backlog in sprint planning meetings. The goal of sprint planning is to select the right set of product backlog items to work on during the sprint, and to feel that each has been discussed enough that the team is prepared to work on it.

Focusing too much on the tasks and their estimates leads teams to spend too much time in sprint planning. So it's the ScrumMaster's role to ensure the sprint planning meeting doesn't get too far off track from its original purpose. While the tasks and the estimates are necessary in order for the team to understand what it's committing to, they are tools a team should use to decide which is the right set of product backlog items to select for a sprint.

[Download my PDF](#)

Posted: July 18, 2013

Tagged: product backlog, estimating, sprint planning, audio, sprint backlog, scrum master

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

[Why I Don't Emphasize Sprint Goals](#)

Sprint goals are considered a mandatory part of Scrum. Here's why I disagree.

Mar 21, 2023

[Read](#)

[Six Attributes of a Great ScrumMaster](#)

Scrum Masters exhibit a shared set of traits. Discover 6 in-demand qualities, plus a bonus ...

Jan 17, 2023 • [32 Comments](#)

[Read](#)

[What Are Agile Story Points?](#)

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • [122 Comments](#)

[Read](#)

[Don't Equate Story Points to Hours](#)

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • [39 Comments](#)

[Read](#)

[Epics, Features and User Stories](#)

I've been getting more and more emails lately from people confused about the difference between ...

Nov 08, 2022 • [93 Comments](#)

[Read](#)

From Project Manager to Scrum Master -3 Tips for Making the Transition

What does a good project manager need to do to become a great Scrum Master?

Aug 23, 2022

[Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

Sprint Burndown Sums All Work Remaining

by
Mike Cohn 27
Tagged: estimating, sprint backlog, burndown chart
[Comments](#)

I want address a couple of common questions surrounding sprint burndown charts:

- 1) Is the burndown adjusted based on all tasks remaining in a sprint backlog or only those tasks that are in process?
- 2) During a sprint, if a bug is discovered on a user story being worked on, should we add additional tasks to the sprint backlog and should we adjust the burndown for those tasks?

These questions and many like them ARE EASILY ANSWERED
if we remember the purpose of the sprint burndown chart. A

sprint burndown chart shows only one thing: How much work remains.

To figure out how much work remains, we need to sum the estimates of all remaining tasks of the sprint. If we're using a task board, that means we'll sum estimates for tasks in the "In Process" column along with tasks still in a "To Do" column plus tasks with work remaining in any other columns a team is using. The sprint burndown represents the sum of the work remaining in a sprint. This also means that a change in the estimate for a task still in the To Do column will be reflected in the burndown value.

Why might a team change an estimate for a task that hasn't been started? (In other words, one that is still in the To Do column.) Perhaps the unstarted task is similar to a task that has been started. In starting the one task, a team member learns that it's going to take longer than expected. If the tasks are similar, it's likely the unstarted task will also take longer than expected and so its estimate might be increased.

If a bug is discovered during a sprint, the ideal situation is for whoever finds the bug to yell to me, "Mike, there's a bug in your code when I do such-and-such" and for me to fix it immediately. In that case we wouldn't see a new task in the sprint backlog and we wouldn't see the burndown chart go up. However, if I can't fix the bug immediately (perhaps because I'm deep into something else), the bug should be added as a new task in the sprint and an estimate quickly assigned to the task. When the burndown is next calculated this extra work should be included when work remaining is summed.

The answers to the questions at the start of this post and many others like them are easily addressed when we remember that the purpose of a sprint burndown chart is to show the total amount of work remaining.

[Get my chapters now!](#)

Posted: June 19, 2013

Tagged: estimating, sprint backlog, burndown chart

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the *Better User Stories* video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

What Are Agile Story Points?

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Don't Equate Story Points to Hours

I've been quite adamant lately that story points are about time, specifically effort.

For Better Agile Planning, Be Collaborative

The best plans are created by developers and stakeholders working together.

Four Reasons Agile Teams

Estimate Product Backlog Items

Estimating product backlog items provides benefits beyond predicting when

Automatically Triangulating Estimates in Planning Poker

Comparing estimates during Planning Poker just got easier.

Estimating with Story Points

Course Available for One Week

Registrations are now open to Estimating with Story Points.

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

Estimating with Tee Shirt Sizes

by
Mike Cohn
10
[Comments](#)

Tagged:
estimating, teams, transitioning to agile

This article has an audio version: [Download Audio Version](#)

I occasionally encounter the use of t-shirt sizes (Small, Medium, Large, or so on) in use as estimating units by teams. T-shirt sizes are an OK approach to getting started with relative estimating, but they suffer from two severe weaknesses:

They aren't additive. You cannot tell a boss
you'll be done in 3 mediums, 4 larges, and 2
petites.

Your view of an XL may not match mine. You
may think it's 50% bigger than an L; I may
think 25% bigger.

Teams get around both weaknesses with some underlying assumptions (hopefully stated) about size. They
may, for example, state, "Let's call a medium a 5 and a large a 10."

The primary advantage to t-shirt sizes is the ease of getting started. If a team needs to start with t-shirt sizes,
that's fine. Later, though, that team will be better off using numbers directly. I might forget that an XL is 33%
bigger than an L. I won't forget that a 10 is twice a 5. (If I do, the team has a different problem!)

T-shirt sizes can be a great way of becoming accustomed to relative estimating. So, start with them if your
team finds that easier. But minimally put some underlying numbers on them (e.g., Medium=5) and then
gradually shift to using the numbers directly.

[Get my chapters now!](#)

Posted: April 2, 2013

Tagged: estimating, teams, transitioning to agile

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the *Better User Stories* video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

[What Is Cross-Functional Collaboration in Agile?](#)

Perhaps the most prevalent and persistent myth in agile is that a cross-functional team is one on ...

Feb 14, 2023 • 49 Comments [Read](#)

[What Are Agile Story Points?](#)

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

[Don't Equate Story Points to Hours](#)

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • 39 Comments [Read](#)

[8 Reasons Scrum Is Hard to Learn \(but Worth It\)](#)

Transitioning to Scrum is worth it, but some aspects are challenging.

Oct 11, 2022 [Read](#)

[Elements of Agile: An Agile Assessment Tool for Iterative Improvements](#)

New to agile, or needing an agile re-boot? We've got a new no-cost assessment and actionable ...

Sep 13, 2022 [Read](#)

[From Project Manager to Scrum Master -3 Tips for Making the Transition](#)

What does a good project manager need to do to become a great Scrum Master?

Aug 23, 2022 [Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile **Agile & Scrum Training** **More...**

New to Agile and Scrum? Online Certifications About Us

Scrum Video Courses Contact Us

User Stories In-Person Certifications Our Students

Planning Poker On-Site Courses Blog

Agile Software Development Training Schedule Books by Mike Cohn

Development Compare Courses Agile Tools

Agile Project Management PDUs and SEUs

Agile Presentations

Mountain Goat on YouTube

Agile Mentors Podcast

Elements of Agile

Assessment

Overheard During a Customer Conversation About Estimates

by

Mike Cohn Tagged:

Cohn product backlog, estimating, teams, story points, sprint planning,

43 velocity, metrics

Comments

I ended up in a Twitter-storm lately defending the idea that all estimation is not waste. It truly concerns me that more and more agilists seem to be saying this. I will be the first to admit that not every project benefits from estimation. But I suspect that it's a pretty rare project where no estimation at all needs to be done.

Estimating is a way of buying knowledge. If having the additional knowledge will lead to different decisions, then acquiring that knowledge might be a good thing to do. (I can't say it will always be a good thing to do because of the cost of acquiring could exceed the benefit.) Anyone who has been in one of my Certified ScrumMaster courses will

recall that during the section on sprint planning I make a point of saying that most teams spend too long estimating.

But that doesn't mean we should go to the opposite extreme and abandon estimating entirely. Whether we estimate, and how much time to spend on it, is a function of the context of the system being built. If your context doesn't require you to invest in estimates, by all means, don't estimate. But don't advise the rest of the world that they never should just because you have one example where estimating is a waste. I can give plenty of examples where estimating is wasteful. I can give more where it's vital.

The recent Twitter-storm on this, led a friend to email me snippets of a conversation he'd participated in. I asked him for permission to share those snippets and he granted it so they will form the bulk of this post. In this situation my friend ("Me" in the following) is a leader in a company doing contract development. They are in the midst of a large project and had something along the lines of the following conversation that included the ScrumMaster, two programmers who wanted to stop estimating as they found it wasteful, and their Customer, Bob.

Me: Customer Bob – The team here is thinking about doing away with estimates and focusing on building high-quality software.

Customer Bob: Don't you focus on high-quality software today? I'm confused.

Me: Well, yes, but these guys believe that we can offer you more value and you can get more functionality if we don't estimate.

Customer Bob: I see. So let me ask a few questions. Are we still on track to complete the first half of the features we identified upfront on the contract?

Programmers Pete/Patti: I'm not really sure. ScrumMaster Sam would be able to answer that.

Customer Bob: Ok, what say you ScrumMaster Sam?

ScrumMaster Sam: Well – we haven't been able to forecast anything beyond this past release because we haven't planned for the future release yet.

Customer Bob: I understand. So I'm going to need to do a pulse check this next Release to see how we're going to end-up at the end of the next quarter. That means I'm going to need to have estimates (at least gross / high-level estimates) on the Backlog.

Programmers Pete/Patti: Why don't we just follow your priorities and focus on development – they're in priority order right?

Customer Bob: Well yes. When we funded this initial increment it was based on your initial estimates. I realize they can change and I get the agile thing, which is why we've written the contract to allow you to focus on my prioritized backlog. That said, we had to do our due diligence because we've collected funds from multiple stakeholders to fund this program. They want to know what they're going to get for their money, and when they're going to get it. Regarding follow-on funding – if we estimate and determine that we're nowhere near where we said we would be then I'm going to have to answer to the stakeholders. I've been very focused on not deviating from our (minimal marketable feature) requirements because we have multiple parties to satisfy.

Programmers Pete/Patti: Well if we don't complete it all in this funding increment – can't we just tack it on to the follow on phase?

Customer Bob: *What follow-on phase?*

Programmers Pete/Patti: The next funding increment we were talking about last time.

Customer Bob: There's no assurance of doing that. If we're not able to estimate completion within a reasonable level of confidence I may have to put any subsequent work up for competitive bids.

Customer Bob: Programmer Pete/Patti - First of all, in order for me to even consider giving future funding I need to know how close our estimating process got us this go-round. When I ask the stakeholders for more money they're going to think it's going into a black hole if we only accomplish $\frac{1}{2}$ of what we said we would. I will manage this but I need to know we have a reasonable approach for coming up with high level estimates. Second – to get follow on funding, I'm going to need to know roughly how much we need to collect (this is a long lead time activity). We'll have to improve our estimates if they're nowhere near accurate and I'll beat the

pavement to get the funding. That said, money doesn't come for free. I have to have some level of metrics that tell me throughout the project that we're tracking to completion. In other words – I'll look at the project burndown and watch our costs throughout the project.

Programmers Pete/Patti: Ok, I get it that we need to do project (high-level) estimates so you know how much money to collect. Can we maybe get rid of the estimates on the Sprint Backlog (hours) and just follow our velocity. We were thinking of going to a pull-based system.

Customer Bob: Well – I'm glad you'll have a velocity now based on a gross estimating approach (story points). A pull-based system sounds nice, but it doesn't sound like it's going to give me a better sense of what we can accomplish in a sprint. Your velocity is somewhat stabilizing, but you're still committing to more work in Sprint Planning than you can accomplish. It's a false sense of security – let's bring it down and be realistic. So far, your team has done a pretty poor job of estimating during Sprint Planning. I say this because the last 4 sprints you've had to drop functionality and I haven't heard you talk about improving your estimates during the Retrospectives at all. Why do I care? I care because the projected velocity you have to meet the backlog was determined based on our MMF's. If every sprint you're coming in at 20% below that rate – guess what – by the end of the project we will not meet our MMF needs. The only way I have to know we're getting there is to track our progress and work with the SM's when I see a problem. Right now – your team is estimating during Sprints that you'll have Velocity of X and you're consistently coming in much lower. Now you want to throw that away altogether (I suppose if it's broke – throw it away instead of fix it?). I'm ok with coming in lower, but then we'll have to add more scrum teams, funding or resources to meet our MMF.

Customer Bob: Bottom line, money doesn't grow on trees. I'm willing to be flexible and accommodate change on the project – however, I need you to take estimating seriously. Being a good steward of other peoples money means we have to perform according to some reasonable set of expectations. We can reset those expectations, but if you don't take estimating seriously we'll never be able to give predictions to the funding community with any level of confidence. Since features do change, that means we need to estimate on this project.

Programmers Pete/Patti: Alright – I guess we're estimating. I didn't realize all the complexities of the "bottom line" and collecting funds from multiple parties. We'll get started on preparing the backlog to estimate this last release. Most importantly, we know what to talk about in this next retrospective.

I got permission to share this because I think it points out a variety of examples of why estimating *may* be needed on some (many, I think) projects.

In follow-up discussions with the friend who shared this, he made a very good observation: Notice that the need for estimates above is present regardless of which agile development process the team is using. It doesn't matter if they are using Scrum, XP, Kanban, etc., their customer needs to know about more than the next week or two.

So: I don't always estimate. But it's a very helpful skill to become proficient at and is useful on many, perhaps most, projects.

Download Scrum Master Guide

Get a free copy of Situational Scrum Mastering: Leading an Agile Team

[Get my free guide now!](#)

Posted: November 30, 2012

Tagged: product backlog , estimating , teams , story points , sprint planning , velocity , metrics

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

Why I Don't Emphasize Sprint Goals

Sprint goals are considered a mandatory part of Scrum. Here's why I disagree.

Mar 21, 2023

What Is Cross-Functional Collaboration in Agile?

Perhaps the most prevalent and persistent myth in agile is that a cross-functional team is one on ...

Read

What Are Agile Story Points?

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments Read

Feb 14, 2023 • 49 Comments Read

Don't Equate Story Points to Hours

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • 39 Comments Read

Epics, Features and User Stories

I've been getting more and more emails lately from people confused about the difference between ...

Nov 08, 2022 • 93 Comments Read

For Better Agile Planning, Be Collaborative

The best plans are created by developer and stakeholders working together.

Jul 12, 2022

Read

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile **Agile & Scrum Training** **More...**

New to Agile and Scrum? Online Certifications About Us

Scrum Video Courses Contact Us

User Stories In-Person Certifications Our Students

Planning Poker On-Site Courses Blog

Agile Software Development Training Schedule Books by Mike Cohn

Development Compare Courses Agile Tools

Agile Project Management PDUs and SEUs

Agile Presentations

Mountain Goat on YouTube

Agile Mentors Podcast

Elements of Agile

Assessment

Presentation by: Mike Cohn **Tagged:** product backlog, estimating, user stories, teams, sprinting, story points, backlog, sprint planning, velocity, prioritizing, presentations

You know that before the development team writes their first line of code, they need a funded project and a prioritized product backlog. What you don't yet know is how all the front-end planning works on an agile project, where just-in-time and just-enough are the rule of the day. And you aren't quite sure how to apply user stories and story points from the very start of a project up through delivery.

Take a look at these highly interactive sessions from agile expert and trainer Mike Cohn. Designed to walk you through an agile project from funding through planning the first iteration, these presentations help put theory into action by allowing participants to create case studies and work in small groups to create product backlogs. When done as workshops, these session start with writing high-level epic user stories that might be used to gain funding, continue with estimating with story points and forecasting the team's velocity, and go all the way through the end of the first iteration and how the results of that iteration affect the release plan. Learn four essential factors to consider when prioritizing. Leave with an understanding of how to establish a baseline release plan & a prioritized product backlog, write user stories that are right-sized for the first few sprints, and maintain the release plan and product backlog throughout the life of the project.

[View the Presentation](#)[Download a PDF](#)

You may also be interested in:

Why I Don't Emphasize Sprint Goals	User Stories: How to Create Story Maps	How to Run a Successful User Story Writing Workshop	What Is Cross-Functional Collaboration in Agile?	What Happens When During a Sprint	What Are Agile Story Points?	Why Goal:
Sprint goals are considered a mandatory part of Scrum. Here's why I disagree. Read	Story maps help to create a shared understanding of the product, visualize user needs, and elicit ... Read	What you need to know to conduct a story-writing workshop with your team. Read	Perhaps the most prevalent and persistent myth in agile is that a cross-functional team is one on ... Read	Succeeding with Scrum is easier when you know when and why to conduct each of the Scrum events ... Read	Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ... Read	Sprint mandates disagree. Read

Mar 21, 2023 [Read](#) Mar 14, 2023 [Read](#) Feb 28, 2023 3 Comments [Read](#) Feb 14, 2023 49 Comments [Read](#) Jan 03, 2023 34 Comments [Read](#) Dec 06, 2022 122 Comments [Read](#) Mar 21,

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations		PDUs and SEUs
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile Assessment		

Presentation by:

Mike Cohn

Tagged:

estimating, teams, planning poker, transitioning to agile, presentations, agile planning

Too many teams view planning as something to be avoided, and too many organizations view plans as something to hold against their development teams. The truth is planning is important for all projects, including agile ones. What you need to know, however, is how to create a project plan that looks forward six to nine months—and yet is accurate and useful.

In these presentations, agile expert and author Mike Cohn describes how to break the cycle of throwaway plans. Learn new skills that can help you create reliable plans for improved decision-making. Explore various approaches to estimating, including unit-less points and ideal time, and find out how Planning Poker can help your team estimate together. Leave with a better understanding of agile planning and techniques that will dramatically increase your project's chances of on-time completion.

[View the Presentation](#)[Download a PDF](#)

You may also be interested in:

What Is Cross-Functional Collaboration in Agile?

Perhaps the most prevalent and persistent myth in agile is that a cross-functional team is one on ...

Feb 14, 2023

49 Comments

[Read](#)

What Are Agile Story Points?

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022

122 Comments

[Read](#)

Don't Equate Story Points to Hours

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022

39 Comments

[Read](#)

8 Reasons Scrum Is Hard to Learn (but Worth It)

Transitioning to Scrum is worth it, but some aspects are challenging.

Oct 11, 2022

[Read](#)

Elements of Agile: An Agile Assessment Tool for Iterative Improvements

New to agile, or needing an agile re-boot? We've got a new no-cost assessment and actionable ...

Sep 13, 2022

[Read](#)

From Project Manager to Scrum Master -3 Tips for Making the Transition

What does a good project manager need to do to become a great Scrum Master?

Aug 23, 2022

[Read](#)

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

Presentation by:

Mike Cohn

Tagged:

estimating, user stories, sprinting, sprint planning, velocity, presentations

Maybe you work for a vendor who must bid for work in response to an RFP. Or perhaps your company wants to hire a software development organization for a particular project, but aren't sure how to structure a contract with an agile shop. Then again, maybe you need to outsource part of your development and need to structure a contract with an agile team. Whatever your situation, you need to know how to successfully do contract development with agile.

This presentation by agile expert and author Mike Cohn starts with a quick introduction to user stories and how to assemble them into a requirements document as part of a contract. It then moves on to cover general advice for planning projects. Explore fixed-date, fixed-scope, and fixed-everything projects. Gain insight into estimating velocity for a new team.

[View the Presentation](#)[Download a PDF](#)**You may also be interested in:**

Why I Don't Emphasize Sprint Goals

Sprint goals are considered a mandatory part of Scrum. Here's why I disagree.

Mar 21, 2023

[Read](#)

User Stories: How to Create Story Maps

Story maps help to create a shared understanding of the product, visualize user needs, and elicit ...

Mar 14, 2023

[Read](#)

How to Run a Successful User Story Writing Workshop

What you need to know to conduct a story-writing workshop with your team.

Feb 28, 2023

• 3 Comments

[Read](#)

What Happens When During a Sprint

Succeeding with Scrum is easier when you know when and why to conduct each of the Scrum events ...

Jan 03, 2023

• 34 Comments

[Read](#)

What Are Agile Story Points?

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022

• 122 Comments

[Read](#)

Don't Equate Story Points to Hours

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022

39 Comments

[Read](#)

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

Presentation by: [Mike Cohn](#) Tagged: estimating, teams, sprinting, story points, sprint planning, velocity, planning poker, presentations, video recorded, keynote

You've probably heard some people say, "Agile teams don't plan." Nothing could be further from the truth.

These sessions, including one presented to the teams at Google, blow that myth out of the water!

These presentations, by leading agile speaker and author Mike Cohn, explain how agile teams plan. Learn why story points have become the most popular unit for estimating work on agile teams. Explore how the popular [Planning Poker](#) technique, paired with story points, helps eliminate common estimating problems. Gain insights into predicting project completion using velocity and confidence intervals, including how to plan a fixed-date agile project. Leave knowing the three issues all teams need to address on large, multi-team projects.

View Mike giving this presentation at Google: [Part One](#), [Part Two](#).

[View the Presentation](#)

[Download a PDF](#)

You may also be interested in:

Why I Don't Emphasize Sprint Goals

Sprint goals are considered a mandatory part of Scrum. Here's why I disagree.

Mar 21, 2023

[Read](#)

What Is Cross-Functional Collaboration in Agile?

Perhaps the most prevalent and persistent myth in agile is that a cross-functional team is one on ...

Feb 14, 2023

• 49 Comments

[Read](#)

What Happens When During a Sprint

Succeeding with Scrum is easier when you know when and why to conduct each of the Scrum events ...

Jan 03, 2023

• 34 Comments

[Read](#)

What Are Agile Story Points?

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022

122 Comments

[Read](#)

Don't Equate Story Points to Hours

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022

39 Comments

[Read](#)

For Better Agile Planning, Be Collaborative

The best plans are created by developers and stakeholders working together.

Jul 12, 2022

[Read](#)

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

Points Are About Relative Effort Not Ranking

by
Mike **Tagged:**
Cohn product backlog, estimating, story points, project management,
21 prioritizing
[Comments](#)

I'm thinking of buying a new car. So I've put together a list of cars to consider. Here they are in priority order:

- Bugatti Veyron Super Sports
- Pagani Zonda Clinque Roadster
- Lamborghini Reventon
- McLaren F1
- Koenigsegg CCX
- Porsche Carrera GT
- Aston Martin Vanquish

Toyota Prius
 Toyota Camry
 Tata Nano

Unfortunately, though, I'm not sure I can afford my top priority car. So let me put some points on each car. I'll start with the least desirable car and put a 1 on it, a 2 on the next car, etc. That reorders our list so with points on each car we get:

1. Tata Nano
2. Toyota Camry
3. Toyota Prius
4. Aston Martin Vanquish
5. Porsche Carrera GT
6. Koenigsegg CCX
7. McLaren F1
8. Lamborghini Reventon
9. Pagani Zonda Clinque Roadster
10. Bugatti Veyron Super Sports

Now I think about my personal spending limits and I can spend between \$25-\$50k on a car. I'd like to be closer to \$25k but a good salesman might get \$40-50k out of me. Since the Tata Nano (at one point) goes for about \$2500 that means I can afford between 10-20 points. So, looking at the list again and the points assigned to each car, I think I'm going to buy a Bugatti (10 points), a Pagani (9) *and* a Tata (1 point).

Unfortunately, when I show up at the Bugatti dealer, I am somewhat informed that the Veyron lists for \$2,400,000. What went wrong here? The problem is that points are not a ranking. When we rank product backlog (or car backlog) items we use ordinal numbers (such as first, second, third). We cannot add ordinal numbers together. We cannot say that the distance between first and second is the same as from second to third. The Bugatti in this example is not ten times the cost of the Tata. Ranking stories (or cars) like this is worthless. We want story points to instead reflect the relative effort involved. For cars we could put points on as follows:

Tata Nano	1
Toyota Camry	12
Toyota Prius	14
Aston Martin Vanquish	102
Porsche Carrera GT	193
Koenigsegg CCX	218
McLaren F1	388
Lamborghini Reventon	640
Pagani Zonda Clinque Roadster	740
Bugatti Veyron Super Sports	960

These points are of course based on the relative costs of these cars. I need to be able to do the math on these estimates that someone would want to do. Someone can afford 20 points on a car—which should they buy? Should I buy this item for 10 points or those other two for 5 points each? You can't do that when points are assigned via a ranking. Story points on an agile product backlog represent the effort to implement the backlog item. Since cost on most software projects is made up almost exclusively of labor (rather than buying parts), we can think of a story point estimate on the product backlog as being the cost, as in the car example here. And, when I look at the relative costs here, I can tell I belong in the Toyota dealership rather than the Bugatti dealership.

[Download my PDF](#)

Posted: February 19, 2012

Tagged: product backlog, estimating, story points, project management, prioritizing

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

What Are Agile Story Points?

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

Don't Equate Story Points to Hours

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • 39 Comments [Read](#)

Epics, Features and User Stories

I've been getting more and more emails lately from people confused about the difference between ...

Nov 08, 2022 • 93 Comments [Read](#)

From Project Manager to Scrum Master -3 Tips for Making the Transition

What does a good project manager need to do to become a great Scrum Master?

Aug 23, 2022 [Read](#)

For Better Agile Planning, Be Collaborative

The best plans are created by developers and stakeholders working together.

Jul 12, 2022 [Read](#)

Relationship between Definition of Done and Conditions of Satisfaction

I'd like to clarify the relationship between two important concepts: a team's Definition of Done ...

Jun 28, 2022 • 38 Comments [Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile **Agile & Scrum Training** **More...**

New to Agile and Scrum? Online Certifications About Us

Scrum Video Courses Contact Us

User Stories In-Person Certifications Our Students

Planning Poker On-Site Courses Blog

Agile Software Development Training Schedule Books by Mike Cohn

Development Compare Courses Agile Tools

Agile Project Management PDUs and SEUs

Agile Presentations

Mountain Goat on YouTube

Agile Mentors Podcast

Elements of Agile

Assessment

by
Mike Cohn
38 Tagged:
estimating, user stories, product owner, sprint planning
[Comments](#)

Because I'm so interested in estimating and planning, I always take notice when I see a new blog post or news group posting claiming, "Estimating is waste! Don't do it!" The thing that never shocks me about these arguments against estimating and planning is that they never come from the business people for whom we are developing products or systems. They understand the value of estimates and plans (and the shortcomings of poor estimates and plans).

Let's consider that perspective for a moment. How many significant things in your personal life would do without at least some planning first? I doubt you would plan a wedding, a relocation to another city, a holiday trip, or any such event without first engaging in some amount of planning.

Suppose you are considering a first-ever trip to Italy. You would plan that--which cities? how long in each city? what's your budget? and so on would be some of the questions you would consider. Now suppose you are planning your one hundredth return visit to the city in which you grew up. You will even plan this trip--even if the extent of that planning is to decide you don't need to plan at all.

Planning is the act of thinking about the future. Sometimes that future holds risk and uncertainty. In those cases we plan more than when the future is highly predictable as a hundredth visit to your childhood home town would be. When a future activity is highly predictable, planning may consist of nothing more than a few milliseconds of rejecting the need to plan further.

Of course on a software project it is rarely this simple.

What about estimating? Do we really need to estimate? Yes, because estimating is a pre-requisite to planning. You cannot plan without estimates in mind. Those estimates may be very informal and very implicit. As I write this I am on a flight to California. Before boarding the plane I got cash from an ATM. I *estimated* my upcoming need for cash to do that. \$200 should do it. That estimate took less than a second and I was perhaps not even conscious of it, but it was made.

When a product owner says, "I'd prefer to add this feature rather than that feature," the product owner is acting with at least some implicit estimate (perhaps guess) of how long each will take. When a programmer chooses late in the day to fix a bug rather than start a new user story before going home, that programmer has made an implicit estimate that fixing the bug fits better with the hours left in the day.

Teams that say, "We won't estimate. We'll just make every user story the same size," are estimating. They are estimating that this user story is the same effort as all other user stories. I'd even argue that it's harder to make each story the same size than it is to use a small range of effort sizes on various stories.

These estimates must be made. Yes, they can be subconscious but they are made. Those who blog and post to newsgroups saying "estimating is waste, don't do it" are ignoring these types of estimates.

But are these casual, perhaps subconscious, estimates OK? Wouldn't teams be better with formal estimates?

Perhaps but not in all cases. A team should estimate and plan only to the extent that further investment in estimating and planning will lead to different actions. If you will do the same thing even if you estimate or plan more, stop. But if further planning is likely to lead to better decisions (more confidence in a delivery date, better prioritization of functionality, or so on), then estimate and plan further.

As an example, we recently added quite a bit of functionality to this website to support our new [eLearning course on Agile Estimating and Planning](#). I did not ask the programmer who did that work to give me more than cursory estimates. I'm still enough of a programmer to have an idea how long the new features would take. I've worked with him long enough to know how fast he is. Asking him for detailed estimates would not have changed anything about that project.

So estimating and planning are necessary. They can (and should be) lightweight. You should stop when further planning is not likely to lead to improved decisions worth the extra effort.

[Reserve a Spot](#)

Posted: February 6, 2012

Tagged: estimating, user stories, product owner, sprint planning

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

Why I Don't Emphasize Sprint Goals

Sprint goals are considered a mandatory part of Scrum. Here's why I disagree.

Mar 21, 2023

[Read](#)



User Stories: How to Create Story Maps

Story maps help to create a shared understanding of the product, visualize user needs, and elicit ...

Mar 14, 2023

[Read](#)

How to Run a Successful User Story Writing Workshop

What you need to know to conduct a story-writing workshop with your team.

Feb 28, 2023 • 3 Comments [Read](#)

Be a Great Product Owner: Six Things Teams and Scrum Masters Need

Learn six ways effective product owners ensure their teams' success.

Jan 31, 2023 • 16 Comments [Read](#)

What Are Agile Story Points?

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

Don't Equate Story Points to Hours

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • 39 Comments [Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile **Agile & Scrum Training** **More...**

New to Agile and Scrum? Online Certifications About Us

Scrum Video Courses Contact Us

User Stories In-Person Certifications Our Students

Planning Poker On-Site Courses Blog

Agile Software Development Training Schedule Books by Mike Cohn

Development Compare Courses Agile Tools

Agile Project Management PDUs and SEUs

Agile Presentations

Mountain Goat on YouTube

Agile Mentors Podcast

Elements of Agile

Assessment

Recommendations, Not Rules

by

Mike Tagged:

Cohn estimating, user stories, story points, continuous improvement,
55 project management, agile requirements, agile problems
[Comments](#)

I seem to be encountering more and more people who want to codify agile into a set of rules. I've seen this lately in authors of books, blogs or PDFs about agile or Scrum that say "You must do this" or "If you don't do this or all of that then you're not doing it right." Over the last few months I also encountered this in conversations with a few Project Management Offices (PMOs). That leads me to my new year's resolution for 2012 and one that I hope a lot of others will make with me: make recommendations not rules. There are very few hard-and-fast rules to agile software development. I'd put things like:

work in iterations of no more than a month long
by the end of each iteration be "done" with something to some pre-agreed upon definition of done and

solicit feedback from your key stakeholders on it
at the start of an iteration, get together and figure out what you're doing to do during the iteration
at the end of the iteration, reflect on how well you did during the iteration
talk a lot during the iteration

Beyond that, it's much more about recommendations. And there are plenty of things we've learned in the nearly 20 years that some agile processes have been around in even informal forms. For example, I recommend teams use user stories as their approach to requirements. I recommend teams use story points for estimating. I recommend that the team pick a day other than Mondays for starting their iterations. I recommend the Szechuan Chicken at Spice China. But, none of these things is required for success with agile. Each may help a team be better, and I have reasons I recommend each. But, these things are not required. So, my resolution for 2012: Make recommendations not rules. I'd kind of like to make it a rule that you join me in this resolution, but I've just resolved not to make such rules.

Get 200 Real Life User Stories

Examples Written by Mike Cohn

See user stories Mike Cohn wrote as part of several real product backlogs.

First Name

Email Address

[Privacy Policy](#)

Posted: January 2, 2012

Tagged: estimating, user stories, story points, continuous improvement, project management, agile requirements, agile problems

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

[User Stories: How to Create Story Maps](#)

Story maps help to create a shared understanding of the product, visualize user needs, and elicit ...

Mar 14, 2023

[Read](#)

[How to Run a Successful User Story Writing Workshop](#)

What you need to know to conduct a story-writing workshop with your team.

Feb 28, 2023 • [3 Comments](#)

[Read](#)

[What Are Agile Story Points?](#)

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • [122 Comments](#)

[Read](#)

[Don't Equate Story Points to Hours](#)

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • [39 Comments](#)

[Read](#)

[Epics, Features and User Stories](#)

I've been getting more and more emails lately from people confused about the difference between ...

Nov 08, 2022 • [93 Comments](#)

[Read](#)

8 Reasons Scrum Is Hard to Learn (but Worth It)

Transitioning to Scrum is worth it, but some aspects are challenging.

Oct 11, 2022

[Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

In Defense of Large Numbers

by
Mike Cohn
Tagged:
product backlog, estimating, user stories, story points, planning
23 poker
[Comments](#)

People are often surprised that I allow (or even encourage) people to estimate with story points as large as 20, 40, and 100. We include these values in the decks of [Planning Poker cards](#) that we sell and give away in classes and at conferences. Yet many people tell me they start out by taking the 20, 40 and 100 cards out of the deck and throwing them away. I find this unnecessary and, in some cases, detrimental to good planning. These large numbers can play a role in estimating and planning on some projects.

Let's see how. Suppose your boss wants to know the general size of a new project being considered. The boss doesn't need a perfect, very precise estimate. Something like "around a year" or "three to six months" is enough in this case. To answer this question you'll want to write a product backlog. You want to put no more

effort into this than you need to. Since the boss wants a high-level estimate, you can write a high-level product backlog. Big user stories ("epics") that describe large swaths of functionality will suffice. And these epic user stories can be estimated with the large story point values. Do you want to do this on every project? No, definitely not.

There are some projects where when the boss asks for a plan you better be close. "Heads will roll if you're wrong," the boss announces on those projects. On those projects you don't want to estimate epics and, therefore, won't use the large values. Other projects (such as contract development) won't have the tolerance for error that may come when estimating epics and using values such as 20, 40 and 100. But some projects can tolerate that amount of error in the estimates. Mis-estimating a few epics is probably not enough to change the answer we give a boss who just wants to know if we think a project can be released in Q1 or Q4 next year.

Removing the large values from a deck of Planning Poker cards is like deciding to strike "millions" and "billions" from our vocabulary just because our bank balances are only in the thousands.

[Download my PDF](#)

Posted: November 28, 2011

Tagged: product backlog, estimating, user stories, story points, planning poker

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and

can be reached at hello@mountaingoatsoftware.com.
If you want to succeed with agile, you can also have
Mike email you a short tip each week.

You may also be interested in:

User Stories: How to Create Story Maps

Story maps help to create a shared understanding of the product, visualize user needs, and elicit ...

Mar 14, 2023

[Read](#)

How to Run a Successful User Story Writing Workshop

What you need to know to conduct a story-writing workshop with your team.

Feb 28, 2023 • 3 Comments [Read](#)

What Are Agile Story Points?

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

Don't Equate Story Points to Hours

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • 39 Comments [Read](#)

Epics, Features and User Stories

I've been getting more and more emails lately from people confused about the difference between ...

Nov 08, 2022 • 93 Comments [Read](#)

Advantages of User Stories over Requirements and Use Cases

At the surface, user stories appear to have much in common with use cases and traditional ...

Oct 05, 2022 • 41 Comments [Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile

New to Agile and Scrum?

Scrum

User Stories

Planning Poker

Agile Software Development

Agile Project Management

Agile Presentations

Mountain Goat on YouTube

Agile Mentors Podcast

Elements of Agile

Assessment

Agile & Scrum Training

Online Certifications

Video Courses

In-Person Certifications

On-Site Courses

Training Schedule

Compare Courses

PDUs and SEUs

More...

About Us

Contact Us

Our Students

Blog

Books by Mike Cohn

Agile Tools

Seeing How Well a Team's Story Points Align from One to Eight

by
Mike Cohn
Tagged:
estimating, user stories, teams, story points, metrics
42 Comments

The topic of how well a team estimates two point stories relative to one point stories (and so on) has come up in a couple of comments and replies on this blog recently, so let's discuss it. Here's a graph showing relevant data from one company:

Each column of data and pair of bars shows data from stories of the size given in the Points row. The first set of data is for one-point stories, the second set of data is for two-point stories and so on. Looking at the one-point data, we see that the median number of hours to complete a one-point story (at this company) was 21 hours. The shortest (From) took 6 hours and the longest (To) took 36 hours. The shortest and longest are shown in the green and blue bars above the table. The median is graphed as the red line. Let's look at the two-point stories. We see there a median effort of 52 hours and a range from 31 to 73. If we assume the 1-point stories were perfectly estimated, we would expect 2 point stories to have a median of 42 instead of the 52 we see here. Or perhaps the 2-point stories were done perfectly and the 52 is right. In that case, the median for one-point stories should have been 26. Most likely, neither is perfect and the "perfect" estimates are somewhere between.

Either way, the team has drifted a little bit because the median of the 2-pointers does not equal twice the median of the 1-pointers. But let's look at the three-point stories. Three-point stories should presumably be triple the median of the 1-point stories. The median should be 63 and we see that it is 64. Wow. Very close. The five-point stories should presumably have a median of $5 \times 21 = 105$ and they come in at 100. The eight-point stories should presumably have a median of $8 \times 21 = 166$ but they only come in at 111. But, hold on, we have to make a bit of an adjustment in this thinking for the 5- and 8-point stories. If you use these numbers the way I recommend, you know to think of them as buckets. An 8 is a bucket that holds all stories from six through eight points in size. That is, if you have a story you think is a six, you can't fit it into the five-point bucket so it goes into the eight-point bucket. This means that our five-point stories are really fours and fives. If we have the same number of each, the average size of a story in the five-point bucket is really 4.5 points. Multiplying 4.5 points \times 21 hours (length of the 1-point story) give 94.5, off by about 5% from our measured median of 101 hours. Not so bad. Since the 8-point bucket holds stories of size 6, 7, and 8, we can assume the average story given an 8 is really worth 7 points. And 7 points times 21 hours = 147 hours. The 8-point stories should have a median of 147. They don't, they have a median observed value of 111.

So this team has definitely drifted by the time they reach their 8-point stories. (Let me add a sidenote here that rather than simply assuming the 1-point stories were perfect and using their 21 for all this multiplication, a better approach would be to do linear regression analysis, which can be done with Excel. You can then look at the r-squared value to see how well the values fit. But, I didn't want to go through all that math here and with data like this, we can, I believe, see that things work out pretty well up to 8.) I encourage you to think about collecting data like this at your company. You need to be careful though. Because you'll be

collecting actual effort expended on each user story, it's possible that team members feel more than the normal amount of pressure to finish within any estimates they give. They may then respond by padding their estimates. This defeats the whole purpose.

So, show a graph like the one above to the team and be clear that having this data can help them. For example, the team above could learn that they put 8s on stories that should perhaps have been 5s. (Looking at the data, they could also learn that they estimated correctly but that they really had more sixes in the eight bucket.) Most teams who do this will find that they are good through about 8, as in this example. With some awareness of data like this and some additional practice and calibration, just about any team can get good across a 1-13 range. Beyond that is tough and numbers above 13 (and possibly starting with 13) should be used with caution or only for answering rough, long-term questions like, "Is this project a couple of months or are we talking about a year to do it?"

If you collect data like this for your teams, I would appreciate it if you would share it with me. You can just email it to mike@mountaingoatsoftware.com rather than posting it here. I've been working on some analysis of data like this and the more teams I have, the better.

Get Free User Stories Book Chapters

Please provide your name and email and we'll send you the sample chapters and we'll send a short weekly tip from Mike on how to succeed with agile.

[Get my chapters now!](#)

Posted: September 19, 2011

Tagged: estimating, user stories, teams, story points, metrics

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the

Better User Stories video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

User Stories: How to Create Story Maps

Story maps help to create a shared understanding of the product, visualize user needs, and elicit ...

Mar 14, 2023

[Read](#)

How to Run a Successful User Story Writing Workshop

What you need to know to conduct a story-writing workshop with your team.

Feb 28, 2023 • [3 Comments](#)

[Read](#)

What Is Cross-Functional Collaboration in Agile?

Perhaps the most prevalent and persistent myth in agile is that a cross-functional team is one on ...

Feb 14, 2023 • [49 Comments](#)

[Read](#)

What Are Agile Story Points?

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • [122 Comments](#)

[Read](#)

Don't Equate Story Points to Hours

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • [39 Comments](#)

[Read](#)

Epics, Features and User Stories

I've been getting more and more emails lately from people confused about the difference between ...

Nov 08, 2022 • [93 Comments](#)

[Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Seeing How Well a Team's Story Points Align from One to Eight

Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

Estimating a Full Backlog Based on a Sample of It

by

Mike Cohn

Tagged:

product backlog, estimating, user stories, teams, sprinting, sprint

planning, velocity, release planning

Comments

I want to address a question I was sent recently and that I get asked about once a month. The question has to do with how we estimate how many hours it will take to deliver a given product backlog if we have no historical data at all. My first bit of advice is always to try to put off answering until you're able to get even one sprint of historical data. But that's not always possible. When it's not my general recommendation is to conduct one or more capacity-driven sprint planning meetings and use those to forecast likely velocity. However, some people are still more comfortable thinking in hours and rather than forecast velocity, they want to forecast the number of hours a product backlog will take. From there they will usually estimate the duration of the project (something that could be done more directly with a velocity estimate). But, since it's a common question, I'd like to address it. Here's essentially the question I was asked. I've paraphrased,

simplified and left out extraneous information:

We have no historical velocity data. We have a product backlog of 300 user stories. Each user story has been estimated, most in the range of 3-13 points. Would the following approach be reasonable:

1. Grab a random sample of 40 stories.
2. Break each of those stories into tasks and estimate the tasks.
3. From the task estimates come up with an average number of hours per story.

For example, suppose the randomly selected 40 user stories total 150 points and that the tasks identified for those 40 user stories total 600 hours. We would then theorize that it is about 4 hours per story point?

Well, yes, the idea here is fine with two problems:

1. It's important to remember that the relationship between points and hours is not an equivalence. It is not that 1 point equals 4 hours (which is what our example showed above). It is that 1 point equals a mean of four hours with a standard deviation of plus or minus say 45 minutes. This would mean that most of the time (68% of the time) the relationship would be that 1 point takes from 3:15 to 4:45 to finish. It would mean that almost always (98% of the time), one point would take between 2-1/2 and 5-1/2 hours to finish (two standard deviations).
2. The above approach assumes that 1-point stories and 13-point stories are estimated perfectly in relative terms. In other words, it assumes that if the mean duration of a one-point story is 4 hours, the mean of a 13-point story will be $13 \times 4 = 52$ hours. For many reasons, this is unlikely to be true. And the data I've collected from a variety of teams, shows that-as we'd expect--teams are not perfect, even though many are amazingly consistent.

So, what can we do to address these problems? A first really simple improvement is to calculate the average number of hours for stories of each size rather than one overall average. For example, in the example above we said that the 40 stories were 150 points in total and 600 hours for an average of 4 hours per point. But, if we averaged the 1-point stories on their own we might find that they were 3.2 hours per point, and the 2-point stories that were broken into tasks were 4.3 hours per point, and the 3-point stories were 4.1 hours, and so on. We can then multiply that average number of hours by the number of stories on the product backlog of each size. An example using the average hours given above is shown in the following table:

Points	Hours Per Story	# of Stories	Total Hours
1	3.2	5	16.0
2	8.6	8	68.8
3	12.3	7	86.1

First, notice in the above that the second column is hours per story, not hours per point. The two-point stories were assumed to take 4.3 hours per point so $8.6 (4.3 \times 2)$ hours per story is shown in that column. This table shows that we have five 1-point user stories. Each is expected to be about 3.2 hours so we expect to spend 16 hours total on the 1-point stories. Summing the last column in this table gives an expected total of 170.9 hours. Note that this approach is subject to all the shortcomings that will have been introduced during the

task identification and estimation step. Most importantly that the team will fail to identify all tasks. So this approach will estimate the number of hours to deliver the tasks identified. Some adjustment will need to be made to estimate the amount of work that the team failed to identify and the duration adjusted accordingly. I'll write about other improvements to this simple approach in upcoming blog posts.

[Take the Assessment](#)

Posted: September 16, 2011

Tagged: product backlog, estimating, user stories, teams, sprinting, sprint planning, velocity, release planning

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of User Stories Applied for Agile Software Development, Agile Estimating and Planning, and Succeeding with Agile as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

Why I Don't Emphasize Sprint Goals

Sprint goals are considered a mandatory part of Scrum. Here's why I disagree.

ad

Mar 21, 2023

[Read](#)

User Stories: How to Create Story Maps

Story maps help to create a shared understanding of the product, visualize user needs, and elicit ...

Mar 14, 2023

[Read](#)

How to Run a Successful User Story Writing Workshop

What you need to know to conduct a story-writing workshop with your team.

Feb 28, 2023 • 3 Comments [Read](#)

What Is Cross-Functional Collaboration in Agile?

Perhaps the most prevalent and persistent myth in agile is that a cross-functional team is one on ...

Feb 14, 2023 • 49 Comments [Read](#)

What Happens When During a Sprint

Succeeding with Scrum is easier when you know when and why to conduct each of the Scrum events ...

Jan 03, 2023 • 34 Comments [Read](#)

What Are Agile Story Points?

Story points are perhaps the most misunderstood topic in agile. Story point are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile **Agile & Scrum Training** **More...**

New to Agile and Scrum? Online Certifications About Us

Scrum Video Courses Contact Us

User Stories In-Person Certifications Our Students

Planning Poker On-Site Courses Blog

Agile Software Development Training Schedule Books by Mike Cohn

Development Compare Courses Agile Tools

Agile Project Management PDUs and SEUs

Agile Presentations

Mountain Goat on YouTube

Agile Mentors Podcast

Elements of Agile

Assessment

Estimating Non-Functional Requirements

by

Mike Tagged:

Cohn product backlog, estimating, user stories, product owner, sprinting,

13 story points, velocity, programming, testing

Comments

A few weeks back I promised someone I would blog about the unique challenges of estimating non-functional requirements. First, let's remember that a non-functional requirement is a requirement that is more about the state of being of the system than about one specific thing the system does. Non-functional requirements often have to do with performance, correctness, maintainability, interoperability, portability, and so on. They are often called the "-ilities" of a system because so many end in "ility." (By the way, in case you're wondering, [non-functional requirements can be written as user stories](#).) The challenge with estimating non-functional requirements is that there are really two costs. First is the cost of initial compliance. Second is the cost of ongoing compliance. To see these two costs at work, let's consider an example.

Suppose we have a performance requirement and that the team is working on a new product. In the first sprint the team may be thinking about performance but they aren't going to do any performance testing. There's no code yet. There's definitely code being developed over the next few sprints and say by sprint five the team decides there's enough code that they want to start doing some performance testing during that sprint. Remember our first cost is the cost of initial compliance. In this case, that is the amount of work the team will spend on performance testing in sprint five. That's not really much harder to estimate than most other product backlog items. The team here thinks about it and they put some number of story points or ideal days on it as their estimate.

To continue our example suppose they do the performance testing and any necessary tuning it reveals during sprint five. Well, in sprint six the team is adding some new features and here is where the second cost comes in--the cost of ongoing compliance. Once the team accepts a non-functional requirement into the project (as our team did here in sprint five) they need to remain in compliance with that non-functional requirement for the remainder of the project. I think of this cost as a tax. Doing performance testing (or staying in compliance with any non-functional requirement) creates some amount of overhead on the team (the tax). This overhead or tax must be paid regularly. In some cases, the team and product owner will decide the tax must be paid every sprint. For this team, that would mean that every time they add a feature, they do performance testing of that feature and, likely, the whole system. In other cases the team and product owner may agree to pay the tax every few sprints. After all, a team might reason, it's unlikely we affected the performance characteristics with the user stories we added this sprint and, besides, we aren't really shipping after this sprint. The first of these cases, in which the team pays the tax every sprint can be thought of like as a sales tax or VAT. The second, in which the team pays every few sprints, is more like an estimated quarterly tax like independent contractors in the US pay.

So, back to the issue of how do we estimate this type of work? Well, estimate both parts separately. Estimate the cost of initial compliance just like any other user story or product backlog item. The team and product owner will need to incorporate an estimate of when they'll do this. Adding performance testing after five sprints is different than adding it after 20 sprints. For the tax portion, the team and product owner need to agree on when they will do the work: every sprint or after every n sprints. The team can then estimate how much work will be involved over the planned number of sprints and allocate that amount of work to each. For example, suppose the team and product owner agree that they will do performance testing in every fourth two-week sprint. The team then estimates that it will take six points of work every fourth sprint. That's about 1.5 points per sprint. If a team has a velocity of 30, 1.5 can be thought of as about a 5% tax. It's likely that the team could be quite wrong on this estimate the first time. Fortunately, the team can easily track how much effort goes into performance testing over a number of sprints and use that to revise their estimate of the ongoing cost.

Get Free User Stories Book Chapters

Please provide your name and email and we'll send you the sample chapters and we'll send a short weekly tip from Mike on how to succeed with agile.

[Get my chapters now!](#)

Posted: June 19, 2011

Tagged: product backlog, estimating, user stories, product owner, sprinting, story points, velocity, programming, testing

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of User Stories Applied for Agile Software Development, Agile Estimating and Planning, and Succeeding with Agile as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

User Stories: How to Create Story Maps

Story maps help to create a shared understanding of the product, visualize user needs, and elicit ...

 Mar 14, 2023 [Read](#)

How to Run a Successful User Story Writing Workshop

What you need to know to conduct a story-writing workshop with your team.

Feb 28, 2023 • 3 Comments [Read](#)

Be a Great Product Owner: Six Things Teams and Scrum Masters Need

Learn six ways effective product owners ensure their teams' success.

Jan 31, 2023 • 16 Comments [Read](#)

What Happens When During a Sprint

Succeeding with Scrum is easier when you know when and why to conduct each of the Scrum events ...

Jan 03, 2023 • 34 Comments [Read](#)

What Are Agile Story Points?

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

Don't Equate Story Points to Hours

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • 39 Comments [Re](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile **Agile & Scrum Training** **More...**

New to Agile and Scrum? Online Certifications About Us

Scrum Video Courses Contact Us

User Stories In-Person Certifications Our Students

Planning Poker On-Site Courses Blog

Agile Software Development Training Schedule Books by Mike Cohn

Development Compare Courses Agile Tools

Agile Project Management PDUs and SEUs

Agile Presentations

Mountain Goat on YouTube

Agile Mentors Podcast

Elements of Agile

Assessment

by

Mike **Tagged:**

Cohn estimating, product owner, sprinting, story points, continuous

80 improvement, defect management, taskboard

Comments

Sometimes teams write a user story for this activity such as: "As a user, I want at least 15 bugs fixed" or, "As a user, I want you to spend about 50 hours this sprint fixing bugs so that the application becomes gradually more high quality." Even a team that doesn't explicitly write such a user story will usually include a row on its taskboard to make the agile defects and bug fixing visible, and to track it.

In these situations, a common question is whether the team should assign some number of story points to the work of fixing these legacy bugs. If the team does not assign a story point value to this work, velocity will show the amount of "forward progress" the team is making each sprint. This has the advantage of making visible that the team is going more slowly through the work than it could if these legacy bugs had been fixed when originally found. On the other hand, if the team does assign points to the agile defect bug-fixing effort, velocity comes to represent the team's true capacity to accomplish work.

When migrating a product from a traditional approach to an agile one, teams commonly bring along a large database of agile defects. These are the result of an inadequate focus on continuous high quality. Shortly into their agile initiative, many teams (and their product owners) decide to aggressively work through the bug backlog. A common approach for doing so is to plan to fix X bugs per sprint or spend Y hours on bugs per sprint.

My usual recommendation is to assign points to bug fixing the agile defects. This really achieves the best of both worlds. We are able to see how much work the team is really able to accomplish, but also able to look at

the historical data and see how much went into the bug-fixing story each sprint.

Knowing this can be helpful to a team and its product owner. For example, imagine a situation in which the product owner is considering suspending bug fixing for the next six sprints in order to add a valuable different feature into a release. If we know that the team's full historical average velocity was 25 but that 5 points went to bug fixing each sprint, we know that suspending bug fixing for the next six sprints will result in 30 (6×5) more points of new functionality.

[Get my chapters now!](#)

Posted: November 13, 2010

Tagged: estimating, product owner, sprinting, story points, continuous improvement, defect management, taskboard

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

[Be a Great Product Owner: Six Things Teams and Scrum Masters Need](#)

Learn six ways effective product owners ensure their teams' success.

Jan 31, 2023 • 16 Comments

[Read](#)

What Happens When During a Sprint

Succeeding with Scrum is easier when you know when and why to conduct each of the Scrum events ...

Jan 03, 2023 • [34 Comments](#)

[Read](#)

What Are Agile Story Points?

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • [122 Comments](#)

[Read](#)

[Don't Equate Story Points to Hours](#)

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • [39 Comments](#)

[Read](#)

[Elements of Agile: An Agile Assessment Tool for Iterative Improvements](#)

New to agile, or needing an agile re-boot? We've got a new no-cost assessment and actionable ...

Sep 13, 2022

[Read](#)

For Better Agile Planning, Be Collaborative

The best plans are created by developers and stakeholders working together.

Jul 12, 2022

[Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

The Problems with Estimating Business Value

by
Mike Cohn Tagged:
estimating, user stories, sprinting, story points
44 Comments

I occasionally see teams that want to put an estimate of "business value" on each user story. They usually do this for either or both of two reasons:

1. to be able to measure the amount of "business value" delivered to the organization, usually graphing this sprint by sprint
2. to be able to prioritize user stories by comparing the business value of each story to its cost

There are, unfortunately, some problems with these practices. First, the value of small bits of functionality are often intertwined. When we get features that are too small (as good user stories are), it is very difficult to put a discrete value on each. Economists call this hedonic pricing. The classic example is putting values on the

individual components of a house's value---size, location, view, etc. Second, the value of a small bit of functionality can often be said to be the total value of the product. For example, what is the value of the left front wheel of a car? Well, since I don't want a car without that wheel, the value of it is the total value of the car. Having identified two problems with assessing the "business value" of a user story, let's look at a problem with determining the cost of the story.

The Problem of Shared Cost

Finally, when attempting to put business value points on small features (such as user stories) there is the additional problem of determining the cost of the feature. There is often a desire to do some form of return on investment (ROI) analysis on features by comparing the business value points to the cost in story points. However, with small features it is often the case that the true cost of implementing a feature is the sum of the cost of implementing multiple smaller parts of the system, some of which (e.g., architectural work) may not be separately estimated. For example, the cost of refunding money to a credit card purchase should include some of the infrastructural work done to accept credit cards in the first place. That cost, however, was likely estimated as part of the story to "buy the items in my shopping cart." Failure to share the cost of this credit card processing infrastructure between both the purchase and refund stories will lead to incorrect ROI decisions. As with hedonic pricing, this is something economists have wrestled with for years. A simple example is a cow raised and sold for beef and leather. How should the costs of raising the cow be apportioned between the user stories of "beef" and "leather"? We could say the beef cost it all and we get the leather for free but that would be no more correct than the opposite. The benefits (beef and leather) need to be considered together in comparison to the cost of raising the cow.

What Do We Do?

So does this mean that we should never assign business value to features and that we should forego ROI analysis of user stories? Not necessarily. But this type of analysis is best used at the level of epics (large user stories) and themes (groups of stories) because value and cost can be appropriately assessed at those levels.

Get Your Free Scrum Cheat Sheet

Sign up below to receive this FREE reference.

First Name

Email Address

We hate spam and promise to keep your email address safe. Unsubscribe at any time. [Privacy Policy](#)

Posted: September 19, 2010

Tagged: estimating, user stories, sprinting, story points

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

User Stories: How to Create Story Maps	How to Run a Successful User Story Writing Workshop	What Happens When During a Sprint	What Are Agile Story Points?	Don't Equate Story Points to Hours	Epics, Features and User Stories
Story maps help to create a shared understanding of the product, visualize user needs, and elicit ...	What you need to know to conduct a story-writing workshop with your team.	Succeeding with Scrum is easier when you know when and why to conduct each of the Scrum events ...	Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...	I've been quite adamant lately that story points are about time, specifically effort. But that does ...	I've been getting more and more emails lately from people confused about the difference between ...

d Mar 14, 2023

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile Assessment		

Estimating Work Shared Between Two Backlog Items

by
Mike Cohn Tagged:
product backlog, estimating, product owner, agile project
management
[Comments](#)

Product backlog items can be ideally written to be independent. It is the hallmark of a good team that its members can implement product backlog items in any order. However, it would be nearly impossible to remove all dependencies between product backlog items and so our goal becomes minimizing important dependencies rather than eliminating them altogether. But any dependencies that remain can raise questions about how to estimate the individual product backlog items. One common agile project management question is what to do with work that could be performed as part of either of two backlog items but that only needs to be done once. As an example, consider these two product backlog items, which are

written as user stories:

As a user, I can add an invoice to the system so that it gets paid.

As a user, I can delete an invoice.

These two user stories share some work. Supposing this is a very simple system, whichever one of these stories is done first will also involve the team much of the database design for invoices. Some database design work (a cascading delete between Invoice and Line_Items, for example) will be done only when a specific one of the stories is developed. But, some work (deciding that a table called Invoice exists at all, perhaps) will be done as part of whichever story is implemented first. Assuming that this work is non-trivial, the estimates given to these stories will be influenced by which will be implemented first. In deciding how to estimate these and how to handle the common work between them, I suggest there are two guiding principles we can rely on:

We should assume work will be done in its natural order.

The sum of all estimates on the product backlog should represent the total size of the project (as understood at that time, of course).

Let's see how these two principles lead to the answer about how to estimate our two user stories about adding and deleting invoices. Although a good agile team should be able to implement these stories in either order, it is most likely that the product owner will ask the team to add invoices first. After all, you can't delete an invoice until it has been added. This is what I mean by the natural order. Although it is natural to add invoices to a database before deleting them, it is not hard to imagine a need to do these in the opposite order.

Suppose our team is writing a new system to interface with an existing invoice database. That existing system is full of old invoices from the 1940s and the first thing the product owner wants is the ability to delete old invoices. When a product wants stories done in the natural order or when it seems likely the product owner will want them done in that order, my advice is that the team should estimate with that assumption in mind and not bother noting it on the story cards. Documenting all logical assumptions about the natural order gets tedious. To document all such assumptions, our add an invoice story would also need to be documented with "assumes login story is done first," and so on. But when a team knows or suspects they will be asked to work out of the natural order, they should document the assumption. If the team assumes that adding an invoice will be quicker to implement because deleting is already done, they should add appropriate notes to these story cards.

Teams are, of course, sometimes wrong in their assumptions or are surprised by product owners who change their minds. These situations are resolved by simple discussion along the lines of saying "We assumed you'd want these stories in a different order. So, a few points [or ideal days] of work move off this story and onto that story." It is even possible that working outside the natural order can sometimes increase the total size of the work.

For example, to do the delete invoices user story before the add invoices story, our team might need to write a short script to preload the database with dummy data just so we can be sure delete is working. Usually when this happens the change in project size is very small and washes out in the noise of other estimating and planning errors. (In other words, it's rarely worse than someone being out sick for a day.) Why not get around this problem of potentially moving points from one story to another by assuming that each story is done first, which would be a worst case for its estimate? This is where our second principle comes in: The sum of the estimates should add up to the total size of work being considered. If the total work is 8 and that's what we'd put on a single add+delete user story then our two user stories need a total of 8. To put a higher

number on them overstates the size of the project. This would lead to incorrect projections of the project completion date based on velocity.

[Download my PDF](#)

Posted: July 14, 2010

Tagged: product backlog, estimating, product owner, agile project management

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the *Better User Stories* video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

[Be a Great Product Owner: Six Things Teams and Scrum Masters Need](#)

Learn six ways effective product owners ensure their teams' success.

Jan 31, 2023 • 16 Comments

[Read](#)

[What Are Agile Story Points?](#)

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • [122 Comments](#)

[Read](#)

[Don't Equate Story Points to Hours](#)

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • [39 Comments](#)

[Read](#)

Epics, Features and User Stories

I've been getting more and more emails lately from people confused about the difference between ...

Nov 08, 2022 • [93 Comments](#)

[Read](#)

For Better Agile Planning, Be Collaborative

The best plans are created by developers and stakeholders working together.

Jul 12, 2022

[Read](#)

Relationship between Definition of Done and Conditions of Satisfaction

I'd like to clarify the relationship between two important concepts: a team's Definition of Done ...

Jun 28, 2022 • [38 Comments](#)

[Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

Separate Estimating from Committing

by
Mike Cohn
19
[Comments](#)

Tagged:
estimating, teams, management, customers

A fundamental and common problem in many organizations is that estimates and commitments are considered equivalent. A development team (agile or not) estimates that delivering a desired set of capabilities will take seven months with the available resources. Team members provide this estimate to their manager who passes the estimate along to a vice president who informs the client. And in some cases the estimate is cut along the way to provide the team with a “stretch goal.”

The problem here is not the team’s estimate of seven months is right or wrong. The problem is the estimate was turned into a commitment. “We estimate this will take seven months” was translated into “We commit to finishing in seven months.” Estimating and committing are both important, but they should be viewed as

separate activities.

I need to pick up my daughter from swim practice tonight. I asked her what time she'd be done (which we defined as finished swimming, showered, and ready to go home). She said, "I should be ready by 5:15." That was her estimate. If I had asked for a firm commitment—be outside the facility by the stated time or I'll drive away without you—she might have committed to 5:25 to allow herself time to recover from any problems, such as a slightly longer practice, the coach's watch being off by five minutes, a line at the showers, and so on. To determine a time she could commit to, my daughter would still have formed an estimate. But rather than telling me her estimate directly, she would have converted it into a deadline she could commit to.

Don't let your estimates become commitments. Remember the difference between an estimate and a commitment and keep the two activities separate, educating management and customers as necessary. I talk much more about agile estimating and committing in my new book, *Succeeding with Agile*.

Get Your Free Scrum Cheat Sheet

Sign up below to receive this FREE reference.

First Name

Email Address

[Privacy Policy](#)

Posted: February 9, 2010

Tagged: estimating, teams, management, customers

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the *Better User Stories* video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

[What Is Cross-Functional Collaboration in Agile?](#)

Perhaps the most prevalent and persistent myth in agile is that a cross-functional team is one on ...

Feb 14, 2023 • 49 Comments [Read](#)

[What Are Agile Story Points?](#)

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

[Don't Equate Story Points to Hours](#)

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • 39 Comments [Read](#)

[For Better Agile Planning, Be Collaborative](#)

The best plans are created by developers and stakeholders working together. But that does ...

Jul 12, 2022 [Read](#)

[Relationship between Definition of Done and Conditions of Satisfaction](#)

I'd like to clarify the relationship between two important concepts: a team's Definition of Done ...

Jun 28, 2022 • 38 Comments [Read](#)

[Four Reasons Agile Teams Estimate Product Backlog Item](#)

Estimating product backlog items provides benefits beyond predicting when a project will be finished.

May 17, 2022 [Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile **Agile & Scrum Training** **More...**

- | | | |
|----------------------------|--------------------------|--------------------|
| New to Agile and Scrum? | Online Certifications | About Us |
| Scrum | Video Courses | Contact Us |
| User Stories | In-Person Certifications | Our Students |
| Planning Poker | On-Site Courses | Blog |
| Agile Software Development | Training Schedule | Books by Mike Cohn |
| Agile Project Management | Compare Courses | Agile Tools |
| Agile Presentations | PDUs and SEUs | |
| Mountain Goat on YouTube | | |
| Agile Mentors Podcast | | |
| Elements of Agile | | |
| Assessment | | |

The Benefits of Feature Teams

by
Mike Cohn 27
Tagged: estimating, teams, sprinting, teamwork, agile project management
[Comments](#)

Moving away from component teams is a difficult but necessary step for those who want to adopt an [agile project management](#) approach. For example, when I first began to consult for a certain California-based game studio, its teams were organized around the specific elements and objects that would exist in the video game it was developing. There was a separate team for each character. There were weapons teams, a vehicle team, and so on. This led to problems, such as weapons too weak to kill the monsters, colors too dark to show secret passages, and obstacles that frustrated even the most patient player. The studio clearly needed to change its team structure. On more traditional, corporate projects, we see equivalent problems when teams organize around the layers of an application, including

- Reduced communication across the layers
- A feeling that design by contract is sufficient
- Ending sprints without a potentially shippable product increment

If structuring teams around the layers of an architecture is the wrong approach, what's better? Rather than organizing around components, each team on a project can ideally be responsible for end-to-end delivery of working (tested) features. There are many advantages to organizing multiteam projects into feature teams:

Feature teams are better able to evaluate the impact of design decisions. At the end of a sprint, a feature team will have built end-to-end functionality, traversing all levels of the technology stack of the application. This maximizes members' learning about the product design decisions they made (Do users like the functionality as developed?) and about technical design decisions (How well did this implementation approach work for us?)

Feature teams reduce waste created by hand-offs. Handing work from one group or individual to another is wasteful. In the case of a component team, there is the risk that too much or too little functionality will have been developed, that the wrong functionality has been developed, that some of the functionality is no longer needed, and so on.

It ensures that the right people are talking. Because a feature team includes all skills needed to go from idea to running, tested feature, it ensures that the individuals with those skills communicate at least daily.

Component teams create risk to the schedule. The work of a component team is valuable only after it has been integrated into the product by a feature team. The effort to integrate the component team's work must be estimated by the feature team, whether it will occur in the same sprint during which it is developed (as is best) or in a later sprint. Estimating this type of effort is difficult because it requires the feature team to estimate the integration work without knowing the quality of the component.

It keeps the focus on delivering features. It can be tempting for a team to fall back into its pre-Scrum habits. Organizing teams around the delivery of features, rather than around architectural elements or technologies, serves as a constant reminder of Scrum's focus on delivering features in each sprint.

Of course, there will be occasions when creating a component team is still appropriate, for example when a new capability will be used by multiple teams or when the risk of multiple solutions being developed for the same problem is high. Overall, however, the vast majority of teams on a large project should be feature teams. Additional advice on feature and component teams can be found in Chapter 10, "Team Structure," of *Succeeding with Agile*.

[Take the Assessment](#)

Posted: December 7, 2009

Tagged: estimating, teams, sprinting, teamwork, agile project management

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of User Stories Applied for Agile Software Development, Agile Estimating and Planning, and Succeeding with Agile as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

What Is Cross-Functional Collaboration in Agile?

Perhaps the most prevalent and persistent myth in agile is that a cross-functional team is one on ...

Feb 14, 2023 • 49 Comments [Read](#)

What Happens When During a Sprint

Succeeding with Scrum is easier when you know when and why to conduct each of the Scrum events ...

Jan 03, 2023 • 34 Comments [Read](#)

What Are Agile Story Points?

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

Don't Equate Story Points to Hours

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • 39 Comments [Read](#)

For Better Agile Planning, Be Collaborative

The best plans are created by developers and stakeholders working together.

Jul 12, 2022 [Read](#)

Relationship between Definition of Done and Conditions of Satisfaction

I'd like to clarify the relationship between two important concepts: a team's Definition of Done ...

Jun 28, 2022 • 38 Comments [Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile

New to Agile and Scrum?

Scrum

User Stories

Planning Poker

Agile Software Development

Agile Project Management

Agile Presentations

Mountain Goat on YouTube

Agile Mentors Podcast

Elements of Agile

Assessment

Agile & Scrum Training

Online Certifications

Video Courses

In-Person Certifications

On-Site Courses

Training Schedule

Compare Courses

PDUs and SEUs

More...

About Us

Contact Us

Our Students

Blog

Books by Mike Cohn

Agile Tools

Clarifying the Purpose of Iteration Planning

by
Mike Cohn
Tagged:
product backlog, estimating, teams, sprint planning, iterations,
iterating, communication, planning meeting
[Comments](#)

I was recently asked whether it would be OK for a team to complete their iteration planning without going to the point of estimating tasks in hours. To answer this, let's consider what the purpose of iteration planning is. In my view, the purpose of the iteration planning is for the team to arrive at a commitment to some set of functionality that they feel reasonably confident they can complete in the coming iteration. The purpose is not to identify tasks.

The purpose is not to estimate the number of hours for each of those tasks. The purpose is to figure out how many and which product backlog items they can commit to delivering in the coming iteration. If a team can do this without discussing tasks and hours, so be it. If a team can walk into the iteration planning meeting and

one team member announces, "Sssh, I've receiving divine inspiration..... We can do the first three items plus the sixth on the product backlog." I would call the meeting over and we'd have our commitment. I'd be impressed and I'd be skeptical, but we'd be done with the meeting.

Since I haven't seen a team do this yet, I strongly recommend that an iteration planning meeting involve identifying tasks, estimating the effort of each, and then using that information to arrive at a set of product backlog items the team can commit to. I think this is the best way for a team to arrive at that reasonable commitment that I believe is the purpose of this meeting. Is it the only way? No. I've also worked with teams who decide that they will split work up such that all tasks are "about half a day". They then can skip explicitly estimating the hours for each task because, in a way they've already done it. But, until I learn how to receive divine inspiration at the start of my iteration planning meetings, I'm sticking with identifying tasks as hours as a good way to figure out how much to commit to.

[Take the Assessment](#)

Posted: November 24, 2008

Tagged: product backlog, estimating, teams, sprint planning, iterations, iterating, communication, planning meeting

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and

can be reached at hello@mountaingoatsoftware.com.
If you want to succeed with agile, you can also have
Mike email you a short tip each week.

You may also be interested in:

Why I Don't Emphasize Sprint Goals

Sprint goals are considered a mandatory part of Scrum. Here's why I disagree.

Mar 21, 2023

What Is Cross-Functional Collaboration in Agile?

Perhaps the most prevalent and persistent myth in agile is that a cross-functional team is one on ...

Read

Feb 14, 2023 • 49 Comments [Read](#)

What Are Agile Story Points?

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

Don't Equate Story Points to Hours

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • 39 Comments [Read](#)

Epics, Features and User Stories

I've been getting more and more emails lately from people confused about the difference between ...

Nov 08, 2022 • 93 Comments [Read](#)

7 Ways to Get and Improve Fast Feedback

Here's how to get the rapid feedback you need to ensure you build the right product.

Jul 26, 2022 [Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile

New to Agile and Scrum?

Scrum

User Stories

Planning Poker

Agile Software Development

Agile Project Management

Agile Presentations

Mountain Goat on YouTube

Agile Mentors Podcast

Elements of Agile

Assessment

Agile & Scrum Training

Online Certifications

Video Courses

In-Person Certifications

On-Site Courses

Training Schedule

Compare Courses

PDUs and SEUs

More...

About Us

Contact Us

Our Students

Blog

Books by Mike Cohn

Agile Tools

Is It a Good Idea to Establish a Common Baseline for Story Points?

by

Mike Tagged:

Cohn estimating, story points, velocity, planning poker, metrics, project management, iterations, iterating, communication, large teams
37 Comments

In a [previous post](#), I wrote about how to establish a common baseline for story points across relatively large teams (a few hundred developers). In this post I want to consider whether doing so is a good idea. The need for a common baseline to story points usually arises from the reasonable desire to know how big the entire project is. To know that, we must know the size of the work to be done by each team.

Unfortunately, along with this goal comes the ability to compare teams based on their velocities. Since many managers are constantly looking for ways to compare team and individual performance it is not surprising

that they begin to make such velocity comparisons. Almost all such comparisons are disruptive to performance of the combined, overall group or department. A chart such as the one that follows can show a lot of interesting information.

Velocities before teams told they would be compared

However, this chart can be very dangerous because of how teams will assume the data is being interpreted. Shown a chart like this a common team response will be to feel that they need to faster than the other teams. Achieving this additional speed may come from working in a more focused manner (a good thing), but it may come instead from sacrificing quality, leaving important refactorings undone, or a variety of other not-so-good manners.

Some teams may respond to the pressure for their abstract measure of velocity to increase by gradually inflating the number of story points assigned to a story. This can happen in subtle and not particularly nefarious ways that can accumulate into large problems. Consider, for example, a team that is arguing over whether a particular story should be estimated at 5 or 8 points. If the team is under pressure (real or just perceived) to increase velocity they will be more likely to assign the 8.

The next story the team considers is slightly larger. They compare it to the newly assigned 8 and decide to give it a 13. Without pressure to improve velocity, this same team may have given the first item a 5 and the second (slightly larger still) item an 8. In this one scenario the team has inflated their points from $5+8=13$ to $8+13=21$, or more than 50%. Story point inflation such as this tends to happen very quickly if it happens at all. Consider what happened in the next few iterations for the four teams shown in the previous figure.

Four teams and their velocities

Not surprisingly, someone in the Project Management Office distributed the chart showing the similarities over the first three iterations. Two of the teams reacted by instantly inflating their story points. After seeing that, the yellow team followed suit. The green team is either extremely virtuous or they haven't noticed the charts yet.

So, should you establish a common baseline? Yes, if there are advantages to doing so on your project. If you do, however, you need to make sure you go out of your way to create safety around that baseline for the teams. Stress that this isn't being done as a way to compare teams and that you (and your bosses know) that there are many factors that influence velocity, not just "how good" a team.

For more on agile estimating and planning, check out [Planning Poker](#).

[Get my chapters now!](#)

Posted: August 9, 2008

Tagged: estimating, story points, velocity, planning poker, metrics, project management, iterations, iterating, communication, large teams

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of User Stories Applied for Agile Software Development, Agile Estimating and Planning, and Succeeding with Agile as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

[What Are Agile Story Points?](#)

; Story points are perhaps the most

[Don't Equate Story Points to Hours](#)

;

[From Project Manager to Scrum Master - 3 Tips for Making the](#)

;

[7 Ways to Get and Improve Fast Feedback](#)

;

[For Better Agile Planning, Be Collaborative](#)

;

[Four Reasons Agile Teams Estimate Product Backlog Item](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile **Agile & Scrum Training** **More...**

New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile Assessment		

Establishing a Common Baseline for Story Points

by
Mike Cohn
Tagged:
product backlog, estimating, story points, teamwork, planning
43 poker, large teams
[Comments](#)

A common criticism of story points is that the meaning of a story point will differ among teams. In this post I want to describe how we can establish a common definition of a story point across multiple teams within an organization.

The best way I've found to do this is to bring a broad group of individuals representing various teams together and have them estimate a dozen or so product backlog items (ideally in the form of user stories in my opinion). Not each estimator needs to understand every item but most people should understand most items. The items being estimated do not need to be new items; some could be from a project finished recently that many estimators remember or worked on. Some items could be artificial; perhaps the team is

asked to estimate, "a typical transaction activity report." If that meant something to most estimators, it would be a good candidate item. I've done with this 46 people in a large conference room--44 estimators plus me and a coach from my client who wanted to watch so he could moderate such a meeting the next time one would be needed. The 44 estimators represented 22 teams; two estimators per team were in the meeting.

If you've seen or used the [Mountain Goat planning poker cards](#), you'll have noticed that they feature a very large number in the middle (plus the number in a smaller font in the corners). We could have done something cute like put eight little goats on the eight card. We put the very large number there deliberately, though: We wanted it to be visible across a potentially large conference room. You can probably imagine how difficult it might be to gain consensus among 46 people playing [planning poker](#). While it will not take proportionately longer to derive estimates, it does take quite awhile with that many people. I think it took us about two hours to estimate twelve items. But when that meeting was over, each pair of estimators went back to their teams with twelve estimates. Those estimates could then be used as the basis for estimating future work. As each team estimated new product backlog items they would do so by comparing them to the initial 12 plus any estimates that had been produced since (by them or any other team).

To see if it is a good idea to establish a common baseline, you can read this [Is It a Good Idea to Establish a Common Baseline for Story Points?](#)

[Download my PDF](#)

Posted: August 6, 2008

Tagged: product backlog, estimating, story points, teamwork, planning poker, large teams

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied* for

Agile Software Development, Agile Estimating and Planning, and Succeeding with Agile as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

[What Are Agile Story Points?](#)

I Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

[Don't Equate Story Points to Hours](#)

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • 39 Comments [Read](#)

[Epics, Features and User Stories](#)

I've been getting more and more emails lately from people confused about the difference between ...

Nov 08, 2022 • 93 Comments [Read](#)

[For Better Agile Planning, Be Collaborative](#)

The best plans are created by developers and stakeholders working together.

Jul 12, 2022 [Read](#)

[Relationship between Definition of Done and Conditions of Satisfaction](#)

I'd like to clarify the relationship between two important concepts: a team's Definition of Done ...

Jun 28, 2022 • 38 Comments [Read](#)

[Scrum, Remote Teams, & Success: Five Ways to Have A Three](#)

In a remote-first world, how does an agile team thrive working in a virtual, distributed way?

Jun 07, 2022 [Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

by

Mike **Tagged:**

Cohn product backlog, estimating, story points, audio, sprint backlog,
152 release planning

Comments

This article has an audio version: [Download Audio Version](#)

As described in [Agile Estimating and Planning](#), I'm a huge fan of using story points for estimating the [product backlog](#). However, I also recommend estimating the [sprint backlog](#) in hours rather than in points. Why this seeming contradiction? I've [previously blogged on the reasons why](#) I recommend using different estimation units (points and hours) for the different backlogs. But I'm often asked this related question I want to address here:

I'm curious why you aren't using story points to do your sprint planning. I thought that the point of measuring story point velocity was partly to determine how much we can take on (or commit to) in a sprint. Do you only use story points for longer-term planning (e.g. release planning)?

I don't use story points for sprint planning because story points are a useful long-term measure. They are not useful in the short-term. It would be appropriate for a team to say "We have an average velocity of 20 story points and we have 6 sprints left; therefore we will finish about 120 points in those six sprints." It would be inappropriate for a team to say, "We have an average velocity of 20 story points so we will finish in the next sprint." It doesn't work that way. Suppose a basketball team is in the middle of their season. They've scored an average of 98 points per game through the 41 games thus far. It would be appropriate for them to say "We will probably average 98 points per game the rest of the season." But they should not say before any

one game, "Our average is 98 therefore we will score 98 tonight." This is why I say *velocity is a useful long-term predictor but is not a useful short-term predictor*.

Velocity will bounce around from sprint to sprint. That's why I want teams to plan their sprints by looking at the product backlog, selecting the one most important thing they could do, breaking that product backlog item / user story into tasks and estimating the tasks, asking themselves if they can commit to delivering the product backlog item, and then repeating until they are full. No discussion of story points. No discussion of velocity. It's just about commitment and we decide how much we can commit to by breaking product backlog items into tasks and estimating each. This is called [capacity-driven sprint planning](#).

When a team finishes planning a sprint in this way it is indeed likely that the number of story points they have unknowingly committed to should be close to their long-term average but it will vary some. It will also be true that a team will commit to approximately the same number of hours from one sprint to the next. I use the term *capacity* to refer to this number of hours because *velocity* is reserved for referring to measuring the amount of work planned or completed as given in the units used to estimate the product backlog (which I recommend be done using story points).

[Download my PDF](#)

Posted: November 7, 2007

Tagged: product backlog, estimating, story points, audio, sprint backlog, release planning

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding

member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

What Are Agile Story Points?

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

Don't Equate Story Points to Hours

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • 39 Comments [Read](#)

Epics, Features and User Stories

I've been getting more and more emails lately from people confused about the difference between ...

Nov 08, 2022 • 93 Comments [Read](#)

For Better Agile Planning, Be Collaborative

The best plans are created by developers and stakeholders working together.

Jul 12, 2022 [Read](#)

Relationship between Definition of Done and Conditions of Satisfaction

I'd like to clarify the relationship between two important concepts: a team's Definition of Done ...

Jun 28, 2022 • 38 Comments [Read](#)

Four Reasons Agile Teams Estimate Product Backlog Item

Estimating product backlog items provides benefits beyond predicting when a project will be finished.

May 17, 2022 [Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

Don't Average During Planning Poker

by

Mike **Tagged:**

Cohn product backlog, estimating, user stories, product owner, teams,
29 backlog, planning poker
[Comments](#)

I like to use [Planning Poker](#) to estimate the user stories on an agile team's product backlog. In this approach individual estimators hold up cards showing their estimates. If estimators disagree they discuss why, ask questions of their product owner (who should be present), and repeat until they come to consensus.

Team members often ask me whether they really need to come to consensus or whether they can just take the mean of the individual estimates. The problem with averaging is that it is too easy--rather than have the fierce discussion that is one of the huge benefits of playing Planning Poker teams fall into a trap of playing one or two rounds and then just averaging.

An obvious dysfunction is that one estimator may play the 100 card not because he thinks it will take that long but because he thinks 20 is the right number and other estimators are thinking 8 and 13. For this reason and others, if a team truly feels compelled to average, they should take the median (middle value) rather than the mean (sum of estimates divided by number of estimates). A lot of dark corners are enlightened through the discussion; teams lose out on that when they average.

So while I want teams to come to agreement, I don't care how heartfelt the agreement is. If we agree on 13 some of us may really believe that's the right number. Others may think 8 is right but that 13 is "close enough." Still others may think we've discussed the item too long and even though it should be a 20 will give in and call it a 13 just to be done with it. So, rather than average if the team is an impasse I suggest going another round. If still stuck, someone should suggest a reasonable number and see if everyone can "support it" rather than "think it's the absolutely perfect number."

[Take the Assessment](#)

Posted: October 11, 2007

Tagged: product backlog, estimating, user stories, product owner, teams, backlog, planning poker

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at

hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

User Stories: How to Create Story Maps

Story maps help to create a shared understanding of the product, visualize user needs, and elicit ...

Mar 14, 2023 [Read](#)

How to Run a Successful User Story Writing Workshop

What you need to know to conduct a story-writing workshop with your team.

Feb 28, 2023 • 3 Comments [Read](#)

What Is Cross-Functional Collaboration in Agile?

Perhaps the most prevalent and persistent myth in agile is that a cross-functional team is one on ...

Feb 14, 2023 • 49 Comments [Read](#)

Be a Great Product Owner: Six Things Teams and Scrum Masters Need

Learn six ways effective product owners ensure their teams' success.

Jan 31, 2023 • 16 Comments [Read](#)

What Are Agile Story Points?

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

Don't Equate Story Points to Hours

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • 39 Comments [Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile **Agile & Scrum Training** **More...**

New to Agile and Scrum? Online Certifications About Us

Scrum Video Courses Contact Us

User Stories In-Person Certifications Our Students

Planning Poker On-Site Courses Blog

Agile Software Development Training Schedule Books by Mike Cohn

Development Compare Courses Agile Tools

Agile Project Management PDUs and SEUs

Agile Presentations

Mountain Goat on YouTube

Agile Mentors Podcast

Elements of Agile

Assessment

To Re-estimate or not; that is the question

by
Mike Cohn 112
Tagged: product backlog, estimating, user stories, story points
[Comments](#)

Should a team that is estimating in story points ever re-estimate? This is a question I'm commonly asked and would like to address here. Most people have a natural feeling that re-estimating is somehow wrong but they can't quite say why. I'll encourage those individuals to stick to their hunches, and hopefully I can provide some reasoning that supports your natural inclination that most re-estimating is wrong. Philosophers talk about two types of knowledge.

The first is *a priori* knowledge, which is knowledge before you experience something. Let's call this *knowledge-before-the-fact*. This is the type of knowledge we have when we estimate something. Before estimating development of the new search screen I think it's about 8 story points, because it seems to be

about the same total effort as some other 8 point story. The other type of knowledge is called *a posteriori* knowledge by the philosophers. This is knowledge *after the fact*. When we estimate it is important that we not mix *knowledge-before-the-fact* with *knowledge-after-the-fact*.

Suppose you are looking at a Scrum product backlog that has just been estimated with none of the work started. Each of those estimates was given before-the-fact (a priori). Now suppose you are looking at the same project a few months later. You've got a list of completed work, some of the items on that list still show their original, before-the-fact estimates but some have been re-estimated with after-the-fact estimates. The product backlog is similarly mixed: mostly the initial, before-the-fact estimates but some estimates that have been revised after-the-fact because of what was learned by developing previous user stories off the backlog. Having both before-the-fact and after-the-fact estimates on your product backlog and list of finished work can cause a lot of confusion for the project. When all estimates are given in before-the-fact numbers we can reason about them and compare them.

Suppose the team is estimating a new item and want to say its equivalent to 20 story points because it's similar to another item that has been estimated at 20 story points. That logic makes sense if the original item has not been re-estimated. If the old item was given an estimate of 10 before the fact and re-estimated to 20 after the fact then it is harder to know if the new item should get a 10 or a 20. With the re-estimation having occurred we're in the position of saying "Before I start this one I think it's a 20 because the other one felt like a 20 after I did it." That's weaker than "Before I do either of these they seem the same size."

So, does this mean you should never re-estimate? Absolutely not. There are times when you want to re-estimate. Generally re-estimating is useful when you completely blew it on the original estimate and can see that the mistake was a rare occurrence. (That is, if every estimate is systematically off by half I wouldn't re-estimate.)

Second, you should re-estimate when there has been a change in relative size. For example, the team has discovered that learning AJAX will be about half as hard as they thought. We'd want to fix that because the new knowledge tells us that our relative estimates are off-kilter for the AJAX-heavy stories.

Get Your Free Scrum Cheat Sheet

Sign up below to receive this FREE reference.

First Name

Email Address

[Privacy Policy](#)

Posted: September 2, 2007

Tagged: product backlog, estimating, user stories, story points

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the *Better User Stories* video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

[User Stories: How to Create Story Maps](#)

Story maps help to create a shared understanding of the product, visualize user needs, and elicit ...

Mar 14, 2023

[Read](#)

[How to Run a Successful User Story Writing Workshop](#)

What you need to know to conduct a story-writing workshop with your team.

Feb 28, 2023 • [3 Comments](#)

[Read](#)

[What Are Agile Story Points?](#)

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • [122 Comments](#)

[Read](#)

[Don't Equate Story Points to Hours](#)

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • [39 Comments](#)

[Read](#)

[Epics, Features and User Stories](#)

I've been getting more and more emails lately from people confused about the difference between ...

Nov 08, 2022 • [93 Comments](#)

[Read](#)

[Advantages of User Stories over Requirements and Use Cases](#)

At the surface, user stories appear to have much in common with use cases and traditional ...

Oct 05, 2022 • [41 Comments](#)

[Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

Sprint and release planning should be in different units

by

Mike Cohn

Tagged: product backlog, estimating, user stories, story points, sprint planning, sprint backlog, release planning

Comments

A common source of confusion on agile teams occurs when the sprint ("iteration") backlog and the product backlog are both estimated in hours. To avoid this confusion I strongly recommend estimating these backlogs in different units. In sprint planning the team should always talk of tasks and hours. Sprint planning covers the horizon of typically two to four weeks out. In release planning the team can choose between "ideal days" and "story points."

Regardless of which they choose, they still do sprint planning in hours. I prefer story points for the product

backlog items (typically "user stories" are on the product backlog for me). What this means is that I may have a user story ("As a vacation planner, I can see photos of hotels so that I can choose the right hotel for my vacation.") that is estimate in points--let's say 5 points. That hangs out on the product backlog (PB) until the product owner prioritizes it such that the team chooses to work on it in a sprint. Once selected it is broken down into tasks and hours:

- code the user interface, 6 hours
- code new stored procedure, 4 hours
- add photo maintenance page, 8 hours
- write automated tests, 5 hours

and so on.

So both are used but at different times and when viewing items at different horizons. Here's a key reason I prefer points: If I estimate the PB items in ideal days then it is too easy to mistakenly think that the PBIs and the items on the sprint backlog are estimated in the same unit. After all, the sprint backlog is estimated in ideal hours and the PB is estimated in ideal days. So, they're the same unit (times 8 for the PB), right? This is a *huge* fallacy. On average the teams I've coached spend 30 minutes breaking a product backlog item into tasks and estimating those tasks.

So, let's not call that estimate "hours". Let's call it "hours I thought a lot about." On the other hand, teams I coach spend 2-3 minutes on average estimating the PBIs. (These items don't need the detailed thought upfront; we just want a rough estimate so we can decide priorities and basic schedule.) So, let's call these "hours I pulled out of the air." When the PB and the SB are estimated in days and hours, it is too tempting to divide the number of days on the PB (times eight) by the number of hours finished per sprint and think that's an estimate of how long the rest of the project will take. However, that's bad math. It's literally dividing apples by oranges. It's "hours I pulled out of the air" divided by "hours I thought a lot about." The result will be meaningless. The problem goes away when teams go to two-level planning (release and sprint) and when they track velocity in story points.

Get Free User Stories Book Chapters

Please provide your name and email and we'll send you the sample chapters and we'll send a short weekly tip from Mike on how to succeed with agile.

[Get my chapters now!](#)

Posted: January 13, 2007

Tagged: product backlog, estimating, user stories, story points, sprint planning, sprint backlog, release planning

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the *Better User Stories* video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

[Why I Don't Emphasize Sprint Goals](#)

[User Stories: How to Create Great Ones](#)

[How to Run a Successful User Story Workshop](#)

[What Are Agile Story Points?](#)

[Don't Equate Story Points to Hours](#)

[Epics, Features and User Stories](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile

New to Agile and Scrum?

Scrum

User Stories

Planning Poker

Agile Software Development

Agile Project Management

Agile Presentations

Mountain Goat on YouTube

Agile Mentors Podcast

Elements of Agile

Assessment

Agile & Scrum Training

Online Certifications

Video Courses

In-Person Certifications

On-Site Courses

Training Schedule

Compare Courses

PDUs and SEUs

More...

About Us

Contact Us

Our Students

Blog

Books by Mike Cohn

Agile Tools

Writing Contracts for Agile Development

by [Mike Cohn](#) **Tagged:** estimating, user stories, teams, sprinting, sprint planning, customers, agile processes, articles
18 originally published in Gantthead on 2006-09-01
[Comments](#)

User stories have become an increasingly popular way for agile teams to express requirements. Teams like that working with user stories shifts the emphasis from writing about requirements to talking about them.

Project customers like that they are not expected to do the impossible and identify all needs upfront. Users of a product built with user stories like that the product is more likely to meet their needs than one built using a requirements technique more focused on writing than talking.

However, even though user stories cause teams and customers to talk more, there is often much still that needs to be written. Contract or outsourced development is one situation in which a team will need to augment conversations with documentation. This article will show one technique I've used for writing

contracts to cover contracted agile development.

A few years ago I worked on a system intended for use by broadcasters during the summer Olympics. Sportscasters, writers, and others would use the system to dredge up facts about competitors in the various events. A sample scenario was that a sportscaster who would be covering the 400-meter hurdles later in the day would use this system to research past performance and find interesting personal facts to interject into the commentary. The system was designed as a large web of information: a user could, for example, start with the event (400-meter hurdles) and drill down from country to athlete. Or starting with an athlete a user could navigate to all events that athlete would compete in and then to another athlete in one of those events, and so on. This application was outsourced to my team which developed it using Scrum, one of the most popular agile processes

Requirements were written as user stories using this template:

As a , I want [so that].

This template allows user stories to clearly identify the user who wants to accomplish something, what it is that she wants to accomplish, and optionally why. There were a handful of user types on this system but for this example we only need to know about two of them: viewers and content providers. A viewer was any user who was primarily looking at data in the system. This includes the sportscasters and researchers discussed earlier. Content providers were the writers who put all of the data into the system in advance of its use. Some of the stories identified for this system were:

1. As a viewer I can view details about a specific athlete.
2. As a viewer viewing an athlete, I can easily navigate to the events he's in and a list of other athletes on his team.
3. As a viewer I can bookmark athletes of interest.
4. As a viewer I can easily return to athletes I've bookmarked.
5. As a viewer I can highlight portions of an athlete's profile so that when I next view the profile I can see items of most interest to me.
6. As a content provider, I can have basic formatting control over how information is displayed.

It would have been challenging and risky for us to estimate work based solely on these user stories and then to commit to a fixed price contract. We needed more information, which we could get through conversations with our customer. But we also needed to document some key assumptions and decisions that would result from those conversations. What we did was identify a small number of high level acceptance criteria for each user story. Today I refer to these as conditions of satisfaction for the story. We then produced a requirements documents that included each user story along with its conditions of satisfaction. Here's how the first story above was augmented with its conditions of satisfaction:

1. As a viewer I can view details about a specific athlete.

Name (multiple, could be long, include pronunciation)

Nickname (include pronunciation)

Prior performance at Olympics

World and Olympic records held

Interesting anecdotes

Citizenship, where trained, coach (link)

6. As a content provider I can have basic formatting control over how information is displayed.

Italics

Bold

Bulleted and numbered lists

Not WYSIWYG but can click a button to see a preview

The conversations that led the team to this list were held before the contract was written and before coding began. Discussions of story #6 were particularly important because the team was able to rule out the WYSIWYG (What You See Is What You Get) editor and save the customer money with a preview button instead. They also determined that very simple formatting was all that was required. Font styles, sizes, and colors, for example, were not needed. This helped avoid a potential conflict that could have resulted in a money-losing low bid or an unnecessarily high bid for the project that the customer would have rejected.

Sometimes a user story that appears very straightforward hides some very risky assumptions. Discussing the conditions of satisfaction can help identify these assumptions so that the developer and customer can be more confident that they are in agreement and that the contract covers the work expected. An example of this can be seen in this user story:

5. As a user I can highlight portions of an athlete's profile so that when I next view the profile I can see items of most interest to me.

At first glance this story seemed clear. The developers were planning an interface that would allow a viewer to select a yellow highlighting marker tool and indicate as many regions as desired. However, our conversations with the customer pointed to more work than that. Like all of our discussions to flesh out details of a user story we asked the customer questions like:

How will we know we're done?

When we demo this story to you what would you like to see so that you know we've done what you expect?

This led to a discussion of requirements that the customer hadn't even thought about. Some of the highlights should work precisely as we'd planned, but now the customer also wanted "global highlights" made by one user but visible to all. This led to a discussion of security privileges (should all users see all highlights?) and other topics. In the end we settled on a yellow highlighter tool for private highlights and a blue tool for global highlights. The story and its conditions of satisfaction appeared in our contract as:

5. As a user I can highlight portions of an athlete's profile so that when I next view the profile I can see items of most interest to me.

Two types of highlighting—yellow (private) and blue (global)

Highlights made during one session are visible in later sessions

Naturally, collecting these conditions of satisfaction can be time-consuming. This is somewhat mitigated by the fact that we are not trying to drive every bit of uncertainty from the requirements as that would be impossible. Instead, we're trying to drive out sufficient uncertainty that each party to the contract can feel comfortable with the amount of risk being assumed. In my experience taking user stories down to the level of conditions of satisfaction is often useful on the first contract between a developer and customer. For subsequent projects—if the two have learned how to work together and have established a level of trust—contracts may only need to include the user stories.

Of course, there's more to writing a good contract for an agile project than just including conditions of

satisfaction for the user stories. A good contract will also align incentives between the parties. Additionally, the team must have a good approach to estimating that provides a high likelihood of a profitable project. Starting with user stories augmented with their conditions of satisfaction, however, is a good start.

[Take the Assessment](#)

Posted: August 31, 2006

Tagged: estimating, user stories, teams, sprinting, sprint planning, customers, agile processes, articles

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

[Why I Don't Emphasize Sprint Goals](#)

Sprint goals are considered a mandatory part of Scrum. Here's why I disagree.

Mar 21, 2023

[Read](#)

User Stories: How to Create Story Maps

Story maps help to create a shared understanding of the product, visualize user needs, and elicit ...

Mar 14, 2023

[Read](#)

How to Run a Successful User Story Writing Workshop

What you need to know to conduct a story-writing workshop with your team.

Feb 28, 2023 • [3 Comments](#)

[Read](#)

[What Is Cross-Functional Collaboration in Agile?](#)

Perhaps the most prevalent and persistent myth in agile is that a cross-functional team is one on ...

Feb 14, 2023 • [49 Comments](#)

[Read](#)

[What Happens When During a Sprint](#)

Succeeding with Scrum is easier when you know when and why to conduct each of the Scrum events ...

Jan 03, 2023 • [34 Comments](#)

[Read](#)

What Are Agile Story Points?

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • [122 Comments](#)

[Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

Estimating With Use Case Points

by Mike Cohn
1 articles
Comments originally published in Methods & Tools on 2005-10-15

Tagged:
estimating, user stories, teams, sprinting, story points, sprint planning, meetings, velocity, programming, management, metrics, project management, scrum master, agile software development,

If you've worked with use cases, you've probably felt there should be an easy way to estimate the overall size of a project from all the work that went into writing the use cases. There's clearly a relationship between use cases and code in that complicated use cases generally take longer to code than simple use cases. Fortunately, there is an approach for estimating and planning with use case points. Similar in concept to function points, use case points measure the size of an application. Once we know the approximate size of an application, we can derive an expected duration for the project if we also know (or can estimate) the team's rate of progress.

Use case points were first described by Gustav Karner, but his initial work on the subject is closely guarded by Rational Software. This article, therefore, primarily documents Karner's work as described by Schneider and Winters (1998) and

Ribu (2001).

Use Case Points

The number of use case points in a project is a function of the following:

the number and complexity of the use cases in the system

the number and complexity of the actors on the system

various non-functional requirements (such as portability, performance, maintainability) that are not written as use cases

the environment in which the project will be developed (such as the language, the team's motivation, and so on)

The basic formula for converting all of this into a single measure, use case points, is that we will "weigh" the complexity of the use cases and actors and then adjust their combined weight to reflect the influence of the nonfunctional and environmental factors.

Fundamental to the use of use case points is the need for all use cases to be written at approximately the same level. Alistair Cockburn (2001) identifies five levels for use cases: very high summary, summary, user goal, subfunction, and too low. Cockburn's very high summary and summary use cases are useful for setting the context within which lower-level use cases operate. However, they are written at too high of a level to be useful for estimating. Cockburn recommends that user goal-level use cases form the foundation of a well thought through collection of use cases. At a lower level, subfunction use cases are written to provide detail on an as-needed basis.

If a project team wishes to estimate with use case points, they should write their use cases at Cockburn's user goal level. Each use case (at all levels of Cockburn's hierarchy) has a goal. The goal of a user goal-level use case is a fundamental unit of business value. There are two tests for the whether a user goal use case is written at the proper level: First, the more often the user achieves the goal, the more value is delivered to the business;

Second, the use case is normally completed within a single session and after the goal is achieved, the user may go on to some other activity.

A sample user goal use case is shown in Figure 1. This use case is from a job posting and search site. It describes the situation in which a third-party recruiter has already posted a job opening on the site and now needs to submit payment for placing that ad.

Since this is not an article on use cases, I won't fully cover all the details of the use case shown in Figure 1; however, it is worth reviewing the meaning of the Main Success Scenario and Extensions sections. The Main Success Scenario is a description of the primary successful path through the use case. In this case, success is achieved after completing the five steps shown. The Extensions section defines alternative paths through the use case. Often, extensions are used for error handling; but extensions are also used to describe successful but secondary paths, such as in extension 3a of Figure 1. Each path through a use case is referred to as a scenario. So, just as the Main Success Scenario represents the sequence of steps one through five, an alternate scenario is represented by the sequence 1, 2, 2a, 2a1, 2, 3, 4, 5

Unadjusted Use Case Weight

If all of a project's use cases are written at approximately the level of detail shown in Figure 1, it's possible to calculate use case points from them. Unlike an expert opinionbased estimating approach where the team discusses items and estimates them, use case points are assigned by a formula. In Karner's original formula, each use case is assigned a number of points based on the number of transactions within the use case. A transaction (at least when working with user goal-level use cases) is equivalent to a step in the use case. Therefore we can determine the number of transactions by counting the steps in the use case. Karner originally proposed ignoring transactions in the extensions part of a use case. However, this was probably largely because extensions were not as commonly used in the use cases he worked with during the era when he first proposed use case points (1993). Extensions clearly represent a significant amount of work and need to be included in any reasonable estimating effort.

Counting the number of transactions in a use case with extensions requires a small amount of caution. That is, you cannot simply count the number of lines in the extension part of the template and add those to the lines in the main success scenario.

In Figure 1, each extension starts with a result of a transaction, rather than a new transaction itself. For example, extension 2a ("The card is not of a type accepted by the system") is the result of the transaction described by step 2 of the main success scenario ("System validates credit card"). So, item 2a in the extensions section of Figure 1 is not counted. The same, of course, is true for 2b, 2c, and 3a. The transaction count for the use case in Figure 1 is then ten. You may want to count 2b1 and 2b2 only once but that is more effort than is worthwhile, and they may be separate transactions sharing common text in the use case.

Table 1 shows the points assigned to each simple, average, and complex use case based on the number of transactions. Since the use case we're considering contains more than seven transactions it is considered complex.

Repeat this process for each use case in the project. The sum of the weights for each use case is known as the Unadjusted Use Case Weight, or UUCW. Table 2 shows how to calculate UUCW for a project with 40 simple use cases, 21 average, and 10 complex.

Unadjusted Actor Weight

The transactions (or steps) of a use case are one aspect of the complexity of a use case, the actors involved in a use case are another. An actor in a use case might be a person, another program, a piece of hardware, and so on. Some actors, such as a user working with a straightforward command-line interface, have very simple needs and increase the complexity of a use case only slightly. Other actors, such as a user working with a highly interactive graphical user interface, have a much more significant impact on the effort to develop a use case. To capture these differences, each actor in the system is classified as simple, average, or complex, and is assigned a weight in the same way the use cases were weighted.

In Karner's use case point system, a simple actor is another system that is interacted with through an API (Application Programming Interface). An average actor may be either a person interacting through a text-based user interface or another system interacting through a protocol such as TCP/IP, HTTP, or SOAP. A complex actor is a human interacting with the system through a graphical user interface. This is summarized, and the weight of each actor type is given, in Table 3.

Each actor in the proposed system is assessed as either simple, average, or complex and is weighted accordingly. The sum of all actor weights is known as Unadjusted Actor Weight (UAW). This is shown for a sample project in Table 4.

Unadjusted Use Case Points

At this point we have the two values that represent the size of the system to be built. Combining the Unadjusted Use Case Weight (UUCW) and the Unadjusted Actor Weight (UAW) gives the unadjusted size of the overall system. This is referred to as Unadjusted Use Case Points (UUCP) and is determined by this equation:

$$UUCP = UUCW + UAW$$

Using our example, UUCP is calculated as:

$$UUCP = 560 + 40 = 600$$

To this estimate of the size of the application, Karner's use case points approach next applies a pair of adjustments to reflect the technical and environmental complexity ! associated with the system being developed.

Adjusting For Technical Complexity

The total effort to develop a system is influenced by factors beyond the collection of use cases that describe the functionality of the intended system. A distributed system will take more effort to develop than a nondistributed system. Similarly, a system with difficult to meet performance objectives will take more effort than one with easily met performance objectives. The impact on use case points of the technical complexity of a project is captured by assessing the project on each of thirteen factors, as shown in Table 5. Many of these factors represent the impact of a project's nonfunctional requirements on the effort to complete the project. The project is assessed and rated from 0 (irrelevant) to 5 (very important) for each of these factors.

An example assessment of a project's technical factors is shown in Table 6. This project is a web-based system for making investment decisions and trading mutual funds. It is somewhat distributed and is given a three for that factor.

Users expect good performance but nothing above or beyond a normal web application so it is given a three for performance objectives. End users will expect to be efficient but there are no exceptional demands in this area. Processing is relatively straightforward but some areas deal with money and we'll need to be more carefully developing, leading to a two for complex processing. There is no need to pursue reusable code and the system will not be installed outside the developing company's walls so these areas are given zeroes. It is extremely important that the system be easy to use, so it is given a four in that area. There are no portability concerns beyond a mild desire to keep database vendor options open. The system is expected to grow and change dramatically if the company succeeds and so a five is given for the system being easy to change. The system needs to support concurrent use by tens of thousands of users and is given a five in that area as well. Because the system is handling money and mutual funds, security is a significant concern and is given a five. Some slightly restricted access will be given to third-party partners and that area is given a three. Finally, there are no unique training needs so that is assessed as a zero.

In Karner's formula, the weighted assessments for these twelve individual factors are next summed into what is called *TFactor*. The TFactor is then used to calculate the *Technical Complexity Factor*, TCF, as follows:

$$TCF = 0.6 + (0.01 \times 42) = 1.02$$

In our example, $TCF = 0.6 + (0.01 \times 42) = 1.02$

Adjusting For Environmental Complexity

Environmental factors also affect the size of a project. The motivation level of the team, their experience with the application, and other factors affect the calculation of use case points. Table 7 shows the eight environmental factors Karner's formulas consider for each project.

An example assessment of a project's environmental factors is shown in Table 8. The weighted assessments for these eight individual factors are summed into what is called the *EFactor*. The EFactor is then used to calculate the *Environment Factor*, EF, as follows:

$$EF = 1.4 + (-0.03 \times EFactor)$$

In our example, this leads to:

$$EF = 1.4 + (-0.03 \times 17.5) = 1.4 + (-0.51) = 0.89$$

Putting It All Together

To come up with our final Use Case Point (UCP) total, Karner's formula takes the Unadjusted Use Case Points (UUCP, the sum of the Unadjusted Use Case Weight and the Unadjusted Actor Weight) and adjusts it by the Technical Complexity Factor (TCF) and the Environmental Factor (EF). This is done with the following formula:

$$UCP = UUCW \times TCF \times EF$$

The values that were determined for these components in the example throughout this article are summarized in Table 9. Substituting values from Table 9 into the UCP formula, we get:

$$UCP = 600 \times 1.02 \times 0.89 = 545$$

Deriving Duration

First, notice that this section is titled "Deriving Duration." It is not called "Estimating Duration." An appropriate approach to planning a project is that we estimate size and derive duration. Use case points are an estimate of the size of a project. We cannot, however, go to a project sponsor who has asked how long a project will take and give the answer "545 use case points" and leave it at that. From that estimate of size we need to derive an appropriate duration for the project. Deriving duration is simple—all we need to know is the team's rate of progress through the use cases.

Karner originally proposed a ratio of 20 hours per use case point. This means that our example of 545 use case points translates into 10,900 hours of development work. Building on Karner's work, Kirsten Ribu (2001) reports that this effort can range from 15 to 30 hours per use case point. A different approach is proposed by Schneider and Winters (1998). They suggest counting the number of environmental factors in E1 through E6 that are above 3 and those in E7 and E8 that are below three. If the total is two or less, assume 20 hours per use case point. If the total is 3 or 4, assume 28 hours per use case. Any total larger than 4 indicates that there are too many environmental factors stacked against the project. The project should be put on hold until some environmental factors can be improved.

Rather than use an estimated number of hours per use case point from one of these sources, a better solution is to calculate your organization's own historical average from past projects. For example, if five recent projects included 2,000 use case points and represented 44,000 hours of work, you would know that your organization's average is 22 hours per use case point ($44,000 \div 800 = 22$). If you are going to estimate with use case points, it is definitely worth starting a project repository for this type of data.

To derive an estimated duration for a project, select a range of hours. For example, you may use Schneider and Winters' range of 20 to 28 hours per use case point. Based on your experience with writing use cases, estimating in use case points, and the domain of the application you might want to widen or narrow this range. Using the range of hours and the number of use case points, you can derive how long the project will probably take. For example, suppose we have the following information:

The project has 545 use case points

The team will average between 20 and 28 hours per use case point

Iterations will be two weeks long

A total of ten developers (programmers, testers, DBAs, designers, etc.) will work on this project

In this case, the complete project will take between 10,900 hours and 15,260 hours to complete ($545 \times 20 = 10,900$ and $545 \times 28 = 15,260$). We estimate that each developer will spend about 30 hours per week on project tasks. The rest of their time will be sucked up by corporate overhead—answering email, attending meetings, and so on. With ten developers, this means the team will make $10 \times 30 = 300$ hours per week or 600 hours of progress per iteration. Dividing 10,900 hours by 600 hours and rounding up indicates that the overall project might take 19 two-week iterations. Dividing 15,260 by 600 hours and rounding up indicates that it might take 26 two-week iterations. Our estimate is then that this project will take between 19 to 26 two-week iterations (38 to 52 weeks), and

Some Agile Adaptations

As originally conceived, a use case point approach to estimating is not particularly suited to teams using an agile software development process such as Scrum or Extreme Programming. This is one of the reasons I ultimately chose not to describe the approach in my book Agile Estimating and Planning (Cohn 2005). In particular, the need to create a complete use case model at the user goal level is incompatible with agile values because it encourages the early creation of a (supposedly complete) set of requirements. However, because many teams work with use cases and because many of them are moving in agile directions, it is worth suggesting how the approach can be applied in a semi-agile context.

Tracking Progress

One of the most useful techniques to come out of agile software development is the burndown chart (Schwaber and Beedle 2001). A typical release burndown chart shows the estimated amount of time remaining in a project as of the start of each iteration. The sample burndown chart in Figure 2 shows a project that had approximately 250 days of work at the start of the first iteration, about 200 by the start of the second iteration, and about 175 by the start of the third iteration. Things didn't go well during the third iteration, and by the start of the fourth iteration the team was back to an estimate of 200 days of work remaining. The cause of this increase is unknowable from the burndown chart. But this is usually the result of adding new requirements to the project or of discovering that some upcoming work had been incorrectly estimated.

Having become addicted to the use of burndown charts as a technique for monitoring the progress of a team, I am reluctant to let go of such a powerful communication and tracking tool. Fortunately, there is a way to use a burndown chart even for projects that estimate in use case points.

The best way to do this is to use only the Unadjusted Use Case Weight on the vertical axis, and to allow a team to burndown 5, 10, or 15 points for every simple, average, and complex use case they finish. (You'll recall these were the weightings shown in Table 1.) For the sample project discussed throughout this article, the intercept on the vertical axis would then be at 560, the Unadjusted Use Case Weight as calculated in Table 2. The burndown chart shown in Figure 3 starts at this point and shows the team's progress through the first two iterations.

Measuring Velocity

Agile teams like to measure their velocity, which is their rate of progress. With a use case point approach and with burndown charts drawn as described in the prior section, velocity is calculated as the sum of the weights of the use cases completed during an iteration.

Advantages and Disadvantages to Estimating with Use Case Points

As with most things, there are some advantages and disadvantages to the use case point approach. The final two sections of this article briefly outline the key issues.

Advantages

The first advantage to estimating with use case points is that the process can be automated. Some use case management tools will automatically count the number of use case points in a system. This can save the team a great deal of estimating time. Of course, there's the counter argument that an estimate is only as good as the effort put into it.

A second advantage is that it should be possible to establish an organizational average implementation time per use case point. This would be very useful in forecasting future schedules. Unfortunately, this depends heavily on the assumption that all use cases are consistently written with the same level of detail. This may be a very false assumption, especially when there are multiple use case authors.

A third advantage to use case points is that they are a very pure measure of size. Good estimation approaches allow us to separate estimating of size from deriving duration. Use case points qualify in this regard because the size of an application will be independent of the size, skill, and experience of the team that implements it.

Disadvantages

A fundamental problem with estimating with use case points is that the estimate cannot be arrived at until all of the use cases are written. Writing user goal use cases is a significant effort that can represent 10–20% of the overall effort of the project. This investment delays the point at which the team can create a release plan. More important, if all the use cases are all written up front, there is no learning based on working software during this period.

Use cases are large units of work to be used in planning a system. As we've seen in this article's example, 71 use cases can drive 38 to 52 weeks of work for a ten-person team. While use case points may work well for creating a rough, initial estimate of overall project size they are much less useful in driving the iteration-to-iteration work of a team.

A better approach will often be to break the use case into a set of user stories and estimate the user stories in either story points or ideal time (Cohn 2005).

A related issue is that the rules for determining what constitutes a transaction are imprecise. Counting the number of steps in a user goal user story is an approximation. However, since the detail reflected in a use case varies tremendously by the author of the use case, the approach is flawed.

An additional problem with use case points is that some of the Technical Factors (shown in Table 5) do not really have an impact across the overall project. Yet, because of the way they are multiplied with the weight of the use cases and actors the impact is such that they do. For example, technical factor T6 reflects the requirement for being able to easily install the system. Yes, in some ways, the larger a system is, the more time-consuming it will be to write its installation

procedure. However, I typically feel much more comfortable thinking of installation requirements on their own (for example, as separate user stories) rather than as a multiplier against the overall size of the system.

References

- Cockburn, Alistair. 2001. *Writing Effective Use Cases*. Addison-Wesley.
- Cohn, Mike. 2004. User Stories Applied for Agile Software Development. AddisonWesley.
- Cohn, Mike. 2005. *Agile Estimating and Planning*. Addison-Wesley.
- Ribu, Kirsten. 2001. *Estimating Object-Oriented Software Projects with Use Cases*. Master of Science Thesis, University of Oslo, Department of Informatics.
- Schwaber, Ken and Mike Beedle. 2001. *Agile Software Development with Scrum*. Prentice Hall.
- Schneider, Geri and Jason P. Winters. 1998. *Applying Use Cases: A Practical Guide*. Addison Wesley.

About The Author

Mike Cohn is the founder of Mountain Goat Software, a process and project management consultancy and training firm. He is the author of *User Stories Applied for Agile Software Development* and *Agile Estimating and Planning*, as well as books on Java and C++ programming. With more than 20 years of experience, Mike has previously been a technology executive in companies of various sizes, from startup to Fortune 40. A frequent magazine contributor and conference speaker, Mike is a Certified ScrumMaster Trainer and a founding member of the Agile Alliance, and serves on its board of directors. He can be reached at mike@mountaingoatsoftware.com.

[Download my PDF](#)

Posted: October 14, 2005

Tagged: estimating , user stories , teams , sprinting , story points , sprint planning , meetings , velocity , programming , management

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the *Better User Stories* video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

Why I Don't Emphasize Sprint Goals

Sprint goals are considered a mandatory part of Scrum. Here's why I disagree.

Mar 21, 2023

[Read](#)

User Stories: How to Create Story Maps

Story maps help to create a shared understanding of the product, visualize user needs, and elicit ...

Mar 14, 2023

[Read](#)

How to Run a Successful User Story Writing Workshop

What you need to know to conduct a story-writing workshop with your team.

Feb 28, 2023 • [3 Comments](#)

[Read](#)

What Is Cross-Functional Collaboration in Agile?

Perhaps the most prevalent and persistent myth in agile is that a cross-functional team is one on ...

Feb 14, 2023 • [49 Comments](#)

[Read](#)

Six Attributes of a Great ScrumMaster

Scrum Masters exhibit a shared set of traits. Discover 6 in-demand qualities, plus a bonus ...

Jan 17, 2023 • [32 Comments](#)

[Read](#)

What Happens When During a Sprint

Succeeding with Scrum is easier when you know when and why to conduct each of the Scrum events ...

Jan 03, 2023 • [34 Comments](#)

[Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile Assessment		

Learn About Agile

An Introduction to Agile and Scrum

[The Scrum Framework](#)

Scrum Roles: An Overview

- [Scrum Master Role and Responsibilities](#)
- [Product Owner Role and Responsibilities](#)
- [Scrum Team Role & Responsibilities](#)
- [The Chicken and the Pig](#)

Scrum Activities: An Overview

- [Sprint Planning Meeting](#)
- [Daily Scrum Meeting](#)
- [Sprint Review Meeting](#)
- [Sprint Retrospective](#)

Scrum Tools: An Overview

- [Scrum Product Backlog](#)
- [Sprint Backlog](#)
- [Scrum Task Board](#)
- [Release Burndown Chart](#)

Free Scrum Resources

- [Reusable Scrum Presentation](#)

User Stories

- [Planning Poker](#)
- [Agile Software Development](#)
- [Agile Project Management](#)

This video is part of our 19-part Scrum Foundations video series. [Click here](#) to watch the rest of the series for free.

The agile product backlog in [Scrum](#) is a prioritized features list, containing short descriptions of all functionality desired in the product. When applying Scrum, it's not necessary to start a project with a lengthy, upfront effort to document all requirements. Typically, a Scrum team and its product owner begin by writing down everything they can think of for agile backlog prioritization. This agile product backlog is almost always more than enough for a first sprint. The Scrum product backlog is then allowed to grow and change as more is learned about the product and its customers.

A typical Scrum backlog comprises the following different types of items:

1. Features
2. Bugs
3. Technical work
4. Knowledge acquisition

By far, the predominant way for a Scrum team to express features on the agile product backlog is in the form of [user stories](#), which are short, simple descriptions of the desired functionality told from perspective of the user. An example would be, "As a shopper, I can review the items in my shopping cart before checking out so that I can see what I've already selected."

Because there's really no difference between a bug and a new feature -- each describes something different that a user wants -- bugs are also put on the Scrum product backlog.

Technical work and knowledge acquisition activities also belong on the agile backlog. An example of technical work would be, "Upgrade all developers' workstations to Windows 7." An example of knowledge acquisition could be a Scrum backlog item about researching various JavaScript libraries and making a selection.

The product owner shows up at the sprint planning meeting with the prioritized agile product backlog and describes the top items to the team. The team then determines which items they can complete during the coming sprint. The team then moves items from the product backlog to the sprint backlog. In

doing so, they expand each Scrum product backlog item into one or more sprint backlog tasks so they can more effectively share work during the sprint.

Conceptually, the team starts at the top of the prioritized Scrum backlog and draws a line after the lowest of the high-priority items they feel they can complete. In practice, it's not unusual to see a team select, for example, the top five items and then two items from lower on the list that are associated with the initial five.

For more, check out this [product backlog example](#).

Learn About Agile

New to Agile and Scrum?

Scrum

User Stories

Planning Poker

Agile Software Development

Agile Project Management

Agile Presentations

Mountain Goat on YouTube

Agile Mentors Podcast

Elements of Agile

Assessment

Agile & Scrum Training

Online Certifications

Video Courses

In-Person Certifications

On-Site Courses

Training Schedule

Compare Courses

PDUs and SEUs

More...

About Us

Contact Us

Our Students

Blog

Books by Mike Cohn

Agile Tools

by

Mike **Tagged:**

Cohn product backlog, product owner, product management, product,
41 software, service, hardware

[Comments](#)

If you work on a Scrum team, you undoubtedly know you should have a product backlog and product owner. But what, exactly, is a product?

For some teams, this can be a rather fundamental question. After all, an organization cannot identify appropriate product owners, teams and roles without first knowing what its products are. And if there is to be one product backlog per product, we need to know what our products are before creating a product backlog for each.

In most cases, it would seem straightforward to identify an organization's products. For example, a watch manufacturer might think of each item they sell as a product. But even that could be oversimplifying things. And, some organizations have a much harder time identifying products.

What Are an Airline's Products?

Take for example, an airline. A few years back, I was doing some work with an airline when the question of what is a product came up. Some people in the company argued that an airline does only one thing: move people from one location to another. They argued, therefore, that despite the company having over 40,000 employees, there should be only one product.

Others argued that there were many products within the airline. For example, they had a passenger-facing website that could be used to do things like make reservations, check in for a flight or check a flight's status. They also had a system for monitoring and scheduling aircraft maintenance. And another that allowed crew members to select the flights they wished to work based on seniority.

Were these also products?

A Definition of Product and Examples

I define a product as something (physical or not) that is created through a process and that provides benefits to a market.

From that, a chair would be a product. Microsoft Office would be a product. Agile consulting services would be a product. A painting would be a product. A product can be a something physical (the chair). It can be a digital product (Microsoft Office, an ebook or streaming videos). It also can be a service (consulting on how to adopt agile).

A product can even just be an idea (a patentable algorithm or the secret to getting more right swipes on Tinder).

Each of these is created through a process or, more generally, one or more activities. Someone milled and assembled the chair. Microsoft Office was designed, coded and tested. The process that creates a product does not need to be formal or defined. The creators may not even be aware of the process. But some form of activity goes into creating every product.

Products Can Be Defined Recursively

A product can exist within another product. A pen, for example, may have replaceable ink cartridges. The pen is a product. But so are the ink cartridges within the pen.

There are even subproducts within a chair. The company manufacturing and selling the chair may have purchased fully milled legs from another company. Chair legs would then be a product.

Products can be defined recursively. A lumber company harvested trees to make wood—a product in its own right. Next, a milling company used that wood to make chair legs. Another company then took those pre-milled chair legs, and used them to assemble chairs of their own design. So products can exist inside other products.

Products Provide Benefits to a Market

When we identify subproducts within a larger product, we need to be careful that each subproduct provides benefits to a market. The definition above does not say that something must be purchased for it to be a product. But, to be considered a product, the item must satisfy a need or desire.

This is true of pre-milled legs for a chair and replacement ink cartridges for a pen. When defining products--and, therefore, product owners and product backlog in Scrum--it will be important to define each product

such that it provides benefits to a market.

Applying this Definition to the Airline Example

So if products are created through some process and benefit some market, are software components products?

To help answer that question, consider the airline company example from earlier. How would knowing that a product is something that is created through a process and that provides benefits to a market help an airline company identify its products?

First of all, it is easy to see that transporting people from one place to another is a product. That activity provides value to a market of people who gladly pay for the ability to fly to a location.

But what about the system for monitoring and scheduling aircraft maintenance? I claim that, too, is a product.

It is created through a process and is also something that provides benefits to a market. What market?

Well, we could go all the way and say that passengers benefit from well-maintained and safe aircraft. But, even closer to the product's development, we can say that the airline employees benefit from having maintenance monitoring and scheduling computerized rather than needing to do that manually with paper.

The same could be said of the airline's website, which enables passengers to make flight reservations. There is a market for that.

So, yes our airline has one big product—and it also has many subproducts. Anything within that company that can be thought of as delivering value to a market is a product.

Applying this Definition to Software Components

With that in mind, consider a software component. If one team develops software used by other teams, can that be thought of as a product? Consider the example of a team building a calendar widget. It provides value to a market: the other teams that will use the widget. I would say, then, that a component built for multiple teams is a product.

I would draw the line, though, with a calendar widget (or any component) that is *used by only one team*. Yes, technically a market can exist with only one customer. A painting, for example, is sold to one person.

But, when talking about product development (as with agile), it can be dangerous to think of something as a product if it is used by only one person or group. It could lead, for example, to thinking of code as a product that is delivered to a *market* of testers. That is not only a step backward into sequential (or phased) development, it's also a form of suboptimization.

Avoiding Suboptimizing Products

According to San Jose State University Economics professor Thayer Watkins, suboptimization "refers to the

practice of focusing on one component of a total and making changes intended to improve that one component ... ignoring the effects on the other components."

While organizations do want to define all of their products in order to best manage the work, they do not want to narrow their focus to the degree that they fail to see the whole because they are fixated on the individual parts. As such, organizations want to define each product as broadly as possible.

That being said, as we saw earlier with the airline company, there is such a thing as too broad when it comes to identifying products. When a product is so large that it serves multiple markets, I often prefer to think of it as multiple products each serving a unique market.

For example, it would be quite easy to think of Microsoft Office as a single product. Yet, Office is a suite of products, each providing different features and serving slightly different (but overlapping) markets.

Because of that, I'd want to think of Word, Excel, PowerPoint and so on each as its own product. Each would have its own product owner and product backlog.

With a product as large as Office, it is quite likely there also would be products based on shared functionality across things like Word, Excel and PowerPoint. Thinking of the spell checker, for example, as a product could make sense. The spell checker provides benefits to a market (the other teams who do not need to write their own spell checker from scratch).

I would not, however, define Excel's Function widget (sum, average, count, etc.) as a product. It is unique to Excel and as such, serves only one customer. To give it a product backlog and product owner outside the context of Excel would be too risky.

In addition to avoiding one-customer products, another way to reduce suboptimal thinking when working with multiple products is to appoint a chief product owner. The chief product owner is a strategic role, responsible for establishing vision across all subproducts. On the Office product, for example, the chief product owner would be looking at how each individual product affects the suite as a whole.

What Do You Think?

Correctly identifying the products in your organization can help you avoid problems related to team structure, product backlog management and having people in the wrong roles. How do you define "product" in your organization? Are there guidelines you can extrapolate to create a general definition of product? Please share your thoughts in the comments below.

[Download my PDF](#)

Posted: September 6, 2016

Tagged: product backlog, product owner, product management, product, software, service, hardware

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

Be a Great Product Owner: Six Things Teams and Scrum Masters Need

Learn six ways effective product owners ensure their teams' success.

Jan 31, 2023 • 16 Comments [Read](#)

What Are Agile Story Points?

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

Epics, Features and User Stories

I've been getting more and more emails lately from people confused about the difference between ...

Nov 08, 2022 • 93 Comments [Read](#)

7 Ways to Get and Improve Fast Feedback

Here's how to get the rapid feedback you need to ensure you build the right product.

Jul 26, 2022 [Read](#)

Relationship between Definition of Done and Conditions of Satisfaction

I'd like to clarify the relationship between two important concepts: a team's Definition of Done ...

Jun 28, 2022 • 38 Comments [Read](#)

Four Reasons Agile Teams Estimate Product Backlog Item

Estimating product backlog items provides benefits beyond predicting when a project will be finished.

May 17, 2022 [Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile

New to Agile and Scrum?

Scrum

User Stories

Planning Poker

Agile Software Development

Agile Project Management

Agile Presentations

Mountain Goat on YouTube

Agile Mentors Podcast

Elements of Agile

Assessment

Agile & Scrum Training

Online Certifications

Video Courses

In-Person Certifications

On-Site Courses

Training Schedule

Compare Courses

PDUs and SEUs

More...

About Us

Contact Us

Our Students

Blog

Books by Mike Cohn

Agile Tools

by

Mike Cohn

Tagged:

product backlog, product owner, meetings, refinement, grooming

61 Comments

Product backlog refinement—sometimes called product backlog grooming in reference to keeping the backlog clean and orderly—is a meeting that is held near the end of one sprint to ensure the backlog is ready for the next sprint.

During a product backlog refinement meeting, the team and product owner discuss the top items on the product backlog. The team is given a chance to ask the questions that would normally arise during sprint planning:

What should we do if the user enters invalid data here?

Are all users allowed to access this part of the system?

What happens if...?

By asking these questions earlier, the product owner is given a chance to arrive at answers to any questions he or she may not be prepared to answer immediately.

If those questions were asked for the first time in sprint planning, and too many could not be answered, it might be necessary to put a high-priority product backlog item aside and not work on it during the sprint.

These questions do not need to be fully resolved in a backlog refinement meeting. Rather, the product owner

needs only to address them just enough so that the team feels confident that the story can be adequately discussed during the coming planning meeting.

Backlog refinement in that sense is really a checkpoint rather than an effort to fully resolve issues.

I like to hold the product backlog refinement meetings three days before the end of the current sprint. This gives the product owner sufficient time to act on any issues that are identified. Some teams find that doing shorter meetings every week rather than once per sprint are more suited to their cadence, and that is, of course, fine.

Unlike other Scrum meetings, I do not think the product backlog refinement meeting requires the participation of the whole team.

If you think about a whole team meeting three days before the end of the sprint, there is likely to be someone who will be frantically busy—someone who, if forced to attend one more meeting, may have to work late to finish the work of the sprint.

I'd prefer to conduct the meeting without such team members. As long as the same team members don't miss the meeting each sprint, I think it's fine to conduct backlog refinement meetings with about half the team plus the product owner and ScrumMaster.

[Download my PDF](#)

Posted: May 26, 2015

Tagged: product backlog, product owner, meetings, refinement, grooming

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and

techniques to build extremely high-performance teams. He is the author of User Stories Applied for Agile Software Development, Agile Estimating and Planning, and Succeeding with Agile as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

[Be a Great Product Owner: Six Things Teams and Scrum Masters Need](#)

Learn six ways effective product owners ensure their teams' success.

Jan 31, 2023 • 16 Comments

[Read](#)

[What Happens When During a Sprint](#)

Succeeding with Scrum is easier when you know when and why to conduct each of the Scrum events ...

Jan 03, 2023 • [34 Comments](#)

[Read](#)

[What Are Agile Story Points?](#)

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • [122 Comments](#)

[Read](#)

[Epics, Features and User Stories](#)

I've been getting more and more emails lately from people confused about the difference between ...

Nov 08, 2022 • [93 Comments](#)

[Read](#)

[Agile Mentors Podcast: Recap of Opening Series on Scrum](#)

Recapping our agile podcast's initial launch series of topics

Sep 27, 2022

[Read](#)

[Relationship between Definition of Done and Conditions of Satisfaction](#)

I'd like to clarify the relationship between two important concepts: a team's Definition of Done ...

Jun 28, 2022 • [38 Comments](#)

[Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Product Backlog Refinement

Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

Writing the Product Backlog Just in Time and Just Enough

by **Tagged:**
Mike product backlog, user stories, teams, sprinting, backlog, testing,
Cohn agile projects, scrum alliance, articles
10 originally published in Scrum Alliance on 2008-02-11
[Comments](#)

Should you write part of your product backlog and user stories before your first sprint? Some people want to take the stance that no work should be done in advance of the sprint. That is clearly untenable. To see why, let's take that view to its extreme: If we did nothing in advance to understand what we're building, we'd show up at the planning meeting and say, "Hey, what should we build this sprint? We were working on an eCommerce site yesterday, but I think maybe we should switch to writing a word processor..." The team would literally have nothing written down—no [user stories](#) and no [product backlog](#).

This approach leads to what I call “billiard ball sprints.” Without any planning, the team starts a new sprint on the day after the prior one ends, but spends the first two to five days “getting ready” to start the sprint. The start date of the second sprint is knocked forward by the end of the first—like one billiard ball knocking into another.

Having dispensed with the possibility that no work should be done in advance we are left to consider the question of how much work should be done in advance. Many say that as little as possible should be done: the team should do no more than write a few words of a user story on an index card to represent each product backlog item. While it’s true that this may be sufficient in many cases, it would clearly not be adequate in all situations.

My view is that each product backlog item (usually reflected as a user story by teams I train or coach) should be captured just in time and in just-enough detail for the team to go from product backlog item to working, tested feature within a sprint. To see why just-in-time and just-enough are appropriate targets, suppose we decide to write a checkbook program to compete with Intuit’s Quicken product. We create an initial product backlog that includes these two user stories:

As a user I can see the current version, company website, and copyright in a “help about” dialog so that I know useful information about the product.

As a user I can enter a check in my register so that I can track who I pay.

That first story can go from that short description to implemented code quite easily within a sprint. The analyst will have time to figure out exactly what the text should say; the user experience designer (UED) will have time to mock up two or three screens, show them to some users, get feedback, and create the best possible “help about” screen. And the programmers will have time to code it and the testers to test it. In short, it meets the just-enough-detail criteria.

The second user story, “As a user I can enter a check in my register,” is too big to be done in a single sprint. It encompasses the entire main user interface (UI) metaphor—will our system look like a paper check register? How many rows in the register? How will users choose things like check numbers or electronic funds transfer (EFT) transactions, and so on. There is no way the UEDs can figure all that out in a two-week sprint. This user story is not explained in just-enough detail. This is where our second attribute comes in: detail should be added to this user story (product backlog item) just in time. If the team will not work on this user story for six months there is no need to expand on what is already written. On the other hand, if the team hopes to start it in a few sprints, it is likely time for the UEDs to figure out the answers to the high-level questions listed above.

To do this, they’ll need to allow time to “figure out the answers” in the upcoming sprints. Let’s say that during the first sprint the team chooses to work on the “help about” story and likely a few others (not shown in our sample product backlog above). They also agree to spend some time figuring out what the main UI metaphor for the system will be. During this time the UED may design a couple of screens and show them to a few dozen or hundred users, get feedback, and do it again. This may happen a few times. When they finish they may have turned that one user story (as a user I can enter a check in my register) into a dozen smaller stories, perhaps specifying what fields are needed and other details such as:

As a user, when I enter a new check it defaults to the last check number + 1

As a user, I can enter a memo for each check, etc.

As a user, I can double-click on a check in a list and see the current item look like a paper check.

In some cases splitting a large story into smaller ones is sufficient to show the new items at the right (just-

enough) level of detail. When it's not, we can augment the user story with additional information. I like to write user stories on index cards. That was sufficient for the "help about" story. It's not for the initial large user story encompassing our system's main UI metaphor. If splitting a story does not add just-enough detail, the UED and others involved may want to staple a UI design spec to the index card. (If you are using a software system for managing your product backlog, your virtual staple may be a hyperlink to a wiki page.)

The UI design spec that is stapled to the user story card will not yet be perfect; rather it will be close enough that remaining details can be figured out during the sprint. For example, the UED may not yet have decided if each check should take two rows or three on the screen and wants to do a bit of user testing early in the sprint while the team codes that story. He'll get them an answer early enough that they can do it either way during the sprint.

The goal of the UED (and anyone else involved at the time) is to add detail to the story in a just-in-time manner. There is usually no value to splitting a large user story up now or stapling a document to it if we won't work on that product backlog item for quite awhile. While working to add detail in a just-in-time manner, those doing so should strive to add just-enough detail that no individual item will take more than one sprint to complete. Just in time and just enough become the targets for establishing a smooth flow from product backlog to sprint backlog.

[Watch Now](#)

Posted: February 10, 2008

Tagged: product backlog, user stories, teams, sprinting, backlog, testing, agile projects, scrum alliance,

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

User Stories: How to Create Story Maps

Story maps help to create a shared understanding of the product, visualize user needs, and elicit ...



Mar 14, 2023

How to Run a Successful User Story Writing Workshop

What you need to know to conduct a story-writing workshop with your team.

Feb 28, 2023 • 3 Comments [Read](#)

What Is Cross-Functional Collaboration in Agile?

Perhaps the most prevalent and persistent myth in agile is that a cross-functional team is one on ...

Feb 14, 2023 • 49 Comments [Read](#)

What Happens When During a Sprint

Succeeding with Scrum is easier when you know when and why to conduct each of the Scrum events ...

Jan 03, 2023 • 34 Comments [Read](#)

What Are Agile Story Points?

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

Don't Equate Story Points to Hours

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • 39 Comments [Re](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile Assessment		

Presentation by:

Mike Cohn

Tagged:

product backlog, sprinting, backlog, sprint planning, prioritizing, presentations

The biggest risk to most projects is building the wrong product. Regardless of how fast your agile team becomes, how brilliant your technical solutions are, or how many automated tests run continuously, you need to ensure that you are building the right thing.

In these presentations, Certified Scrum Trainer and agile expert Mike Cohn, examines both non-financial and financial ways of prioritizing product backlog items and choosing among competing project ideas. Learn about relative weighting, theme screening, theme scoring and Kano analysis. Take away practical knowledge about how to apply these straightforward yet powerful techniques to your product backlog.

[View the Presentation](#)[Download a PDF](#)**You may also be interested in:**

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

Why There Should Not Be a “Release Backlog”

by Mike Cohn
53 Comments Tagged:
product backlog, user stories, product owner, story points, velocity, project management, sprint backlog, iterations, iterating, release planning

I haven't heard the term "release backlog" in many months, but it's come up in three conversations over the past week. So, I want to share my thoughts on whether a team using an [Agile project management](#) approach should have a backlog known as a release in addition to the conventional Product and Sprint (or Iteration) Backlogs.

First, let's clarify what people mean when they refer to a "release backlog." A release backlog is a subset of the product backlog that is planned to be delivered in the coming release, typically a three- to six-month horizon. It would presumably contain the same type of items as on a product backlog. My preference for this, of course, would be user stories. So, at the start of some release planning horizon, a product owner with help

from the team and other stakeholders would examine the product backlog, select some amount of high priority work and move those items to the release backlog. Later, in sprint (iteration) planning, the team would examine the release product and select some amount of work to do. (Notice how this already goes against existing agile literature that says the team selects top priority items for the sprint *from the product backlog?*)

I am opposed to the use of a release backlog for a couple of reasons:

First, in all projects that are not being done as fixed-scope contracts (and, arguably, even then) we don't know exactly what user stories or features will be delivered in a release. To say that there is a release backlog may not imply a guarantee that all features on it are delivered. However, it does create additional work as items are moved off back and forth between the product and release backlogs. This will happen as the team's velocity changes (even by small amounts) from sprint to sprint. If the release backlog is to show what a team will deliver in a release, it should contain an amount of work equal to the number of story points (or ideal days) times the number of remaining sprints. If you have an average velocity of 20 and 6 sprints left, then this type of backlog should contain 120 points worth of work. Suppose the team finishes 24 points of work in a sprint. The backlog is now down to 96 points (120-24) but should contain 100 (5 sprints left x an average velocity of 20). Things get even more difficult if that good velocity of 24 increases the team's average velocity to 21. In that case the backlog should contain $5 \times 21 = 105$ points of work; we need to move 9 points of work from the product backlog to the sprint backlog. If there's a release backlog, this moving of items back and forth from product to release backlog will happen every sprint.

Second, introducing a release backlog creates additional confusion. We've already overloaded the word "backlog" with "product backlog" and "sprint backlog." Why confuse things further with a third type of backlog unless there is an immensely compelling reason to do so?

Third, introducing the concept of a Release Backlog actually makes it harder for product owners to do one of the most important things they should be doing--looking at the features or user stories that just miss the cut of being in a release. When I look at a product backlog and count down the number of story points that I anticipate will be in the release, the most interesting thing to me is not necessarily the set of user stories that will make it into the release. I'm often just as interested in the next few user stories below that cut-off point. That is, I want to know what user stories won't quite make it into the release. After all, we say that one of the benefits of agile (on some, not all, projects) is the ability to decide later if we should release a few sprints early (if a lot has been done), on the planned date, or perhaps a sprint or two late if we want to add more functionality before a release. This is made more difficult with a Release Backlog because a product owner now needs to examine both Product and Release Backlogs to make these decisions.

So, what do I propose instead?

I advocate that all teams track their velocities. This leads to a graph like this:

Velocity as graphed over the last nine sprints.

This graph shows a team calculating three average velocities:

1. Long term average, defined as the mean of the last eight sprints
2. Worst case average, defined as the mean of the worst three chosen among the last eight sprints
3. Best case average, defined as the mean of the best three chosen among the last eight sprints

You can choose your own ways of calculating long-term, best-case, and worst-case average velocities.

These are just the ones I like. We use these average velocities as shown in this figure:

In this figure, the product backlog is shown on the left as a stack of index cards. We use the three average velocities multiplied by the number of remaining sprints to draw arrows pointing into the product backlog. The range created by these arrows indicates the likely amount of work finished by the planned date. Our release plan is the work defined by the arrows. Rather than a Release Backlog as a new work product or artifact, my equivalent is the product backlog and these arrows pointing into it.

As you can imagine, predicting velocity as a range is much more likely to be accurate than using a single point estimate ("Our velocity is 31.") as has been implied by everyone I've heard talk about a Release Backlog. Additionally, looking at figures like these should make it apparent that the amount of work expected to be delivered in a release will change from sprint to sprint. Pointing arrows into a product backlog is much easier and more helpful than creating a new work product, the Release Backlog.

Get Your Free Scrum Cheat Sheet

Sign up below to receive this FREE reference.

First Name

Email Address

[Privacy Policy](#)

Posted: February 7, 2009

Tagged: product backlog, user stories, product owner, story points, velocity, project management, sprint backlog, iterations, iterating, release planning

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and*

Planning, and Succeeding with Agile as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

User Stories: How to Create Story Maps

Story maps help to create a shared understanding of the product, visualize user needs, and elicit ...

 Mar 14, 2023 [Read](#)

How to Run a Successful User Story Writing Workshop

What you need to know to conduct a story-writing workshop with your team.

Feb 28, 2023 • [3 Comments](#) [Read](#)

Be a Great Product Owner: Six Things Teams and Scrum Masters Need

Learn six ways effective product owners ensure their teams' success.

Jan 31, 2023 • [16 Comments](#) [Read](#)

What Are Agile Story Points?

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • [122 Comments](#) [Read](#)

Don't Equate Story Points to Hours

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • [39 Comments](#) [Read](#)

Epics, Features and User Stories

I've been getting more and more emails lately from people confused about the difference between ...

Nov 08, 2022 • [93 Comments](#) [Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile Assessment		

by
Mike Cohn **Tagged:**
38 product backlog, user stories, teams
[Comments](#)

I'd like to clarify the relationship between two important concepts: a team's *Definition of Done and the Conditions of Satisfaction (or Acceptance Criteria)* for a [user story](#). Let's start by reviewing each of these concepts.

What Is the Definition of Done?

The definition of done is an agreed-upon set of things that must be true before *any* product backlog item is considered complete. Let me say that a bit differently: *every* product backlog item for a particular product must satisfy the definition of done criteria to be considered potentially shippable.

What Are Some Definition of Done Examples?

Every team's definition of done will be slightly different, but a good starting point might be:

- the code is well-written. That is, the team does not feel they need to immediately refactor or rewrite it.
- the code is checked in.
- the code comes with [automated tests at all](#)

appropriate levels.

the code has been either pair programmed or has been code inspected.

the feature the code implements has been documented as needed in any end-user documentation.

Who Creates the Definition of Done?

The developers (aka the development team) create the definition of done through a shared understanding of what it means to create a high-quality, shippable product in their context. The definition of done applies to all product increments the team creates and complies with organizational standards of quality. The definition of done may also contain additional elements that are unique to that product.

What Are Conditions of Satisfaction?

In contrast, conditions of satisfaction are specific to a given product backlog item and define what must be true for that *particular* product backlog item to be considered done.

Acceptance Criteria vs Conditions of Satisfaction

Many people call conditions of satisfaction, acceptance criteria. And that's perfectly OK. I prefer to ask for conditions of satisfaction vs acceptance criteria for one reason: it makes more sense to product owners. And product owners are the people primarily responsible for creating conditions of satisfaction.

Product owners (and some programmers) consider writing acceptance criteria to be something specific that testers do. When I ask them to write the acceptance criteria for a user story, many product owners seem confused, saying they don't know how to write tests for code.

Product owners have a much easier time answering the question if I avoid the term acceptance criteria. So instead I ask them, what needs to be true in order for you to consider this particular story done? And we capture those as conditions of satisfaction.

What Are Some Examples of Conditions of Satisfaction?

To understand the difference between conditions of satisfaction and definition of done, it might be helpful to look at some examples. Consider a user story such as, "As a user, I am required to login before using the site." That user story might include these conditions of satisfaction:

user is logged in only when proper credentials are provided
a "remember me" option is available

user can request a password reminder
user is locked out after three failed attempts

For this example user story to be done, all of these conditions of satisfaction must be true **and** the team's definition of done must also be true.

Acceptance Criteria vs Definition of Done

Acceptance criteria and conditions of satisfaction (CoS) are two terms that mean almost the same thing. When it comes to product owners writing CoS for user stories, it's perfectly fine to call those conditions acceptance criteria. So acceptance criteria have the same relationship to definition of done as conditions of satisfaction do.

Think of the definition of done as a special set of acceptance criteria (aka conditions of satisfaction) that are added to every user story (product backlog item). For the user story above to be done, two things must be true. 1) All of the acceptance criteria (conditions of satisfaction) must be fulfilled, and 2) All of the items that make up the definition of done must be complete.

Because I like to write user stories on the front sides of index cards and the conditions of satisfaction (acceptance criteria) on the back sides, I tend to think of the definition of done as something I have written on a custom-made rubber stamp. I could then stamp each user story card with those items in addition to the hand-written acceptance criteria for this specific story.

What Do You Think?

How do you view the difference between the acceptance criteria for a product backlog item and a team's definition of done. Please share your thoughts in the Comments section below.

Get Your Free Scrum Cheat Sheet

Sign up below to receive this FREE reference.

First Name

Email Address

[Privacy Policy](#)

Posted: June 28, 2022

Tagged: product backlog, user stories, teams

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of User Stories Applied for Agile Software Development, Agile Estimating and Planning, and Succeeding with Agile as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

User Stories: How to Create Story Maps

Story maps help to create a shared understanding of the product, visualize user needs, and elicit ...

Mar 14, 2023

[Read](#)

How to Run a Successful User Story Writing Workshop

What you need to know to conduct a story-writing workshop with your team.

Feb 28, 2023 • [3 Comments](#)

[Read](#)

What Is Cross-Functional Collaboration in Agile?

Perhaps the most prevalent and persistent myth in agile is that a cross-functional team is one on ...

Feb 14, 2023 • [49 Comments](#)

[Read](#)

What Are Agile Story Points?

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • [122 Comments](#)

[Read](#)

Don't Equate Story Points to Hours

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • [39 Comments](#)

[Read](#)

Epics, Features and User Stories

I've been getting more and more emails lately from people confused about the difference between ...

Nov 08, 2022 • [93 Comments](#)

[Read](#)

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile Assessment		

by
Mike **Tagged:**
Cohn product backlog
Comments

I've never been a fan of MoSCoW prioritization. It involves sorting product backlog items (or requirements on a non-agile project) into those that a product

- Must Have
- Should Have
- Could Have
- Will Not Have

The first letters of those (MSCW) form the basis of the acronym. The premise is that must-have items are true requirements—include these or don't bother building the product.

The product could be released with its should-have items but if too many are absent, the product will probably not be very satisfying.

And could-have items are ones our customers and users want, but they acknowledge they're of lower value than the should-haves and must-haves.

It's possible to think of the must-haves as analogous to a minimum viable product. The should-haves and could-haves then move a product from viable to delightful.

What I Dislike about MoSCoW

There are a couple of things I dislike about the MoSCoW technique. First is that I can't recall a project ever including any could-have features. A feature on the could-have list might as well be put on the Will-Not-Have list.

This is why I've written previously about preferring a simple approach of Now and Not Now. Are we building a particular feature Now? Or Not Now?

Needs, Wants, and Wishes

A few months ago, I came across a different technique that is similar to MoSCoW. And I liked it.

Once a year, I meet with a financial planner who reviews my investment accounts and tells me if I'm on track toward things like retirement. (I'm not retiring any time soon—I love what I do far too much for that.)

Since my previous meeting with the planner, she's begun using new software. And to prepare for our meeting she asked me to log into it and fill out a few pages of information.

One of those pages asked me to enter my needs, wants, and desires. At first I thought "needs, wants, and desires" was an old song from either [Dusty Springfield](#) or [John Denver](#).

I clicked on a little question mark icon on the page to get some clarification. I read that Needs are things I will absolutely plan for in my future. Obviously that includes food, shelter, medical care, and such.

I also included in my retirement needs that my wife and I would like to visit our daughters, who may not live near us as they pursue their lives. Sure, maybe that's a want instead of a need. But since this was a retirement planning effort, I was saying that I'd make the choice to work a little longer if that's what it takes to be able to visit my daughters.

The software informed me that my wants were things I really, really wanted to plan on. But I'd be willing to give them up if necessary in order to make the plan work. I enjoy eating at restaurants so I wanted a retirement budget that supported that. But it's something I want, not something I need.

Finally, the software wanted to know my wishes. There really wasn't much definition other than they're a notch below my wants. And I didn't really have anything to put in that category—probably because of my view that Could-Haves never get delivered anyway. But, I was told, this is where I'd add things like taking a month-long cruise around Greece.

Finding It Helpful

Despite my dislike of Must-Have, Should-Have, and Could-Have, I found Needs, Wants, and Wishes quite useful.

I think it's because of the clarity. I find must-have, should-have, and could-have to be unclear. The product must have these features or what? It must have these features according to whom?

I find the could-have list particularly troublesome. Just about everything I've ever wanted could be added to a could-have list. Building a new e-commerce website? It could have a feature to cook me breakfast every morning.

Thinking of the Needs, Wants, and Wishes of your users seems to get around these problems.

Since encountering the idea in the financial planning software, I've had the opportunity to try these categories out with three companies. In each case it went well, and I found explaining wants vs. wishes easier than explaining could-haves.

What Do You Think?

I *need* to decide if this is a technique that warrants further use. I *want* to hear your thoughts on the topic. I *wish* you'd leave a comment below. Thanks.

[Download my PDF](#)

Posted: October 12, 2021

Tagged: product backlog

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and

can be reached at hello@mountaingoatsoftware.com.
If you want to succeed with agile, you can also have
Mike email you a short tip each week.

You may also be interested in:

What Are Agile Story Points?

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

Epics, Features and User Stories

I've been getting more and more emails lately from people confused about the difference between ...

Nov 08, 2022 • 93 Comments [Read](#)

Relationship between Definition of Done and Conditions of Satisfaction

I'd like to clarify the relationship between two important concepts: a team's Definition of Done ...

Jun 28, 2022 • 38 Comments [Read](#)

Four Reasons Agile Teams Estimate Product Backlog Items

Estimating product backlog items provides benefits beyond predicting when a project will be finished.

May 17, 2022

[Read](#)

How Detailed Should a User Story Be?

Capturing too much or too little detail in a user story causes problems. Here's how to get it ...

May 04, 2021

[Read](#)

Agile Requirements Gathering: Three Types of Requirements

Deadlines are often missed because teams fail to consider emergent requirements. Learn about the 3 ...

Feb 02, 2021

[Read](#)

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

by
Mike Cohn
Tagged:
product backlog, user stories, acceptance criteria
9 Comments

An extremely common mistake people make with writing good [user stories](#) has to do with including the right amount of detail at the right time.

It's an important balancing act. Bring an agile user story into an iteration before it is sufficiently understood and there will be too many open issues for the team to complete it within the iteration. But try to resolve every open issue and add every little detail to a story and you end up with functionality that takes far too long to make it into the hands of users.

Like Goldilocks and the bears, we don't want items in the [product backlog](#) with too little or too much detail—we want detail that is just right.

The Right Amount of Detail for User Stories

To understand the right amount of detail for user stories, or more generally any product backlog item, consider an example.

I had groceries delivered a few minutes ago. My order included five apples. I requested Honeycrisp apples, and that's exactly what I got.

I didn't specify that the apples should have a diameter of exactly three inches. I didn't specify apples that were firm and not moldy.

When placing the order, I had to provide the right level of detail to ensure I got what I wanted: the variety of apples and how many.

It is much the same for a [product owner](#) when asking a team to build a new feature. Product owners need to provide the right level of detail, at the right time, to get what they want

Problems with Too Little Detail

If a product owner writes a user story and includes too little detail, the [development team](#) won't know enough during [sprint planning](#) to understand what to build. This means:

- The team will use valuable time during the sprint asking questions
- Team members may build something incorrectly
- Estimates are more likely to be wrong
- Commitments to stakeholders or customers based on those estimates are likely to be wrong

As bad as these problems can be, scrum teams need to be careful not to solve them by swinging to the opposite extreme and adding too much detail.

Problems with Too Much Detail

"User stories are written with too much detail" might at first sound like an oxymoron—you can't be too rich, too thin, or have too much detail on a product backlog item. Except it is entirely possible to have too much detail in product backlog items.

When excessive detail is included, the time and money spent adding the unnecessary detail is wasted. Telling my grocery shopper to find apples that were three inches in diameter would have been a waste of my time and the shopper's, unless there was some rare circumstance that truly required apples of precisely that diameter.

Perhaps worse than the wasted time and money is that developers may feel artificially constrained by the excessive detail. Suppose I've provided a grocery shopper with the following specification:

- Buy five apples
- Honeycrisp variety
- Three inches diameter
- Firm
- No bruises

What will the shopper do if there are not five apples that meet all these requirements? Is it OK to select a slightly larger last apple? Or is the three-inch diameter most important? Would I prefer a larger apple without bruises to an apple of the perfect size with a bruise?

This may seem trivial with apples. It's not when developing a product. When a team is provided a lengthy list of requirements a user story must fulfill, many teams will resist deviating from anything on that list even if a

better way of building a feature is possible.

Get User Story Details Right, Iteratively

It's unlikely a team's product backlog will be detailed perfectly right off the bat. This means the team will likely have to iterate toward the right amount of detail.

I find it much easier for team members to find the right amount when they start with too little detail. Maybe start by filling in the [user story template](#) with the bare minimum amount of product features and detail.

When a team instead starts with *too much* detail, team members may not even notice there's a problem. Remember, the big problem with excessive detail is the time is wasted collecting that detail before it's needed. A team may, however, not notice this or think of it as wasteful. Team members may just see their iterations running smoothly, with very few last-minute discoveries derailing them, and be content with that amount of detail.

When a team starts with *too little* detail, though, they'll notice the problems it creates. Team members will struggle to finish all items within the iteration; there are too many open issues to be resolved. They may feel they are being asked to estimate something when they know only the basics of what will be required.

Because these are real problems with obvious implications, it is clear to team members when they're gathering too little detail. And the solution is obvious: gather more detail.

By starting with insufficient detail on its product backlog items, a team will be able to quickly inspect and adapt to find the right amount of detail.

What Does the Right Amount of Detail Look Like?

When a product backlog item includes the right amount of detail, team members will feel as though they were just barely able to finish the item during the iteration. They'll feel as though enough details had been determined in advance but not so many that the creativity had been removed from the iteration.

In striving for these feelings it's important to not tip over to the side of adding too much detail too early to product backlog items.

Example of a Detailed User Story

It will be helpful to look at an example detailed user story. Suppose a team is building a new application and is working on a user story that says **members are allowed to log into the system**. Various acceptance criteria, otherwise known as [conditions of satisfaction](#), are identified for that story stating that a user:

- Must provide a unique email address
- Must specify a strong password
- Can log in through their Facebook account

We can learn several lessons from this example user story.

Lesson 1: Not Everything Needs to Be Known Before Starting

When thinking about working on a user story, it is useful to realize that the team doesn't need to know everything before it starts a story. All open questions need to be answered before a story is *finished*, but not all need to be answered before the story is *started*.

Consider the requirement to provide a strong password. Very little detail is provided in that statement.

How long must the password be? Does it need to include any special characters? Which special characters? How many? Can a user change their password to a previously used password?

The team members who will code and test this user story need to know the answers to those questions. But they do not need to know the answers to those questions before starting on the user story that enables users to log in.

The answers can be uncovered after team members begin working on the story. Perhaps these details are added in the hour after a programmer and tester begin work on the story. They go talk to the company's chief security officer with a list of questions and return knowing the detailed answers.

The key is that "Must specify a strong password" is sufficient when the story is first written. The details about what constitutes a strong password can be determined later.

(I'm open to being wrong about that in the case of some super-duper secure system. But, for the majority of systems most of us log in to, I've described the situation accurately.)

Lesson 2: Too Much Detail Too Early Is Harmful

Another acceptance criterion for our log in story said that users could log in through their Facebook accounts. This is an example of providing too much detail too early.

When the log in story was first written, it would have been better to state merely that *users can log in through social media* rather than *users can log in through Facebook*. For the most part, there would be no reason to determine upfront that Facebook is the only additional way a user can log in.

The Joy of the Right Amount of Detail

Learning to include the right amount of detail on product backlog items is a strong indicator that a team and its product owner have really grasped what it means to be agile.

Everyone will realize that by trying to add the right amount of detail, sometimes too little will be included. And that means some product backlog items won't be completed within their iterations.

But that will be fine because everyone will understand that avoiding that would require sometimes including too much detail. And the team, product owner, and stakeholders will be aware that adding too much detail is worse than occasionally missing an item because too little detail was included.

How Is Your Team Doing?

How is your team doing at including the right amount of detail? What problems have you seen from including too little or too much detail? What has your team done to include the right amount of detail?

Get 200 Real Life User Stories Examples Written by Mike Cohn

See user stories Mike Cohn wrote as part of several real product backlogs.

First Name

Email Address

[Privacy Policy](#)

Posted: May 4, 2021

Tagged: product backlog, user stories, acceptance criteria

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and*

Planning, and Succeeding with Agile as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

[User Stories: How to Create Story Maps](#)

Story maps help to create a shared understanding of the product, visualize user needs, and elicit ...

Mar 14, 2023

[Read](#)

[How to Run a Successful User Story Writing Workshop](#)

What you need to know to conduct a story-writing workshop with your team.

Feb 28, 2023 • [3 Comments](#)

[Read](#)

[What Are Agile Story Points?](#)

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • [122 Comments](#)

[Read](#)

Epics, Features and User Stories

I've been getting more and more emails lately from people confused about the difference between ...

Nov 08, 2022 • [93 Comments](#)

[Read](#)

Advantages of User Stories over Requirements and Use Cases

At the surface, user stories appear to have much in common with use cases and traditional ...

Oct 05, 2022 • [41 Comments](#)

[Read](#)

Relationship between Definition of Done and Conditions of Satisfaction

I'd like to clarify the relationship between two important concepts: a team's Definition of Done ...

Jun 28, 2022 • [38 Comments](#)

[Read](#)

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile Assessment		

by
Mike **Tagged:**
Cohn product backlog, requirements
Comments

There's a grand myth about requirements: If you write them down, users will get exactly what they want.

That's not true. Even with the best set of project requirements in place, users will get exactly what was written down, which may or may not be anything like what they really want.

That's one of the reasons agile project management frameworks, like Scrum, forego a lengthy, upfront requirements phase and the resulting product specification in favor of a dynamic [product backlog](#), often written in the form of [user stories](#).

Product Backlog vs Requirements

Product backlogs don't eliminate project requirements or the need to work with stakeholders and customers to gather requirements. They do, however, help to account for three truths:

- Teams can never know every requirement upfront
- Conversations are the most effective way to share information
- Risks are less risky when they are uncovered early

Three Types of Requirements

In discussing agile and requirements management, it's important to realize there are really three different types of requirements: known, overlooked, and emergent.

1. Known Requirements

First are the known requirements. **Known requirements** are ones users tell us about. Business analysts and product managers have honed good requirements gathering techniques for agile projects: interviews, [story-writing workshops](#), open-ended questions, and more.

2. Overlooked Requirements

Overlooked requirements are what we call the requirements that we missed during our talks with users. We're fallible. We don't always ask good questions or the right type of questions. Users seldom think of everything. Maybe a user interview was cut short, or a user was absent from a story-writing workshop.

Overlooked requirements are just as important as the known requirements, but we somehow missed them, didn't hear them, or the [stakeholders](#) never mentioned them.

3. Emergent Requirements

There's one more type of requirement that no requirements gathering technique can uncover. It isn't something we overlook or something known. **Emergent requirements** are ones that surface through the act of building the product.

The team demonstrates what has been developed so far, and users say, "What would really make this great is..." Emergent requirements develop as we learn more about what we're creating.

Emergent requirements are not things the team should have uncovered during a story-writing workshop or from interviews. They are not things the development team could have identified if they'd just thought harder or longer when asked about what they need.

Examples of the 3 Different Requirements Types

Suppose you're at a grocery store, doing your shopping for the week. Items on your shopping list represent your *known requirements*. You knew they were needed, so you added each to the list.

As you walk the aisles, you see orange juice and realize you'd forgotten to put that on the list. Orange juice is an *overlooked requirement*.

Then, you round the corner and put something in your cart that you didn't even know you needed? It wasn't on your list (a known requirement) and it wasn't something you'd merely overlooked. Instead, you found something you didn't even know you wanted until you saw it. It was an *emergent requirement*.

This happened to me about two years ago. I noticed something that looked like a huge, pimpled orange. It was called a Sumo.

A store employee offered me a sample. It was delicious. And I immediately put four Sumos in my cart.

Before I saw them and tasted one, I had no idea I wanted one. I didn't even know Sumos existed. On that shopping trip, Sumos represented the third type of requirement: an emergent requirement.

The same thing happens on projects. Having seen a partial implementation, users identify new things the product should do. New business requirements, solution requirements, and stakeholder requirements come to light that could never have been anticipated.

Risk & Planning for the Unknown

Emergent requirements often cause projects to be delivered late. Since emergent requirements cannot be eliminated, the best strategy is to seek them out as early in the development process as possible.

This means that product owners should consider parts of the system that are most likely to include [emergent requirements alongside the most desirable features](#) when prioritizing their product backlogs. That way, teams can put working product in the hands of their users and surface any emergent requirements.

The Universe of Requirements

The universe of requirements can be conceptualized using the following figure. On the left are the known requirements that come out of our discussions with users and other stakeholders. On the right are the overlooked and emergent requirements, which collectively make up a project's *unknown requirements*.

Every project—well, maybe not a rewrite of Minesweeper—has emergent requirements. No matter how adept team members are at asking questions or how thoroughly users have thought about their own needs, not everything can be identified upfront.

Teams can often do better at reducing the number and significance of overlooked requirements. We can ask better questions, listen more actively, spend adequate time with users, and so on. But a truly emergent requirement is one that a team cannot rightfully be expected to have uncovered until users start seeing early

versions of the product.

How Have You Handled Emergent Requirements?

Emergent requirements often cause projects to be delivered late. Since the needs haven't been discovered yet, teams often fail to consider them when planning. Has a project you worked on been affected this way by emergent requirements? How have you handled them?

Please share your thoughts in the comments section below.

[Download my PDF](#)

Posted: February 2, 2021

Tagged: product backlog, requirements

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

[What Are Agile Story Points?](#)

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

[Epics, Features and User Stories](#)

I've been getting more and more emails lately from people confused about the difference between ...

Nov 08, 2022 • 93 Comments [Read](#)

[Relationship between Definition of Done and Conditions of Satisfaction](#)

I'd like to clarify the relationship between two important concepts: a team's Definition of Done ...

Jun 28, 2022 • 38 Comments [Read](#)

[Four Reasons Agile Teams Estimate Product Backlog Items](#)

Estimating product backlog items provides benefits beyond predicting when a project will be finished.

May 17, 2022

[Read](#)

[Needs, Wants, and Wishes on Your Product Backlog](#)

To prioritize a backlog, think about what users need, what they want, and what they merely wish for.

Oct 12, 2021

[Read](#)

[How Detailed Should a User Story Be?](#)

Capturing too much or too little detail in a user story causes problems. Here's how to get it ...

May 04, 2021

[Read](#)

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

by
Mike Cohn **Tagged:**
product backlog, user stories, job stories
42 Comments

As useful as [user stories](#) can be, they've never been right for every team. An exciting alternative for some teams is the *job story*. A job story is focused less on the user performing some function than on the job to be done by that story. Job stories originated at [Intercom](#) and were best explained by [Alan Klement](#).

A Job Story Template

To see how a job story shifts emphasis from the user to a job to be done, let's take a look at the recommended job story template:

As with the common [user story template](#), there are three parts to complete in the job story template. The first is known as the *situation*. This follows the word *when* in the template and provides context on when the story is being performed or perhaps what triggered the story. Examples could be:

- When an order is submitted...
- When searching by postal code...
- When no matches are found...
- When looking at recent orders...

The second element of the job story template follows the *I want to* and provides the *motivation* for the story. Think of the motivation as a user's stated or first-order goal.

As an example, I want pizza for dinner tonight. Why do I want pizza tonight? Because I am meeting some friends tonight to watch a football game, and it's easy to feed a group of us with different dietary needs and preferences with pizza. In the job story world, easily feeding a group is referred to as the *expected outcome* and it follows the so *I can* portion of the template.

Putting my desire for pizza into a job story would lead to

When it's dinner time tonight, I want to have pizza so I can easily feed my friends.

This isn't a particularly perfect job story, but it illustrates the difference between a user's motivation and their expected outcome.

Contrasting Job and User Stories

To see the times when job stories may be better than user stories, let's look at some sample job stories and their corresponding user stories.

[When an Order Is Submitted...](#)

Let's start with this job story:

Job Story:

[When an order is submitted, I want to see a warning message so I can avoid resubmitting the order.](#)

This story describes the behavior seen on most eCommerce sites warning a user not to submit an order multiple times.

The user story equivalent of this might have been

User Story:

[As a customer, I want to be shown a message telling me not to submit an order twice so that I don't place a duplicate order.](#)

The job story is superior in this case for two reasons. First, this story applies to everyone making a purchase on the site. So it's not important to know the person doing this is a customer. (In fact, calling the person a customer could be misleading because the person may not be a customer until this order has been placed.)

Second, the job story is better because it provides more context about when this is happening. It is happening "when an order is submitted," as the job story tells us. Look carefully at the user story and you'll notice that it never tells us when this message is displayed. The team could "successfully" implement the user story by adding an item on an FAQ page warning against double submitting orders. That is almost certainly not what the product owner wants.

When an Order Is Submitted...

Let's look next at a job story about searching for an address by United States ZIP (postal) code.

Job Story:

When searching by US ZIP code, I want to be required to enter a 5- or 9-digit code so I don't waste time searching for a clearly invalid postal code.

This story is about making sure a user enters something reasonable for a postal code before allowing a search. The United States postal, or ZIP, codes are either 5 or 9 digits. This story says users should be prevented from clicking search with only entering two letters entered in the postcode field.

The equivalent user story would be:

User Story:

As a user, I want to be required to enter a 5- or 9-digit postal code so I don't waste time searching for a clearly invalid postal code

These two stories highlight that the difference between user and job stories exists in the first part of the templates. The when... and As a ... clauses differ but in this example, the remainder of each story is identical in both user and job story format.

As in the first example, the job story is better here because of the additional context it provides around *when* the story is being performed. Who is performing the action (the search in this case) is not important, which is why the user story is written with the generic, "As a user..."

When to Use Job Stories

In deciding when to use job stories, I think it's important to acknowledge that both user and job stories have unique strengths.

I still find user stories most helpful for products that have users who vary significantly and deeply understanding those users is important. This is why user stories start with As a... We start user stories that way because that puts the user right upfront. With user stories, *who* will be doing the story is perhaps as

important as *what* they'll be doing.

In contrast for a job story, it is not necessarily important who is doing the story. This makes job stories the better option when your product has users but their needs are not very distinct.

If you've ever written a lengthy set of user stories and started each with "As a user...", you've encountered this problem. When a large set of user stories all begin with "As a user..." you've got a set of stories for whom the user is not very important.

Writing those as job stories rather than user stories would be helpful. Doing so would allow the story to include the additional context of when the story is being performed. In some cases, knowing when a story might happen is more important than knowing who will perform it.

Combining the Strengths of Job and User Stories

So, while job and user stories each have their own strengths, it's possible to merge them and get both benefits in one story. Let's revisit our postal code stories and see how to do this. First, we had the job story:

Job Story:

When searching by postal code, I want to be required to enter a valid code so I don't waste time searching for a clearly invalid postal code.

It's never clear who is performing this story. Is it a normal user? A site admin? Someone else? We're never told. If we think knowing who is doing this is important, we can augment the job story by adding a role to the story in place of I. This changes our story to be the following:

Job Story:

When searching by postal code, a buyer wants to be required to enter a valid code so the buyer doesn't waste time searching for a clearly invalid postal code.

The changes are in bold. You can see that I just changed from "I want..." to "a buyer wants..." and then made the corresponding change later in the story.

We can do something similar to a user story. Our initial, unmodified story was:

User Story:

As a user, I want to be required to enter a valid postal code so I don't waste time searching for a clearly invalid postal code

To provide additional context, we add a phrase telling us when the user is doing this. Our modified user story then becomes "

User Story:

As a user who is searching by postal code, I want to be required to enter a valid postal code so I don't waste time searching for a clearly invalid postal code

The modified user and job stories are semantically the same. Which you choose is entirely up to you. I personally prefer the modified user story over the modified job story because it keeps the story in first person. I've written elsewhere about the [benefits of stories being in the first person](#).

When to Use Job Stories

So when should you prefer user or job stories over the other?

First, each is great and has its own advantages. During the course of any week, I will write some user stories and some job stories. The two techniques are quite compatible and there's no reason to view them as mutually exclusive.

If your product has users and those users needs differ in important ways, I suggest user stories. The additional emphasis a user story puts on who is performing the action can lead to insights about user behavior.

If, however, your product's users do not differ in significant ways, job stories are likely the better approach.

A good starting point is to mix user and job stories in the same product backlog. Start by writing job stories any time you are tempted to write a batch of stories all beginning with "As a user..."

What's Your Experience?

What do you think of job stories? Would they be helpful on your product backlog? Have you worked with job stories before? Were they helpful? Please share your thoughts in the comments section below.

[Download my PDF](#)

Posted: October 29, 2019

Tagged: product backlog, user stories, job stories

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

by
Mike Cohn **Tagged:**
23 product backlog, user stories
[Comments](#)

There is no magic template that must be used for [user stories](#). They can be written in any number of ways. But the most popular way of writing user stories is with this template:

As a ..., I ... so that

This template originated with agile coach Rachel Davies at an English company, Connextra, in the early 2000s. It has since become a recognized standard for expressing user stories.

In this article, let's take a look at the three elements of this standard template, why that template has stood the test of time, and the strengths and weaknesses of the template.

The Three Elements of the Standard Template

I've described the three parts of the standard template in various ways over the years. In [User Stories Applied](#), I described the three elements this way:

As a (role), I want (function) so that (business value).

I later shifted to referring to the three elements as the role, goal, and reason. Ultimately I settled on just referring to them much more simply as who, what, and why. The three elements of the standard user story template address:

- Who wants the functionality
- What it is they want
- Why they want it

Let's look at each of these parts of the user story template.

Who: The First User Story Template Element

The users of a typical product or system can generally be categorized. As an example, I'm typing this into Google Docs right now. I could be considered a *casual user* of Docs. I don't use most of its features. I've never clicked on its Add-Ons menu to even see what's there. I don't do a lot of fancy formatting because most of what I write gets moved onto my website and is formatted there. So, let's call me a *Casual User*.

Others who do use those features could be called *Power Users*.

I usually send my blog drafts to an editor. And so, *Editor* could be another role when identifying the various different types of Google Docs users.

These user roles form the first part of the standard user story template. So someone developing a word processor might have a user story of "As a *power user*, I want a spellchecker..."

What: The Second User Story Template Element

The second part of the standard template states what is desired or needed. This is commonly stated as "I want...." In fact, for years my conference presentations and other trainings on user stories had a slide saying "I want..."

I eventually realized that was inaccurate. Sometimes the functionality being described is not something the user role wants at all. For example:

As a member, I am required to enter a strong password....

No (normal) user *wants* to enter a password with lots of characters, three special symbols, no repeating characters, and at least a couple of uppercase letters. At best, we understand why it's needed, but personally I'd prefer the system should just magically know it's me and let only me in.

So, these days I no longer include *want* when showing the template in courses or presentations. Instead of always being I *want*, sometimes it's I'm *required*, *forced*, *need to* or more.

Why: The Third User Story Template Element

The final part of the standard template is *why* the user wants the functionality being described in the user story. This is provided after the *so-that* portion of the template. For example, a fully expressed version of the

earlier spell checker story could be “As a power user, I want a spell checker so that I don’t need to worry about spelling mistakes.”

The *so-that* clause of a story is important because understanding why a user wants what is described in the *what* portion of the story helps the team decide how best to implement the functionality.

As an example, look no further than our spell checker story. Suppose it were provided to a team without a *so-that* clause as simply: *As a power user, I want a spell checker.* That very likely would lead a team to develop what all early spell checkers were: After-the-fact tools run on a document after it was written.

But our power user doesn’t want an extra step after finishing writing a document. Rather, the power user wants what seems more standard today: real-time correction of mistakes as they are made. What the user really wants is given by the *so-that* clause: the user doesn’t want to worry about spelling mistakes.

Should the So-That Clause Be Optional?

When I present on user stories during in-person courses, I am often asked to justify my seemingly contradictory view that the *so-that* clause is the most important part of a story yet it shouldn’t be mandatory.

I don’t consider it mandatory because sometimes it just doesn’t add any value. Consider a story about logging in: *As a member, I am required to log in.* What *so-that* clause can you add to this story that adds value or clarifies the intent of the story? Do any of these really help or are they just superfluous text added to comply with a template:

As a member, I am required to log in so that only I can access my personal data.

As a member, I am required to log in so that others can’t access my personal data unless I’ve given them my credentials.

As a member, I am required to log in so that the system knows it’s me.

As a member, I am required to log in so that hackers are kept out.

While I don’t consider the *so-that* clause mandatory, I do write it nearly all of the time. I reviewed a recent backlog I’d written and 62 of 64 stories (97%) had *so-that* clauses. A small number of occasions prevent me from considering it mandatory.

Strengths of The 3-Part User Story Template

So what’s so good about this template that it has stood the test of time? After all, a practice introduced in the early 2000s and growing in acceptance so many years later must have something going for it. I think there are four main strengths to the template.

It Covers Three of the Five Ws

I started university as a journalism major. I think I romanticized movies filled with cigar-chomping newspaper reporters—films like *Ace in the Hole*, *Citizen Kane*, *It Happened One Night*, and *All the President’s Men*. Journalists are trained that any newspaper article needs to answer five questions that each begin with a W:

Who?

What?

When?
Where?
Why?

User stories address three of the five--Who, What and Why. When discussing product or system requirements, it seems reasonable to leave When and Where out as the answers would usually be "right now" and "in the product."

The Elements Are Presented in the Right Order

I think the elements--who, what, and why--are presented in the right order. Think about any story--not a user story but a movie, a novel, or an anecdote or joke you want to tell a friend. The most important thing in that story is almost always who is doing it. We call that person the *protagonist*.

When we watch a movie, we need to care about the protagonist before we care about the plot. I don't care one way or the other the Death Star being blown up until I see a little of myself in Luke, Han or Leia and can *relate* to them.

Only after I know *who*, do I care about *what* and *why*. The standard user story template puts these in that order.

The Story Is Told from a First-Person Perspective

We like stories about ourselves. (Well, unless we're teenagers and our parents are telling stories about the not-so-cute things we did when we were babies.) The next best thing to a story about me is a story about you. The least interesting is a story about the guy across town whom I've never met.

Stories about I and you and we and she and he are interesting.

The Beatles knew this. They very deliberately stuffed as many personal pronouns as they could into their song titles. And if they couldn't fit a personal pronoun in the song title, they put as many as they could in the song's lyrics.

The Beatles' first British album had pronouns in 8 of 14 song titles (57%). In the 19 minutes and 30 seconds of those eight songs The Beatles used 325 personal pronouns, one pronoun every 3.6 seconds.

That worked so well their second album had pronouns in 64% of the titles. The Beatles then pushed it to 79% on their third album.

In an interview with Billboard magazine, Beatle Paul McCartney said this was very deliberate: "All our early songs contained 'me' or 'you.' We were completely direct and shameless to the fans: *Love Me Do*, *Please Please Me*, *I Want to Hold Your Hand*."

The standard user story template starts with *I*, the most personal of personal pronouns. I have no basis for this claim--I'm no brain scientist--but I swear something happens when we have a user story that starts with *I*. We relate to that story more closely than we would if the same thing were written as a traditional *The system shall...* statement.

Paul McCartney and John Lennon knew this and used it to boost their careers. Agile teams do the same when using the three-part user story template.

Our Stakeholders Are Immediately Comfortable Filling in

The Blanks Before user stories came along, I loved use cases. Or I tried to love use cases. I really did love them. But I could never get the stakeholders with whom I worked to love them as much as I did.

They were just too far afield from how stakeholders thought about their work. Stakeholders don't think about secondary actors or preconditions or postconditions. And so use cases never worked as well in practice as I thought they should.

I've never had that problem with user stories. I can conduct a story-writing workshop with stakeholders simply by writing As a _____, I _____ so that _____ on a whiteboard and telling them we've gathered to fill in the blanks as many times as we can.

Stakeholders get it. It's a natural way of speaking for them.

Other templates for expressing stories have been proposed, and some have advantages, but most are less natural ways of speaking. For example, Behavior-Driven Development is a great way for expressing tests or specifying by example. [Martin Fowler](#) describes its given-when-then syntax as "a style of representing tests." And it's fantastic for writing [test specifications](#) but it's not as good for communicating with customers. In part this is because its template is less natural. I've been speaking since I was 2 or 3. I don't know if I've ever started a sentence with *given*. I've definitely never used *given*, *when* and *then* in that order in a sentence. I've countless said "I want *this* so that *that*."

Drawbacks of The 3-Part User Story Template

There are two drawbacks to the standard user story template that are worth mentioning.

Too Many Stories Are Written as Just "As a user..."

Too often team members fall into a habit of beginning each user story with "As a user..." Sometimes this is the result of lazy thinking and the story writers need to better understand the product's users before writing so many "as a user..." stories.

But other times it may indicate a system not well suited to user stories. This happens because too many people associate being agile with writing user stories. To be agile, they reason, you have to write user stories even when there are no users.

I worked on a financial compliance tracking system. The vast majority of what the system did would never be seen or reported. If a compliance issue was discovered, however, reports would be generated and individuals would be notified. This system benefited from user stories for that small subset of the product's overall functionality. But user stories were inappropriate for the rest of the system.

In cases like these, teams need to use alternative ways of expressing what the product needs to do. The syntax used by job stories or Feature-Driven Development's [features](#) might be better choices.

The Template Is Too Often Slavishly Followed

By all means, common sense needs to be a guiding principle for agile teams. When expressing something in the standard user story template doesn't make sense, don't use that template. Write it another way, including very possibly just free-form text.

Earlier today I wrote a product backlog item of "Change how webinar replay reminders are sent on the last day the way we discussed on Friday's call."

As a product backlog item, that sucks. It's not a user story. It's not BDD or even a job story. It's horrible. And if it doesn't get fixed soon, no one will even remember what bug was talked about on some forgotten phone call.

But, I knew the team would fix it soon and so what I wrote was good enough. The last thing I need would be a Scrum Master insisting I write it to follow a template created around the turn of the millennium, no matter how well that template works for most product backlog items.

What Do You Think

As a reader of this blog, I want to know what you think of the three-part story template so that I can learn how you're using it (or not). (See what I did there?) Please share your experiences in the comments section below.

Get 200 Real Life User Stories Examples Written by Mike Cohn

See user stories Mike Cohn wrote as part of several real product backlogs.

First Name

Email Address

[Privacy Policy](#)

Posted: May 28, 2019

Tagged: product backlog, user stories

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile Assessment		

by
Mike Cohn
Tagged:
product backlog, backlog, refinement, grooming
31 Comments

You should always strive to keep your product backlog small and manageable. With too many items on the product backlog, three problems arise:

First, **it is harder to work with an excessively large product backlog.** Time will be lost looking for items. ("I know it's in here somewhere.") Prioritizing the backlog will take longer. Duplicate items will appear as it will be easier to add an item "just in case" rather than be sure it's not already listed.

Second, **the development team barely notices any progress they make.** A team that finishes 10 of 50 product backlog items can sense that they've made progress. A team that finishes the same 10 but out of 1,000 items will not feel the same sense of accomplishment. They will begin to wonder if it would matter if they had only gotten 9 done instead.

Third, **someone spent valuable time creating all those product backlog items.** Having visibility into the future a bit is great. How far depends on a lot of factors but many backlogs attempt to provide insight too far into their products' futures.

Because these problems of an overly large product backlog can be so harmful, I'd like to present four things you can do to keep your product backlog to a more manageable size.

Delete Things You'll Never Do

The first thing you should do to keep your product backlog small and manageable is delete items you'll never do.

I fully acknowledge this can be hard to do. For years, I worried that if I deleted things, I'd walk into the office one day and my team would say, "We're caught up. What next?" and I wouldn't have an answer because I'd deleted all the long-term items.

I was finally able to let this fear go by realizing: *I can come up with new ideas faster than they can develop them.*

I encourage being fairly ruthless in purging items from your product backlog. If you don't think you'll realistically ever do it, just get rid of it.

Otherwise, a product backlog just accumulates more and more over time. That task of "learn to dance the Macarena" that's been on my personal backlog since 1995 just got deleted. By this point, it's not happening.

Keep Items You Aren't Ready for Off the Product Backlog

For a handful of years, I've had a lot of success keeping product backlogs manageable by maintaining a separate list of items that the product owner wants but isn't truly ready to have the team work on.

I refer to this as a *holding tank*. A holding tank allows me to restrict the product backlog to items that are desired immediately. The holding tank contains items that are either not quite ready or may not be needed after all.

I think of the difference like this: The product backlog contains all the things the product owner wants and can answer questions about right now.

The holding tank contains things the product owner probably wants but either hasn't thought through in enough detail or may decide aren't needed after all.

I've found this approach extremely helpful because it leads to an immediate reduction in the size of the product backlog as items are moved from it and into another temporary holding place.

Metaphorically, you can think of the difference between a product backlog and holding tank as simply a line drawn below some item in the product backlog. Above that line are the items the product owner would pay the team tonight to deliver; below it are items that could change or aren't well understood yet.

More practically, what I'll do in whatever tool I'm using is create a second project, name it the holding tank, and move items to it from the product backlog.

Doing this allows the team to focus on the much smaller set of items that form their product backlog.

Periodically Review the Product Backlog

The third step in keeping your product backlog to a reasonable size is to institute a regular review process. Nothing fancy is needed here. It can be as simple as the product owner reviewing the product backlog and deleting or moving items as described above. I think doing this quarterly is best.

Don't Add Things Unless You Plan to Do Them Fairly Soon

Finally, if you want to keep your product backlog small and manageable, you need to be disciplined in not adding items to the backlog unless you fully intend to do them.

Most product backlogs become unwieldy because it was easier for a product owner to say, "I'll put it on the product backlog" than to tell someone their feature would never see the light of day.

To keep a product backlog in good shape, product owners need to learn to say no or not now. I've found that the other tips in this article help them in doing that.

What's Your Experience?

Have you worked with an unwieldy product backlog? Do you have any additional tips for keeping a product backlog small and manageable? Please share in the comments below.

[Download my PDF](#)

Posted: March 5, 2019

Tagged: product backlog, backlog, refinement, grooming

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the *Better User Stories* video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

What Are Agile Story Points?

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

Epics, Features and User Stories

I've been getting more and more emails lately from people confused about the difference between ...

Nov 08, 2022 • 93 Comments [Read](#)

Relationship between Definition of Done and Conditions of Satisfaction

I'd like to clarify the relationship between two important concepts: a team's Definition of Done ...

Jun 28, 2022 • 38 Comments [Read](#)

Four Reasons Agile Teams Estimate Product Backlog Items

Estimating product backlog items provides benefits beyond predicting when a project will be finished.

May 17, 2022

[Read](#)

Needs, Wants, and Wishes on Your Product Backlog

To prioritize a backlog, think about what users need, what they want, and what they merely wish for.

Oct 12, 2021

[Read](#)

How Detailed Should a User Story Be?

Capturing too much or too little detail in a user story causes problems. Here's how to get it ...

May 04, 2021

[Read](#)

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

by
Mike Cohn **Tagged:**
product backlog, product owner
22 Comments

At any given time, an [agile product backlog](#) will contain a prioritized list of desired features, each sized differently and [written at varying levels of detail](#). Because the product backlog is [prioritized](#), the smaller, high priority items reside at the top of the list, while the larger, less urgent items fall toward the bottom.

As such, product backlogs tend to take on the shape of an iceberg, as shown in the image below.

At the top of the product backlog iceberg are the high-priority features the team will implement relatively soon. These items should be small and should contain sufficient detail that each can be programmed, tested, and integrated within a single sprint. As we look further down the product backlog iceberg (and therefore further into the future), items on the backlog become increasingly larger and less detailed as we approach the waterline. Teams and product owners often have only a vague idea of what lurks beneath there; some of those features are only known in enough detail that each can be estimated approximately and prioritized.

Why Create a Product Backlog That Evolves Over Time?

It makes some people uncomfortable to have items near the waterline or beneath the surface that aren't well understood. They're used to starting a new project by identifying "all" of the requirements. However, those who are well versed in agile know that, because every project has some emergent requirements, you can never define all of the requirements upfront.

A much more agile approach to requirements is to create an iceberg-shaped product backlog that is progressively refined over time. Here's why.

Things will change.

Over the course of a project, priorities will shift. Some features that were initially thought to be important will become less so as the system is shown to potential users and customers. Other needs will be discovered and have to be properly prioritized.

If we acknowledge that change is inevitable, the advantages of structuring your product backlog like an iceberg become more apparent.

The features most likely to change are those that will be done further into the future. To account for the increased likelihood of change, these features are described only at a high level.

There's no need.

Novelist E. L. Doctorow has written that “writing a novel is like driving at night in the fog. You can only see as far as your headlights, but you can make the whole trip that way.”

Software development is the same way. My headlights don’t illuminate everything between me and the horizon because they don’t need to. They light the way far enough for me to see and respond at the speeds my car can safely travel.

The iceberg-shaped product backlog works similarly. Enough visibility is provided into upcoming items that teams see far enough into the future to avoid most issues. The faster a team goes, the further ahead in the product backlog it will need to peer.

Time is scarce.

Nearly all projects are time constrained. We want more than will fit in the time allotted. Treating all requirements as equivalent is wasteful.

With a limited supply of one of a project’s most critical resources (time), we need to be protective of it.

If it is sufficient for now to describe a future feature at a high level, this is all that should be done.

When that future feature needs to be better understood—whether because it has moved to the top of the product backlog or because we expect it to influence the implementation of another feature—we can describe it in more detail.

This is not to say that a team cannot choose to put some time into understanding items further down on the product backlog iceberg. In fact, doing so is often necessary.

If the team thinks an item further down the product backlog may have an impact on items above it, the team can put some effort into understanding it. This often results in the item being split into multiple, smaller product backlog items.

However, given our history of favoring up-front understanding of all features, teams should be careful to make sure there is a real need to better understand an item before putting more early effort into it than would otherwise be warranted based on the item’s position on the product backlog.

How to Progressively Refine the Product Backlog

Now that we understand why an iceberg-shaped product backlog is desirable, let’s spend some time talking about how to manage the product backlog.

As high-priority items are brought into sprints for development, they are removed from the top of the product backlog iceberg. As such, the iceberg develops a flat spot and begins to lose its shape.

To counter this effect, teams and product owners must spend time reshaping the product backlog—a process known as [product backlog refinement](#) (sometimes referred to as product backlog grooming).

Product backlog refinement can be a regularly occurring, formal meeting or can happen more informally and

frequently during each sprint.

A good rule of thumb seems to be that about 10 percent of the effort in each sprint should be spent refining the backlog in preparation for future sprints.

This time may come from one individual (perhaps an analyst) whose role on the team is largely focused on the backlog. Or it may represent smaller efforts coming from each team member.

Conversations about the product backlog are not limited to a single time or meeting; they can happen any time and among any team members. These conversations enable developers to understand what needs to be built.

What Happens When a Team Refines its Backlog

A few things happen as teams refine their product backlog icebergs.

First, through conversations with the product owner, teams ensure that the items toward the top of the backlog iceberg are well understood, small enough, and sufficiently detailed to be brought into an upcoming sprint.

Remember, that the stories near the top are high priority, so the team and product owner know they need to be implemented in the next sprint or two.

When refining these product backlog items, teams should take care in how they define *well understood* and *sufficiently detailed*.

A good Scrum team does not need a perfect understanding of a feature before it starts working on it. Rather, at the start of the sprint, the team needs to know they have a reasonably strong chance of finishing each feature during the sprint.

Second, the team and product owner might want to look at some of the lower priority items that have come nearer to the top but are still several sprints out. These items might need to be split, rewritten, or even discarded based on what the team has learned in previous sprints.

Lastly, the team might take a look at some of the features that have risen recently above the waterline. These items, which were previously lurking below the surface, are likely to be large and lacking detail.

The team might choose one or two of these items to split into smaller, slightly more detailed stories. The stories that result from this split are likely to still be relatively low in priority, so they don't need to be sprint-ready, but they do need to contain enough detail that they can be estimated more accurately and perhaps reprioritized.

(For a great primer on this whole process, including how to split user stories, check out my [video training series on Better User Stories](#).)

What Happens When a Team Refines its Backlog

Having an iceberg-shaped product backlog should be a goal for any agile team. If you structure your backlog to have small, ready-to-develop items near the top and larger items with less clarity below you'll find your team spending less time overall in product backlog refinement. And the time they do spend will be invested in the items considered most important by the product owner.

What's Your Experience?

What does your product backlog look like? Have you experienced any of the advantages I've described? Please share your thoughts in the comments below.

[Download my PDF](#)

Posted: August 14, 2018

Tagged: product backlog, product owner

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

Be a Great Product Owner: Six Things Teams and Scrum Masters Need

Learn six ways effective product owners ensure their teams' success.

Jan 31, 2023 • 16 Comments [Read](#)

What Are Agile Story Points?

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

Epics, Features and User Stories

I've been getting more and more emails lately from people confused about the difference between ...

Nov 08, 2022 • 93 Comments [Read](#)

Relationship between Definition of Done and Conditions of Satisfaction

I'd like to clarify the relationship between two important concepts: a team's Definition of Done ...

Jun 28, 2022 • 38 Comments [Read](#)

Four Reasons Agile Teams Estimate Product Backlog Items

Estimating product backlog items provides benefits beyond predicting when a project will be finished.

May 17, 2022

[Read](#)

The Product Owner's Second Team

Successful product owners will see their stakeholders as a team, not merely a set of individuals.

Nov 02, 2021

[Read](#)

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

by
Mike Cohn **Tagged:**
6 product backlog, sprinting, scrum master, musing, popular
[Comments](#)

Because I wrote a lot last year--25 blog posts and 50 [weekly email tips](#)--I wanted to start something new this year. So here's a list of the most popular blog posts here during 2016. I hope it helps you catch up on any you missed during the year.

Using our own little algorithm that is a combination of page views, comments and time spent on the pages, here are my top 10 blog posts from 2016, counting down from number 10:

10) Applying Agile Beyond Software Development

Agile can be applied well beyond software development. It's been used for construction, planning weddings, marketing and more. These are my thoughts on how agile could have saved a hotel chain from an expensive mistake.

9) What Are Story Points?

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just one factor--such as complexity, as is often mistakenly claimed. Instead, story points are based on a combination of factors.

8) Advice on How to Split Reporting User Stories

Splitting stories has long been one of the biggest challenges facing agile teams. Here are some examples of splitting some reporting stories to demonstrate ways of splitting stories.

7) Does a Scrum Team Need a Retrospective Every Sprint?

Conventional wisdom says that a team should do a retrospective every sprint. But if your sprints are one week, can you do them every few sprints? That would still be more often than a team doing four-week sprints.

6) How to Prevent Estimate Inflation

A rose by any other name may smell as sweet, but a five-point story better not go by any other names. Or numbers. Here's how to maintain consistency across estimates.

5) Summarizing the Results of a Sprint

Although you may wish it weren't the case, some Scrum Masters need to document how a sprint went. Here's advice on how to do that in a lightweight, agile manner.

4) The Dangers of a Definition of Ready

After seeing the value of a Definition of Done, some teams introduce a Definition of Ready. For many teams, this is a big mistake and a first step towards a waterfall process.

3) Don't Estimate the Sprint Backlog Using Task Points

Some teams like story points so much, they invent task points and use those for sprint planning. Bad idea. Here's why.

2) Sprint Planning for Agile Teams That Have Lots of Interruptions

Most of the Scrum literature describes a situation in which a team is allowed to work without interruption. But that's not realistic. Here's how an interrupt-driven team can plan its sprints.

1) A Simple Way to Run a Sprint Retrospective

There are many ways you can run a sprint retrospective. Here's the simplest way and still my favorite.

What Do You Think?

Please let me know what you think. Is this list missing any of your favorites?

Get Your Free Scrum Cheat Sheet

Sign up below to receive this FREE reference.

First Name

Email Address

[Privacy Policy](#)

Posted: January 17, 2017

Tagged: product backlog, sprinting, scrum master, musing, popular

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

[Six Attributes of a Great ScrumMaster](#)

Scrum Masters exhibit a shared set of traits. Discover 6 in-demand qualities, plus a bonus ...

Jan 17, 2023 • 32 Comments [Read](#)

[What Happens When During a Sprint](#)

Succeeding with Scrum is easier when you know when and why to conduct each of the Scrum events ...

Jan 03, 2023 • 34 Comments [Read](#)

[What Are Agile Story Points?](#)

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

[Epics, Features and User Stories](#)

I've been getting more and more emails lately from people confused about the difference between ...

Nov 08, 2022 • 93 Comments [Read](#)

[From Project Manager to Scrum Master -3 Tips for Making the Transition](#)

What does a good project manager need to do to become a great Scrum Master?

Aug 23, 2022 [Read](#)

[Relationship between Definition of Done and Conditions of Satisfaction](#)

I'd like to clarify the relationship between two important concepts: a team's Definition of Done ...

Jun 28, 2022 • 38 Comments [Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile **Agile & Scrum Training** **More...**

New to Agile and Scrum? Online Certifications About Us

Scrum Video Courses Contact Us

User Stories In-Person Certifications Our Students

Planning Poker On-Site Courses Blog

Agile Software Development Training Schedule Books by Mike Cohn

Development Compare Courses Agile Tools

Agile Project Management PDUs and SEUs

Agile Presentations

Mountain Goat on YouTube

Agile Mentors Podcast

Elements of Agile

Assessment

by
Mike Cohn **Tagged:** product backlog, teams, backlog, definition of ready
64 Comments

Although not as popular as a Definition of Done, some Scrum teams use a Definition of Ready to control what product backlog items can enter an iteration.

You can think of a Definition of Ready as a big, burly bouncer standing at the door of the iteration. Just as a bouncer at a nightclub only lets certain people in—the young, the hip, the stylishly dressed—our Definition-of-Ready bouncer only allows certain [user stories](#) to enter the iteration.

And, as each nightclub is free to define who the bouncers should let into the club, each team or organization is free to define its own definition of ready. There is no universal definition of ready that is suggested for all teams.

A Sample Definition of Ready

So what types of stories might our bouncer allow into an iteration? Our bouncer might let stories in that meet rules such as these:

The conditions of satisfaction have been fully identified for the story.

The story has been estimated *and* is under a certain size. For example, if the team is using story points, a team might pick a number of points and only allow stories of that size or smaller into the

iteration. Often this maximum size is around half of the team's velocity.

The team's user interface designer has mocked up, or even fully designed, any screens affected by the story.

All external dependencies have been resolved, whether the dependency was on another team or on an outside vendor.

A Definition of Ready Defines Pre-Conditions

A Definition of Ready enables a team to specify certain pre-conditions that must be fulfilled before a story is allowed into an iteration. The goal is to prevent problems before they have a chance to start.

For example, by saying that only stories below a certain number of story points can come into an iteration, the team avoids the problem of having brought in a story that is too big to be completed in an iteration.

Similarly, not allowing a story into the iteration that has external dependencies can prevent those dependencies from derailing a story or an entire iteration if the other team fails to deliver as promised.

For example, suppose your team is occasionally dependent on some other team to provide part of the work. Your user stories can only be finished if that other team also finishes their work—and does so early enough in the iteration for your team to integrate the two pieces.

If that team has consistently burned you by not finishing what they said they'd do by the time they said they'd do it, your team might quite reasonably decide to not bring in any story that has a still-open dependency on that particular team.

A Definition of Ready that requires external dependencies to be resolved before a story could be brought into an iteration might be wise for such a team.

A Definition of Ready Is Not Always a Good Idea

So some of the rules our bouncer establishes seem like good ideas. For example, I have no objection against a team deciding not to bring into an iteration stories that are over a certain size.

But some other rules I commonly see on a Definition of Ready can cause trouble—big trouble—for a team. I'll explain.

A Definition of Ready can be thought of like a gate into the iteration. A set of rules is established and our bouncer ensures that only stories that meet those rules are allowed in.

If these rules include saying that something must be 100 percent finished before a story can be brought into an iteration, the Definition of Ready becomes a huge step towards a sequential, [stage-gate approach](#). This will prevent the team from being agile.

A Definition of Ready Can Lead to Stages and Gates

Let me explain. A stage-gate approach is characterized by a set of defined *stages* for development. A stage-

gate approach also defines *gates*, or checkpoints. Work can only progress from one stage to the next by passing through the gate.

When I was a young kid, my mom employed a stage-gate approach for dinner. I only got dessert if I ate all my dinner. I was not allowed to eat dinner and dessert concurrently.

As a product development example, imagine a process with separate design and coding stages. To move from design to coding, work must pass through a design-review gate. That gate is put in place to ensure the completeness and thoroughness of the work done in the preceding stage.

When a Definition of Ready includes a rule that something must be *done* before the next thing can *start*, it moves the team dangerously close to stage-gate process. And that will hamper the team's ability to be agile. A stage-gate approach is, after all, another way of describing a waterfall process.

Agile Teams Should Practice Concurrent Engineering

When one thing cannot start until another thing is done, the team is no longer overlapping their work. Overlapping work is one of the most obvious indicators that a team is agile. An agile team should always be doing a little analysis, a little design, a little coding, and a little testing. Putting gates in the development process prevents that from happening.

Agile teams should practice *concurrent engineering*, in which the various activities to deliver working software overlap. Activities like analysis, design, coding, and testing will never overlap 100%—and that's not even the goal. The goal is overlap activities as much as possible.

A stage-gate approach prevents that by requiring certain activities to be 100% complete before other activities can start. And a definition of ready can lead directly to a stage-gate approach if such mandates are included in the Definition of Ready.

That's why, for most teams, I do not recommend using a Definition of Ready. It's often unnecessary process overhead. And worse, it can be a large and perilous step backwards toward a waterfall approach.

In some cases, though, I do acknowledge that a Definition of Ready can solve problems and may be worth using.

Using a Definition of Ready Correctly

To use a Definition of Ready successfully, you should avoid including rules that require something be 100 percent done before a story is allowed into the iteration—with the possible exception of dependencies on certain teams or vendors. Further, favor guidelines rather than *rules* on your Definition of Ready.

So, let me give you an example of a Definition of Ready rule I'd recommend that a team rewrite: "Each story must be accompanied by a detailed mock up of all new screens."

A rule like this is a gate. It prevents work from overlapping. A team with this rule cannot practice concurrent engineering. No work can occur beyond the gate until a *detailed design* is completed for *each* story.

A better variation of this would be something more like: "If the story involves significant new screens, rough mock ups of the new screens have been started and are just far enough along that the team can resolve remaining open issues during the iteration."

Two things occur with a change like that.

1. The rule has become a guideline.
2. We're allowing work to overlap by saying the screen mockups are sufficiently far along rather than *done*.

These two changes introduce some subjectivity into the use of a definition of ready. We're basically telling the bouncer that we still want young, hip and stylishly dressed people in the nightclub. But we're giving the bouncer more leeway in deciding what exactly "stylishly dressed" means.

[Take the Assessment](#)

Posted: August 9, 2016

Tagged: product backlog, teams, backlog, definition of ready

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and

can be reached at hello@mountaingoatsoftware.com.
If you want to succeed with agile, you can also have
Mike email you a short tip each week.

You may also be interested in:

What Is Cross-Functional Collaboration in Agile?

Perhaps the most prevalent and persistent myth in agile is that a cross-functional team is one on ...

Feb 14, 2023 • 49 Comments [Read](#)

What Are Agile Story Points?

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

Don't Equate Story Points to Hours

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • 39 Comments [Read](#)

Epics, Features and User Stories

I've been getting more and more emails lately from people confused about the difference between ...

Nov 08, 2022 • 93 Comments [Read](#)

Relationship between Definition of Done and Conditions of Satisfaction

I'd like to clarify the relationship between two important concepts: a team's Definition of Done ...

Jun 28, 2022 • 38 Comments [Read](#)

Four Reasons Agile Teams Estimate Product Backlog Item

Estimating product backlog items provides benefits beyond predicting when a project will be finished.

May 17, 2022 [Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

by
Mike Cohn **Tagged:**
product backlog, user stories
46 Comments

A lot of traditionally managed projects begin with a Software Requirements Specification (SRS). Then sometime during the project, a decision is made to adopt an agile approach. And so a natural question is whether the SRS can serve as the newly agile project's product backlog. Some teams contemplate going so far as rewriting the SRS into a product backlog with user stories. Let's consider whether that's necessary.

But before taking up this question, I want to clarify what I mean by a Software Requirements Specification or SRS. I find this type of document to vary tremendously from company to company. In general, though, what I'm referring to is the typical document full of "The system shall ..." type statements.

But you can imagine any sort of traditional requirements document and my advice should still apply. This is especially the case for any document with numbered and nested requirements statements, regardless of whether each statement is really written as "the system shall ...".

Some Drawbacks to Using the SRS as a Product Backlog

On an agile product, the product backlog serves two purposes:

- It serves as a repository for the work to be done
- It facilitates prioritization of work

That is, the product backlog tells a team what needs to be done *and* can be used as a planning tool for sequencing the work. In contrast, a traditional SRS addresses only the issue of what is to be done on the project.

There is no attempt with an SRS to write requirements that can be implemented within a sprint or that are prioritized. Writing independent requirements is a minor goal at best, as shown by the hierarchical organization of most SRS documents, with their enumerated requirements such as 1.0, 1.1, 1.1.1, and so on.

These are not problems when an SRS is evaluated purely as a requirements document. But when the items within an SRS will also be used as product backlog items and prioritized, it creates problems.

With an SRS, it is often impossible for a team to develop requirement 1.1.1 without concurrently developing 1.1.2 and 1.1.5. These dependencies make it not as easy as picking a story at a time from a well-crafted product backlog.

Prioritizing sub-items on an SRS is also difficult, as is identifying a subset of features that creates a minimum viable product.

A Software Requirements Specification is good at being a requirements specification. It's good at saying what a system or product is to do. (Of course, it misses out on all the agile aspects of emergent requirements, collaboration, discovery, and so on. I'm assuming these will still happen.)

But an SRS is not good for planning, prioritizing and scheduling work. A product backlog is used for both of these purposes on an agile project.

My Advice

In general, I do not recommend taking the time to rewrite a perfectly fine SRS. Rewriting the SRS into user stories and a nice product backlog could address the problems I've outlined. But, it is not usually worth the time required to rewrite an SRS into user stories.

Someone would have to spend time doing this, and usually that person could spend their time more profitably. I would be especially reluctant to rewrite an SRS if other teammates would be stalled in starting their own work while waiting for the rewritten product backlog.

A ScrumMaster or someone on the team will have to find ways of tracking progress against the SRS and making sure requirements within it don't fall through the cracks. Usually enlisting help from QA in validating that everything in an SRS is done or listed in the product backlog is a smart move.

One additional important strategy would be educating those involved in creating SRS documents to consider doing so in a more agile-friendly manner for future projects. If you can do this, you'll help your next project avoid the challenges created by a mismatch between agile and starting with an SRS document.

What Do You Think?

Please join the discussion and share your thoughts and experiences below. Have you worked on an agile project that started with a traditional SRS? How did it go? Would it have been different if the SRS had been rewritten as a more agile product backlog?

Get Free User Stories Book Chapters

Please provide your name and email and we'll send you the sample chapters and we'll send a short weekly tip from Mike on how to succeed with agile.

[Get my chapters now!](#)

Posted: January 5, 2016

Tagged: product backlog, user stories

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

User Stories: How to Create Story Maps

Story maps help to create a shared understanding of the product, visualize user needs, and elicit ...

Mar 14, 2023

[Read](#)

How to Run a Successful User Story Writing Workshop

What you need to know to conduct a story-writing workshop with your team.

Feb 28, 2023 • 3 Comments [Read](#)

What Are Agile Story Points?

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

Epics, Features and User Stories

I've been getting more and more emails lately from people confused about the difference between ...

Nov 08, 2022 • 93 Comments [Read](#)

Advantages of User Stories over Requirements and Use Cases

At the surface, user stories appear to have much in common with use cases and traditional ...

Oct 05, 2022 • 41 Comments [Read](#)

Relationship between Definition of Done and Conditions of Satisfaction

I'd like to clarify the relationship between two important concepts: a team's Definition of Done ...

Jun 28, 2022 • 38 Comments [Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile Assessment		

by

Mike Cohn

Tagged: product backlog, user stories, product owner, teams, backlog

41

Comments

My wife, daughters and I use Wunderlist to manage our shared grocery list. Any time one of us notices something we're out of, that person adds it to the grocery list. Then, whoever is taking their turn at doing the weekly grocery shopping checks items off the list while in the store. It works really well.

I'm thinking about this because someone asked me recently about who on an agile team can add items to the product backlog. Can only the product owner add items to the product backlog?

My view is that anyone on a team should be allowed to add to the product backlog. A tester may come with a good new feature idea. The tester is allowed to add that item to the product backlog. The same is true for any other role on the team.

Although anyone can add to the product backlog, it is the product owner who prioritizes what the team works on. This means that while you or I may contribute a new idea to the product backlog, if it's a bad idea, the product owner will probably put it near the bottom of the product backlog (where it may never get worked on).

The product owner may even take it off the product backlog and throw it away if it's something the product owner does not want, or is so low priority that the product owner knows the team will never get to it.

Note that I still consider this as a team having had a chance to add that item to the product backlog. The item was added even if only momentarily. It was then briefly considered before the product owner removed it.

So, anyone can add an item to the product backlog. But it's the product owner who determines what happens to the product backlog item.

When adding groceries to Wunderlist, I like to tease my wife that my dog has gotten a hold of my iPhone and entered his grocery request for "BONEZ." Yes, although, my dog is capable of using Wunderlist on an iPhone, he types in all caps and cannot properly spell "bones."

I'll often enter this two or three times into the grocery list when it's my wife's turn to shop. And as the product owner for that week's shopping, she will promptly ignore her team's request to buy BONEZ.

Very practically speaking, what I recommend product owners do is have a location in the product backlog tool into which team members can add new stories. The product owner then periodically reviews new items in this section, and handles them appropriately.

Some are moved to areas in the tool where they'll be worked on. Others (like "BONEZ") might be deleted as low priority.

In general, a product would suffer if all ideas had to originate with the product owner. New ideas should be valued regardless of the source. The product owner prioritizes what the team works on, but that does not mean that all ideas have to originate in the product owner's head.

[Take the Assessment](#)

Posted: November 10, 2015

Tagged: product backlog, user stories, product owner, teams, backlog

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

[User Stories: How to Create Story Maps](#)

Story maps help to create a shared understanding of the product, visualize user needs, and elicit ...

Mar 14, 2023

[Read](#)

[How to Run a Successful User Story Writing Workshop](#)

What you need to know to conduct a story-writing workshop with your team.

Feb 28, 2023 • [3 Comments](#)

[Read](#)

[What Is Cross-Functional Collaboration in Agile?](#)

Perhaps the most prevalent and persistent myth in agile is that a cross-functional team is one on ...

Feb 14, 2023 • [49 Comments](#)

[Read](#)

[Be a Great Product Owner: Six Things Teams and Scrum Masters Need](#)

Learn six ways effective product owners ensure their teams' success.

Jan 31, 2023 • [16 Comments](#)

[Read](#)

[What Are Agile Story Points?](#)

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • [122 Comments](#)

[Read](#)

Don't Equate Story Points to Hours

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • [39 Comments](#)

[Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Who Can Add Items to the Product Backlog?

Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

by

Mike **Tagged:**

Cohn product backlog, sprinting, velocity, iterations, iterating, unfinished

58 work

[Comments](#)

It's quite common for a team to have a bit of unfinished work at the end of an agile sprint or iteration. Ideally, a team would finish every item on its sprint backlog every sprint. But, for a variety of reasons, that isn't always the case. This leads to a couple of questions I'll address here:

- What should be done with the product backlog item itself?
- Should be it split or should it be carried it into the next sprint?
- Should the team receive any velocity "credit" for completing a portion of the story?

First, Be Sure the Item Is Still Important

When a product backlog item is not finished at the end of an agile sprint, it should first technically be put back onto the product backlog. Work never moves automatically from one sprint to the next.

I'm perhaps being overly pedantic here, but the point is that each sprint begins with the product owner making a conscious, deliberate decision about what the team should work on. If there's unfinished work from the prior sprint, it's very likely the product owner will want the team to finish that work in the new sprint. But, the product owner should still make an actual decision to do that.

(As a note, let me also say that I'm not suggesting you make extra work for yourself if you are using a tool

that would make this difficult. All I'm saying is that work does not automatically move into the next sprint. The product owner must decide if the work is still valuable.)

Splitting or Carrying the Item Forward

When a product backlog is unfinished, teams are often torn on whether they should leave the product backlog item description alone (even though part of that functionality might be complete) or rewrite it to describe just the missing piece.

For example, consider a team building typical print preview functionality in a desktop application. If the team builds everything but the ability to page backward through the pages, it can either carry forward the original user story or write a smaller, replacement story like, "As a user, I can page backward while on the print preview screen."

I recommend that if you are going to finish the story in the coming sprint, just leave it alone. Don't rewrite it. Everyone is used to it as it's written. Assuming it's still of value to the product owner, it moves forward exactly as written.

However, if the remaining work will be deferred to a later sprint, write a new story that describes just that subset of functionality.

Does the Team Earn Velocity Credit

I always want to take a conservative stance towards calculating velocity. This means that a team should only take credit for work that is truly complete.

So, in the case the unfinished product backlog item is rolling forward to be completed in the next agile sprint, do not take any velocity credit. The team instead earns all credit in the sprint in which the work is finished. Since I advocate working with average velocities anyway, this averages out and avoids the risk of overstating velocity.

But when the team splits the story and puts the remaining subset onto the product backlog to be done in the future, go ahead and take some amount of velocity credit. The team will need to do its best to estimate a fair number of points for the subset of work that was completed. And, even though it did not finish the entire original story, the team may give itself all the original points if it feels the story was larger than originally planned. I'd be reluctant to do that very often. But, it is OK.

Always Look for a Root Cause

Finally, whenever work is unfinished at the end of an agile sprint, the team should take time in the retrospective to consider whether it was caused by something preventable. Sometimes, it's just bad luck or bad timing, such as a team member becoming ill or a problem being found late in the sprint that could not have been found earlier. But, it's always worth considering whether there is a root cause and whether something can be done to prevent it from affecting future sprints.

What Do You Do?

How do you handle work left at the end of the sprint? And have you found good ways of minimizing how often you have work left unfinished at the end?

Get Your Free Scrum Cheat Sheet

Sign up below to receive this FREE reference.

First Name

Email Address

[Privacy Policy](#)

Posted: October 6, 2015

Tagged: product backlog, sprinting, velocity, iterations, iterating, unfinished work

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com.

If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

What Happens When During a Sprint

Succeeding with Scrum is easier when you know when and why to conduct each of the Scrum events ...

Jan 03, 2023 • 34 Comments [Read](#)

What Are Agile Story Points?

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

Epics, Features and User Stories

I've been getting more and more emails lately from people confused about the difference between ...

Nov 08, 2022 • 93 Comments [Read](#)

Relationship between Definition of Done and Conditions of Satisfaction

I'd like to clarify the relationship between two important concepts: a team's Definition of Done ...

Jun 28, 2022 • 38 Comments [Read](#)

Four Reasons Agile Teams Estimate Product Backlog Items

Estimating product backlog items provides benefits beyond predicting when a project will be finished.

May 17, 2022 [Read](#)

How Implementation Intentions Help My Sprints

Planning an exact time to do something within a sprint helps make sure you get the important stuff ...

Mar 22, 2022 [Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile **Agile & Scrum Training** **More...**

New to Agile and Scrum? Online Certifications About Us

Scrum Video Courses Contact Us

User Stories In-Person Certifications Our Students

Planning Poker On-Site Courses Blog

Agile Software Development Training Schedule Books by Mike Cohn

Development Compare Courses Agile Tools

Agile Project Management PDUs and SEUs

Agile Presentations

Mountain Goat on YouTube

Agile Mentors Podcast

Elements of Agile

Assessment

Quickly Help Users Identify Their Needs

by
Mike Cohn
18
[Comments](#)
Tagged:
product backlog, user stories, requirements, users

Many projects bog down during requirements gathering. This is dangerous because it wastes valuable time, often setting up a project to be late.

To avoid this problem, I like to engage users and stakeholders in very lightweight story-writing workshops.

Before anyone arrives in the meeting room, I will prepare the room by making sure there are pens and index cards within reach of every chair in the room. Even if you will be using a tool to maintain your product backlog, pen and paper are great for this type of early brainstorming.

I then write on a whiteboard or large sheet of paper:

As a _____

I want _____

so that _____

You'll probably recognize that as the usual format of a user story, but your users may not. So I tell everyone that we're gathered to fill in the blanks. We may start with a brief effort to identify who the products users are (the first of the three blanks). But most of the time will be spent filling in the second and third blanks (what and why).

I find that 90 minutes of this is usually plenty of brainstorming to load up a few months worth of high-priority items in a product backlog. That's a guideline and will, of course, need to be adjusted based on the number of participants, previous agreement on the product vision, and other factors. Save time by avoiding prioritization discussions. Leave that to the product owner to be done outside the story-writing workshop.

Get 200 Real Life User Stories Examples Written by Mike Cohn

See user stories Mike Cohn wrote as part of several real product backlogs.

First Name

Email Address

[Privacy Policy](#)

Posted: August 18, 2015

Tagged: product backlog, user stories, requirements, users

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of User Stories Applied for Agile Software Development, Agile Estimating and Planning, and Succeeding with Agile as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile Assessment		

by

Mike Tagged:

Cohn product backlog, user stories, features, feature driven

49 development

Comments

User stories are great. When you've got users, that is. Sometimes, though, the users of a system or product are so far removed that a team struggles to put users into their stories. A sign that this is happening is when teams write stories that begin with "As a developer..." or "as a product owner...."

There are usually better approaches than writing stories like those. And a first step in exploring alternative approaches is realizing that not everything on your product backlog has to be a user story.

A recent look at a product backlog on a product for which I am the product owner revealed that approximately 85% of the items (54 out of 64) were good user stories, approximately 10% (6 of 64) were more technical items, and about 5% (4 of 64) were miscellaneous poorly worded junk.

Since I'm sure you'll want to know about the junk, let's dispense with those first. These are items that I or someone else on the project added while in a hurry. Some will later be rewritten as good stories but were initially just tossed into the backlog so they wouldn't be forgotten. Others are things like "Upgrade Linux server" that could be rewritten as a story. But I find little benefit to doing that. Also, items like that tend to be well understood and are not on the product backlog for long.

My point here: No one should be reading a product backlog and grading it. A *little* bit of junk on a product backlog is totally fine, especially when it won't be there long.

What I really want to focus on are the approximately 10% of the items that were more technical and were not written as user stories using the canonical "As a , I want so that " syntax.

The product in question here is a user-facing product but not all parts of it are user facing. I find that to be fairly common. Most products have users somewhere in sight but there are often back-end aspects of the product that users are nowhere near. Yes, teams can often write user stories to reflect how users benefit from these system capabilities. For example: *As a user, I want all data backed up so that everything can be fully recovered.*

I've written plenty of stories like that, and sometimes those are great. Other times, though, the functionality being described starts to get a little too distant from real users and writing user stories when real users are nowhere to be found feels artificial or even silly.

In situations like these I'm a fan of the syntax from the Feature-Driven Development agile process. Feature-Driven Development (FDD) remains a minor player on the overall agile stage despite having been around since 1997. Originally invented by Jeff De Luca, FDD has much to recommend it in an era of interest in scaling agile.

Wikipedia has a [good description of FDD](#) so I'm only going to describe one small part of it: features. Features are analogous to product backlog items for a Scrum project. And just like many teams find it useful to use the "As a , I want so that " syntax for user stories as product backlog items, FDD has its own recommended syntax for features.

An FDD feature is written in this format:

[action] the [result] [by|for|of|to] a(n) [object]

As examples, consider these:

- Estimate the closing price of stock
- Generate a unique identifier for a transaction
- Change the text displayed on a kiosk
- Merge the data for duplicate transactions

In each case, the feature description starts with the action (a verb) and ends with what would be an object within the system. (FDD is particularly well suited for object-oriented development.)

This can be a particularly good syntax when developing something like an Application Programming Interface (API). But I find it works equally well on other types of back-end functionality. As I said at the beginning, about 10% of my own recent product backlog I examined was in this syntax.

If you find yourself writing product backlog items for parts of a system and are stretching to think of how to write decent user stories for those items, you might want to consider using FDD's features. I think you'll find them as helpful as I do.

Get Your Free Scrum Cheat Sheet

Sign up below to receive this FREE reference.

First Name

Email Address

[Privacy Policy](#)

Posted: July 21, 2015

Tagged: product backlog, user stories, features, feature driven development

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

User Stories: How to Create Story Maps

Story maps help to create a shared understanding of the product, visualize user needs, and elicit ...

Mar 14, 2023

[Read](#)

How to Run a Successful User Story Writing Workshop

What you need to know to conduct a story-writing workshop with your team.

Feb 28, 2023 • 3 Comments [Read](#)

What Are Agile Story Points?

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

Epics, Features and User Stories

I've been getting more and more emails lately from people confused about the difference between ...

Nov 08, 2022 • 93 Comments [Read](#)

Advantages of User Stories over Requirements and Use Cases

At the surface, user stories appear to have much in common with use cases and traditional ...

Oct 05, 2022 • 41 Comments [Read](#)

Relationship between Definition of Done and Conditions of Satisfaction

I'd like to clarify the relationship between two important concepts: a team's Definition of Done ...

Jun 28, 2022 • 38 Comments [Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile Assessment		

by

Mike **Tagged:**

Cohn product backlog, user stories, teams, sprint planning, sprint

103 backlog

[Comments](#)

What's the difference between a user story and a task?

Well that's an easy question, I thought, the first time it was asked of me in a Certified ScrumMaster class.

"The difference is ...," I began to reply and realized it wasn't actually such an easy difference after all.

I'd been using the two terms, "user story" and "task" in my classes for years, and they seemed pretty distinct in my head. User stories were on the product backlog and tasks were identified during sprint planning and became part of the sprint backlog.

That was fine but wasn't very helpful—it was like saying "salt is what goes in a salt shaker and pepper is what goes in a pepper grinder." Sure, stories go on the product backlog and tasks go on a sprint backlog. But what is the essential difference between the two?

I paused for a second the first time I was asked that question, and realized I *did* know what the difference was. A story is something that is generally worked on by more than one person, and a task is generally worked on by *just one* person.

Let's see if that works ...

A user story is typically functionality that will be visible to end users. Developing it will usually involve a programmer and tester, perhaps a user interface designer or analyst, perhaps a database designer, or others.

It would be very rare for a user story to be fully developed by a single person. (And when that did happen, the person would be filling multiple of those roles.)

A task, on the other hand, is typically something like code this, design that, create test data for such-and-such, automate that, and so on. These tend to be things done by one person.

You could argue that some of them are or should be done via pairing, but I think that's just a nuance to my distinction between user story and task. Pairing is really two brains sharing a single pair of hands while doing one type of work. That's still different from the multiple types of work that occur on a typical story.

I have, however, used a couple of wiggly terms like saying tasks are *typically* done by one person. Here's why I wiggled: Some tasks are meetings—for example, have a design review between three team members—and I will still consider that a task rather than a user story.

So, perhaps the better distinction is that stories contain multiple types of work (e.g., programming, testing, database design, user interface design, analysis, etc.) while tasks are restricted to a single type of work.

Get Free User Stories Book Chapters

Please provide your name and email and we'll send you the sample chapters and we'll send a short weekly tip from Mike on how to succeed with agile.

[Get my chapters now!](#)

Posted: February 24, 2015

Tagged: product backlog, user stories, teams, sprint planning, sprint backlog

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and

techniques to build extremely high-performance teams. He is the author of User Stories Applied for Agile Software Development, Agile Estimating and Planning, and Succeeding with Agile as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

[Why I Don't Emphasize Sprint Goals](#)

Sprint goals are considered a mandatory part of Scrum. Here's why I disagree.

Mar 21, 2023

[Read](#)

[User Stories: How to Create Story Maps](#)

Story maps help to create a shared understanding of the product, visualize user needs, and elicit ...

Mar 14, 2023

[Read](#)

[How to Run a Successful User Story Writing Workshop](#)

What you need to know to conduct a story-writing workshop with your team.

Feb 28, 2023 • 3 Comments [Read](#)

[What Is Cross-Functional Collaboration in Agile?](#)

Perhaps the most prevalent and persistent myth in agile is that a cross-functional team is one on ...

Feb 14, 2023 • 49 Comments [Read](#)

[What Are Agile Story Points?](#)

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

[Don't Equate Story Points to Hours](#)

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • 39 Comments [Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile **Agile & Scrum Training** **More...**

New to Agile and Scrum? Online Certifications About Us

Scrum Video Courses Contact Us

User Stories In-Person Certifications Our Students

Planning Poker On-Site Courses Blog

Agile Software Development Training Schedule Books by Mike Cohn

Development Compare Courses Agile Tools

Agile Project Management PDUs and SEUs

Agile Presentations

Mountain Goat on YouTube

Agile Mentors Podcast

Elements of Agile

Assessment

Focus on Benefits Rather than Features

by
Mike Cohn
Tagged: product backlog, teams, benefits, self organizing teams, features
8 Comments

Suppose your boss has not bought into trying an agile approach in your organization. You schedule a meeting with the boss, and stress how your organization should use Scrum because Scrum:

- Has short time boxes
- Relies on self-organization
- Includes three distinct roles

And based on this discussion, your boss isn't interested.

Why? Because you focused on the *features* of Scrum rather

than its *benefits*.

Product owners and Scrum teams make the same mistake when working with the product backlog. A feature is something that is *in* a product—a spell-checker is in a word processor. A benefit is something a user gets *from* of a product. By using a word processor, the user gets documents free from spelling errors. The spell-checker is the feature, mistake-free documents is the benefit.

It is generally considered a good practice for the items at the top of a product backlog to be small. Each must be small enough to fit into a sprint, and most teams will do at least a handful each sprint.

The risk here is that small items are much more likely to be features than benefits. When a Scrum team (and specifically its product owner) becomes overly focused on features, it is possible to lose sight of the benefits.

Scrum teams commonly mitigate this risk in two common ways. First, they leave stories as [epics](#) until they move toward the top of the product. Second, they include a [so-that clause](#) in their user stories. These help, but do not fully eliminate the risk.

Let's return to your attempt to convince your boss to let your team use Scrum. Imagine you had focused on the benefits of Scrum rather than its features. You told your boss how using Scrum would lead to more successful products, more productive teams, higher quality software, more satisfied stakeholders, happier teams, and so on.

Can you see how that conversation would have gone differently than one focused on short time boxes, self-organization and roles?

Download Scrum Master Guide

Get a free copy of Situational Scrum Mastering: Leading an Agile Team

[Get my free guide now!](#)

Posted: January 20, 2015

Tagged: product backlog, teams, benefits, self organizing teams, features

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile **Agile & Scrum Training** **More...**

New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

If it needs to happen: Schedule it!

by

Lisa Tagged:

Crispin product backlog, meetings, testing, planning, schedule

Comments

The following is a guest post from Lisa Crispin. Lisa is the co-author with Janet Gregory of "[Agile Testing: A Practical Guide for Testers and Agile Teams](#)" and the newly published "[More Agile Testing: Learning Journeys for the Whole Team](#)". I highly recommend both of these books--in fact, I recommend reading everything Lisa and Janet write. In the following post, Lisa argues the benefits of scheduling anything that's important. I am a fanatic for doing this. Over the holiday I put fancy new batteries in my smoke detectors that are supposed to last 10 years. So I put a note in my calendar to replace them in 9 years. But, don't schedule time to read Lisa's guest post--just do it now. --Mike

During the holidays, some old friends came over to our house for lunch. We hadn't seen each other in a few

months, though we live 10 minutes apart. We had so much fun catching up. As they prepared to leave, my friend suggested, "Let's pick a date to get together again soon. So often, six months go by without our seeing each other." We got out our calendars, and penciled in a date a few weeks away. The chances are good that we will achieve our goal of meeting up soon.

Scheduling time for important activities is key in my professional life, too. Here's a recent example. My current team has only three testers, and we all have other duties as well, such as customer support. We have multiple code bases, products and platforms to test, so we're always juggling priorities.

Making time

The product owner for our iOS product wanted us to experiment with automating some regression smoke tests through its UI. Another tester on the team and I decided we'd pair on a spike for this. However, we had so many competing priorities, we kept putting it off. As much as we try avoid multi-tasking, it seems there is always some urgent interruption.

Finally, we created a recurring daily meeting on the calendar, scheduled shortly after lunchtime when few other meetings were going on. As soon as we did that, we started making the time we needed. We might miss a day here or there, but we're much more disciplined about taking our time for the iOS test automation. As a result, we made steady, if slow, progress, and achieved many of our goals.

Scheduling help

Even though both of us were new to iOS, pairing gave us courage, and two heads were better than one. We'd still get stuck, though, and we needed the expertise of programmers and testers with experience automating iOS tests. Simply adding a meeting to the calendar with the right people has gotten us the help we needed. Even busy people can spare 30 minutes or an hour out of their week. Our iOS team is in another city two time zones away. If we put a meeting on the calendar with a Zoom meeting link, we can easily make contact at the appointed time. Screensharing enables us to make progress quickly, so we can stick to short time boxes.

Another way we ensured time on our schedule for the automation work was to add stories for it to our backlog. For example, we had stories for specific scripts, starting with writing an automated test for logging into the iOS app. Once we had some working test scripts, we added infrastructure-type chores, for example, get the tests running from the command line so we can add them to the team's continuous integration later. These stories make our efforts more visible. As a result, team members anticipate our meeting invitations and think of ideas to overcome challenges with the automation.

Time for testing

Putting time on the calendar works when I need to pair with a programmer to understand a story better, or when we need help with exploratory testing for a major new feature. I can always ask for help at the morning standup, but if we don't set a specific time, it's easy for everyone to get involved with other priorities and

forget.

The calendar is your friend. Once you create a meeting, you might still need to negotiate what time works for everyone involved, but you've started the collaboration process. Of course, if it's easy to simply walk over to someone's desk to ask a question, or pick up the phone if they're not co-located, do that. But if your team is like ours, where programmers and other roles pair full time, and there's always a lot going on, a scheduled meeting helps everyone plan the time for it.

If you have a tough project ahead, find a pair and set up a recurring meeting to work together. If you need one-off help, add a time-boxed meeting for today or tomorrow. If you need the whole team to brainstorm about some testing issues, schedule a meeting for the time of day with the fewest distractions. And if you haven't seen an old friend in too long, schedule a date for that, too!

[Download my PDF](#)

Posted: January 13, 2015

Tagged: product backlog, meetings, testing, planning, schedule

About the Author

Lisa Crispin is the co-author, with Janet Gregory, of *More Agile Testing: Learning Journeys for the Whole Team* (Addison-Wesley 2014), *Agile Testing: A Practical Guide for Testers and Agile Teams* (Addison-Wesley, 2009), co-author with Tip House of *Extreme Testing* (Addison-Wesley, 2002), and a contributor to *Experiences of Test Automation* by Dorothy Graham and Mark Fewster (Addison-Wesley, 2011) and *Beautiful Testing* (O'Reilly, 2009). Lisa was voted by her peers as the Most Influential Agile Testing Professional Person in 2012. Lisa enjoys working as a tester with an awesome agile team. She shares her experiences via writing, presenting, teaching and participating in agile testing communities around the world. For more about Lisa's work, visit www.lisacrispin.com, www.agiletester.ca, and follow [@lisacrispin](https://twitter.com/lisacrispin) on Twitter.

You may also be interested in:

What Happens When During a Sprint

Succeeding with Scrum is easier when you know when and why to conduct each of the Scrum events ...

Jan 03, 2023 • 34 Comments [Read](#)

What Are Agile Story Points?

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

Epics, Features and User Stories

I've been getting more and more emails lately from people confused about the difference between ...

Nov 08, 2022 • 93 Comments [Read](#)

Agile Mentors Podcast: Recap of Opening Series on Scrum

Recapping our agile podcast's initial launch series of topics

Sep 27, 2022 [Read](#)

For Better Agile Planning, Be Collaborative

The best plans are created by developers and stakeholders working together.

Jul 12, 2022 [Read](#)

Relationship between Definition of Done and Conditions of Satisfaction

I'd like to clarify the relationship between two important concepts: a team's Definition of Done ...

Jun 28, 2022 • 38 Comments [Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

by

Mike **Tagged:**

Cohn product backlog, sprint planning, velocity, planning, commitment,
36 capacity driven

Comments

In the prior two blog posts, I've described two alternative approaches to sprint planning:

[Velocity-Driven Sprint Planning](#)

[Capacity-Driven Sprint Planning](#)

In this post, I want to address why I prefer capacity-driven sprint planning. First, though, here is a very brief refresher on each of the approaches.

Brief Summary of Sprint-Planning Approaches

In velocity-driven sprint planning, a team selects a set of product backlog items whose high-level estimates (usually in story points but possibly in ideal days) equals their average velocity.

In capacity-driven sprint planning, a team selects one product backlog item at a time by roughly identifying and estimating the tasks that will be involved, and stopping when they feel the sprint is full.

Velocity is Variable

To begin to see why I prefer a capacity-driven approach to sprint planning, consider the graph below which

shows the velocities of a team over the course of nine sprints. The first thing you should notice is that velocity is not stable. It bounces around from sprint to sprint.

The first big drawback to velocity-driven planning is that velocity is too variable to be reliable for short-term (i.e., sprint) planning.

My experience is that velocity bounces around in about a plus or minus 20% range. This means that a team whose true velocity is 20 could easily experience a velocity of 16 this sprint and 24 next sprint, and have that just be random variation.

When a team that averages 20 sometimes completes 24 units of work, but sometimes completes only 16, we might be tempted to say they are accomplishing more or less work in those sprints.

And that is undoubtedly part of it for many teams. However, some part of the variation is attributable to the imprecision of the units used to estimate the product backlog items.

For example, most teams who estimate in story points use a subset of possible estimates such as the Fibonacci sequence of 1, 2, 3, 5, 8, 13 or a simple doubling (1, 2, 4, 8, 16). When a team using the Fibonacci sequence takes credit for finishing a 5-point story, they may have really only finished a 4-point story. This tends to lead to a slight overstating of velocity.

In the long-term this won't be a problem as the law of big numbers can kick in and things will average out. In the short term, though, it can create problems.

Anchoring

A second problem with velocity-driven sprint planning is due to the effect of anchoring. Anchoring is the tendency for an estimate to be unduly influenced by earlier information.

A good example of anchoring is the Christmas season. Suppose you go into a store and see a jacket you like. There's a sign saying \$400 but that price is crossed out and a new price of \$200 is shown. Your brain instantly thinks, "Awesome! A \$400 jacket for only \$200!" And, of course, this is why the store shows you the original price.

Who cares what that jacket used to sell for? It's \$200 today. That should be all that matters in your buy-or-not decision. However, once that \$400 price is in your brain, it's hard to ignore it. It anchors you into thinking you're getting a good deal when the jacket is \$200.

I won't go into the details here, but the best paper I've read on anchoring is from Magne Jørgensen and Stein Grimstad and is called "[The Impact of Irrelevant and Misleading Information on Software Development Effort Estimates](#)."

Anchoring and Velocity-Driven Planning

But what does anchoring have to do with sprint planning? Consider a team in a velocity-driven sprint planning meeting.

They've selected a set of stories equal to their average velocity. They then ask themselves if that set of stories feels like the right amount of work. (As described, in the post on velocity-driven sprint planning they may also identify tasks and estimate those tasks as an aid in answering that.)

A team doing velocity-driven sprint planning is predisposed to say, "Yes, we can do this," even if they can't. They are anchored by knowing that the selected amount of work is the same as their average velocity.

It's like me showing you that jacket with the \$400 sign next to it and asking, "How much do you think this jacket sells for?" Even if you don't say exactly \$400, that \$400 is in your head and will anchor you.

So, because of anchoring, a team with an average velocity of 20 is inclined to say the sprint is appropriately filled with 20 points – even if that work is really more like 16 or 24 units of work.

This will lead to teams sometimes doing less than they could have. It could also lead to teams sometimes doing more. But in those cases, my experience is that teams will be more likely to drop a product backlog item or two. Experience definitely tells me that teams are more likely to drop than to add.

Velocity Is Great for Longer-Term Planning

By this point, you may be thinking I'm pretty opposed to velocity-driven planning. That's only half true. Although I'm not a fan of using velocity to plan a single sprint, I am quite likely the world's biggest fan of using velocity to plan for the longer term. I've certainly written more about it than anyone I know.

The problem with velocity-driven sprint planning is that velocity is simply too variable to be useful in the short term. To illustrate why, suppose you get a job working at a car wash. On your first morning, your boss announces that you have a quota: You need to wash four cars per hour.

At the end of the first hour, though, you've only washed three cars. Should you be fired? Of course not. It was only one hour and perhaps you washed three large cars. Or perhaps it was overcast and only three drivers brought cars in to be washed.

What about at the end of your first day, an 8-hour shift? By then you should have washed 32 cars. But you've only washed 30. Should you be fired now? Again, almost certainly not.

What about the end of your first month? Figuring 20 days and 32 cars per day means you should have washed 640 cars. Suppose, though, you only washed 600. (That's the same percentage as washing 30 instead of 32 in a day.) Should your boss fire you now? Perhaps still not, but it's starting to be clear that you are not making quota.

What if you're off by the same percentage at the end of the year? If that quota was well chosen—and all other employees are meeting it—your boss at some point should consider letting you go (or dealing with your below expected productivity in some way, but this post isn't about good ways of dealing with productivity issues).

That quota is useful in the same way velocity is useful: over the long term. Think of how poorly run we'd consider that car wash if an employee was fired in the first hour of missing quota. The longest tenured employee would have been on the job for three days. So while that quota may be useful when measured over a month, it is not useful when measured hourly.

Velocity is useful in the long term, not the short term. A team with 30 product backlog items to deliver can sum the estimates (likely in story points) on those items and forecast when they'll be done. A team with three product backlog items to deliver would be better off doing good ol' task decomposition on those three items rather than relying on velocity.

So, Now What?

If you are already doing velocity-driven sprint planning and it's working for you, don't switch. However, if your team is new to Scrum or if your team is experiencing some of the issues I've described here, then I recommend using capacity-driven sprint planning.

Please let me know what you think in the comments below.

[Download my PDF](#)

Posted: November 4, 2014

Tagged: product backlog, sprint planning, velocity, planning, commitment, capacity driven

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

Why I Don't Emphasize Sprint Goals

Sprint goals are considered a mandatory part of Scrum. Here's why I disagree.

Mar 21, 2023

[Read](#)

What Are Agile Story Points?

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

Epics, Features and User Stories

I've been getting more and more emails lately from people confused about the difference between ...

Nov 08, 2022 • 93 Comments [Read](#)

For Better Agile Planning, Be Collaborative

The best plans are created by developers and stakeholders working together.

Jul 12, 2022 [Read](#)

Relationship between Definition of Done and Conditions of Satisfaction

I'd like to clarify the relationship between two important concepts: a team's Definition of Done ...

Jun 28, 2022 • 38 Comments [Read](#)

3 Ways to Help Agile Teams Plan Despite Uncertainty

We might not like ambiguity, but it's a fact of life. Find out how to plan with uncertainty in mind.

Jun 21, 2022 [Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile

New to Agile and Scrum?

Scrum

User Stories

Planning Poker

Agile Software Development

Agile Project Management

Agile Presentations

Mountain Goat on YouTube

Agile Mentors Podcast

Elements of Agile

Assessment

Agile & Scrum Training

Online Certifications

Video Courses

In-Person Certifications

On-Site Courses

Training Schedule

Compare Courses

PDUs and SEUs

More...

About Us

Contact Us

Our Students

Blog

Books by Mike Cohn

Agile Tools

by

Mike **Tagged:**

Cohn product backlog, sprint planning, velocity, sprint backlog, tasks,
18 estimates

Comments

There are two general approaches to planning sprints: velocity-driven planning and capacity-driven planning. Over the next three posts, I will describe each approach, and make my case for the one I prefer.

Let's begin with velocity-driven sprint planning because it's the easiest to describe. Velocity-driven sprint planning is based on the premise that the amount of work a team will do in the coming sprint is roughly equal to what they've done in prior sprints. This is a very valid premise.

This does assume, of course, things such as a constant team size, the team is working on similar work from sprint to sprint, consistent sprint lengths, and so on. Each of these assumptions is generally valid and violations of the assumptions are easily identified—that is, when a sprint changes from 10 to nine working days as in the case of a holiday, the team knows this in advance.

The steps in velocity-driven sprint planning are as follows:

1. Determine the team's historical average velocity.
2. Select a number of product backlog items equal to that velocity.

Some teams stop there. Others include the additional step of:

3. Identify the tasks involved in the selected user stories and see if it feels like the right amount of work.

And some teams will go even further and:

4. Estimate the tasks and see if the sum of the work is in line with past sprints.

Let's look in more detail at each of these steps.

Step 1. Select the Team's Average Velocity

The first step in velocity-driven sprint planning is determining the team's average velocity. Some agile teams prefer to look only at their most recent velocity. Their logic is that the most recent sprint is the single best indicator of how much will be accomplished in the next sprint. While this is certainly valid, I prefer to look back over a longer period if the team has the data.

I prefer to take the average of the eight most recent sprints. I've found that to be fairly predictive of the future. I certainly wouldn't take the average over the last 10 years. How fast a team was going during the Bush Administration is irrelevant to that team's velocity today. But, similarly, if you have the data, I don't think you should look back only one sprint.

Step 2: Select Product Backlog Items

Armed with an average velocity, team members select the top items from the product backlog. They grab items up to or equal to the average velocity.

At this point, velocity-driven planning in the strictest sense is finished. Rather than taking an hour or two per week of the sprint, sprint planning is done in a matter of seconds. As quickly as the team can add up to their average velocity, they're done.

Step 3: Identify Tasks (Optional)

Most teams who favor velocity-driven sprint planning realize that planning a sprint in a few seconds is probably insufficient. And so many will add in this third step of identifying the tasks to be performed.

To do this, team members discuss each of the product backlog items that have been selected for the sprint and attempt to identify the key steps necessary in delivering each product backlog item. Teams can't possibly think of everything so they instead make a good faith effort to think of all they can.

After having identified most of the tasks that will ultimately be necessary, team members look at the selected product backlog item and the tasks for each, and decide if the sprint seems full. They may conclude that the sprint is overfilled or insufficiently filled and add or drop product backlog items. At this point, some teams are done and they conclude sprint planning with a set of selected product backlog items and a list of the tasks to deliver them. Some teams proceed though to a final step:

Step 4: Estimate the Tasks (Optional)

With a nice list of tasks in front of them, some teams decide to go ahead and estimate those tasks in hours, as an aid in helping them decide if they've selected the right amount of work.

In next week's post, I'll describe an alternative approach, [capacity-driven sprint planning](#). And after that, I'll share some recommendations.

[Download my PDF](#)

Posted: October 21, 2014

Tagged: product backlog, sprint planning, velocity, sprint backlog, tasks, estimates

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

[Why I Don't Emphasize Sprint Goals](#)

Sprint goals are considered a mandatory part of Scrum. Here's why I disagree.

Mar 21, 2023

[Read](#)

[What Are Agile Story Points?](#)

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • [122 Comments](#)

[Read](#)

[Epic, Features and User Stories](#)

I've been getting more and more emails lately from people confused about the difference between ...

Nov 08, 2022 • [93 Comments](#)

[Read](#)

For Better Agile Planning, Be Collaborative

The best plans are created by developers and stakeholders working together.

Jul 12, 2022

[Read](#)

Relationship between Definition of Done and Conditions of Satisfaction

I'd like to clarify the relationship between two important concepts: a team's Definition of Done ...

Jun 28, 2022 • [38 Comments](#)

[Read](#)

3 Ways to Help Agile Teams Plan Despite Uncertainty

We might not like ambiguity, but it's a fact of life. Find out how to plan with uncertainty in mind.

Jun 21, 2022

[Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

Handling Requests for Unnecessary Artifacts

by

Mike Cohn

Tagged:
product backlog, product owner, agile manifesto, documentation,
artifacts, working software
Comments

"Working software over comprehensive documentation." You've certainly seen that statement on the Agile Manifesto. It is perhaps the most important of the Manifesto's four value statements—working software is, after all, the reason a team has undertaken a software development effort.

It is also one of the most misused parts of the Manifesto. This is the quote people cite when trying to get out of *all* documentation, which is not what the Manifesto says we should value.

Some documentation on a project can be great. But most non-agile teams write too much and talk too little. Some agile teams go to the opposite extreme, but many seem to find a good balance.

Occasionally, though, a team and product owner may disagree on the necessity of a document—usually with the product owner wanting a document and the team saying it's not necessary.

I've found two guidelines helpful in determining how to handle requests for various artifacts, especially documentation, on an agile project.

Guideline No. 1: If a team would produce an artifact while in the process of creating working software, that artifact is just naturally produced.

This guideline covers essentially everything a team would want to produce while on the way to building a system or product. It includes, for example, source code. It also includes any design documents, user guides and other items that the team wants to produce for the benefit of the current team, future teams maintaining the software or end users.

Guideline No. 2: If an artifact would not naturally be produced in the process of creating working software, the artifact is added to the product backlog.

The second guideline is there to cover cases when the product owner (or any other outside stakeholder) wants an artifact produced (usually a document) that the team would not normally produce.

For example, suppose the product owner asks the team to write a document describing every table and field in the database. I've certainly seen projects where such a document has been extremely helpful. (In fact, I've both requested and written such a document before.) But, I've always seen projects where this would have been unnecessary.

If the team thought this database description document were helpful, they would produce it in the process of creating the working software. And Guideline No. 1 would apply. But if they don't think this document is necessary, they won't produce it. Unless, that is, the product owner insists, which is where Guideline No. 2 comes in.

If the product owner wants this document, the product owner creates a new product backlog item saying so. The team can then estimate the time it will take to develop this document, just like they'd estimate any other product backlog item.

Putting an estimate on creating the document makes its cost explicit. This forces a product owner to think

about the opportunity cost of developing that document. The product owner will be able to ponder: *This five-point document or five points worth of new features?*

I don't know which the product owner will choose, but this approach makes the cost of that artifact explicit, allowing it to be compared with the value of additional features instead.

I'd love to know your thoughts on this. How does your team handle product owner requests for artifacts the team wouldn't naturally produce? What artifacts does your team find helpful? Please share your thoughts in the discussion below.

Get Your Free Scrum Cheat Sheet

Sign up below to receive this FREE reference.

First Name

Email Address

[Privacy Policy](#)

Posted: September 30, 2014

Tagged: product backlog, product owner, agile manifesto, documentation, artifacts, working software

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and

techniques to build extremely high-performance teams. He is the author of User Stories Applied for Agile Software Development, Agile Estimating and Planning, and Succeeding with Agile as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

[Be a Great Product Owner: Six Things Teams and Scrum Masters Need](#)

Learn six ways effective product owners ensure their teams' success.

Jan 31, 2023 • 16 Comments [Read](#)

[What Are Agile Story Points?](#)

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

[Epics, Features and User Stories](#)

I've been getting more and more emails lately from people confused about the difference between ...

Nov 08, 2022 • 93 Comments [Read](#)

[Relationship between Definition of Done and Conditions of Satisfaction](#)

I'd like to clarify the relationship between two important concepts: a team's Definition of Done ...

Jun 28, 2022 • 38 Comments [Read](#)

[Four Reasons Agile Teams Estimate Product Backlog Items](#)

Estimating product backlog items provides benefits beyond predicting when a project will be finished.

May 17, 2022 [Read](#)

[The Product Owner's Second Team](#)

Successful product owners will see their stakeholders as a team, not merely a set of individuals.

Nov 02, 2021 [Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile **Agile & Scrum Training** **More...**

New to Agile and Scrum? Online Certifications About Us

Scrum Video Courses Contact Us

User Stories In-Person Certifications Our Students

Planning Poker On-Site Courses Blog

Agile Software Development Training Schedule Books by Mike Cohn

Development Compare Courses Agile Tools

Agile Project Management PDUs and SEUs

Agile Presentations

Mountain Goat on YouTube

Agile Mentors Podcast

Elements of Agile

Assessment

Critiquing One of My Own Real User Stories

by
Mike Cohn
Tagged:
product backlog, user stories, teams, user roles, feedback, story writing
35 Comments

In case you haven't noticed, a few months ago we launched [Front Row Agile](#), a site dedicated to video training courses on agile and Scrum. This has placed me in the role of product owner for the site. And recently, the team on the project criticized my story writing! It was a valid criticism, so I want to share it here.

As you'd expect on a video training site, someone who buys a course is asked to review the course after watching it. Unfortunately very few course participants were leaving reviews. In fact, we'd only had one review given.

I looked into why this was so and discovered that when a course was complete, our on-screen request for a

review was hard to notice. See the following image to see what I mean.

After seeing that, I added a user story to our product backlog:

[As a participant who has just finished a course, I'd like a more prominent call to action asking me for a review.](#)

And this was the story the team member didn't like. He told me that this story made total sense for the participant taking action, but he said he would have written this:

[As Front Row Agile, I want participants to be encouraged to review a course after finishing the course, so we can have more feedback about how folks like our courses.](#)

The developer was right—there were a couple of possible problems with the story as I wrote it.

Jumping Right into How

First, the story I wrote bypassed what was needed and jumped right into how we would solve the problem. In general, product backlog items should start with the goal of the change being made to the system. The goal here was to get more reviews.

The goal was not a more prominent call to action. A more prominent call to action was how I thought we could achieve what we wanted.

However, I was very aware of this when I wrote the story and deliberately wrote the story to say how I wanted a change made.

I had already spoken to the designer about the lack of reviews, and he and I agreed that the call to action asking for a review needed to be more prominent. It would have been wrong of me to say I was open to any solution that achieved a goal of more reviews when I, as the product owner, had already decided on what I wanted.

I wrote this story as I did so that it would remind the designer of what we'd already discussed. If I'd written it

in some more vague way, the designer would have wondered, "Is the story Mike wanted about improving the call to action to get reviews?"

Minimally, though, I should have added an explicit "so that" clause to the story:

As a participant who has just finished a course, I'd like a more prominent call to action asking me for a review, so that there are more reviews on the site.

Writing for the Right User

The developer also criticized my story for being about the course participant rather than about Front Row Agile itself or perhaps me as the site's owner. That is, rather than start with "As a participant..." perhaps I should have written, "As Front Row Agile ..."

I try to write stories that put the reader in the most useful state of mind for understanding the story. Generally, that means I'll write a story about the actor in the story. In this case, Front Row Agile isn't doing anything; the course participant is. So, I wrote the story from that perspective.

So Which Story Is Better?

This leaves us with the question of which story is better?

As a participant who has just finished a course, I'd like a more prominent call to action asking me for a review.

As Front Row Agile, I want participants to be encouraged to review a course after finishing the course, so we can have more feedback about how folks like our courses.

I'm going to fall back on the old standby answer of: "It depends."

The "As a participant" story is better in the case where a product owner and team have discussed what is needed, settled on a solution, and the story is just reminding everyone involved of what has already been decided.

The "As Front Row Agile" story is better when the product owner is open to any number of possible solutions to solving a problem. If the Front Row Agile call to action asking for reviews had already been more prominent, I would prefer the second story above.

This would have shown that I was open to things like simply rewording the text, perhaps sending an email to participants after they finish a course, or perhaps moving the call to action to another screen entirely.

There are very few hard and fast rules in writing user stories. Even things like "write a story from the perspective of the person who benefits from the story" can best be thought of as guidelines.

What do you think? Do you have any similar examples?

Get 200 Real Life User Stories

Examples Written by Mike Cohn

See user stories Mike Cohn wrote as part of several real product backlogs.

First Name

Email Address

[Privacy Policy](#)

Posted: September 23, 2014

Tagged: product backlog, user stories, teams, user roles, feedback, story writing

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

User Stories: How to Create Story Maps

Story maps help to create a shared understanding of the product, visualize user needs, and elicit ...



Mar 14, 2023

[Read](#)

How to Run a Successful User Story Writing Workshop

What you need to know to conduct a story-writing workshop with your team.

Feb 28, 2023 • 3 Comments [Read](#)

What Is Cross-Functional Collaboration in Agile?

Perhaps the most prevalent and persistent myth in agile is that a cross-functional team is one on ...

Feb 14, 2023 • 49 Comments [Read](#)

What Are Agile Story Points?

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

Don't Equate Story Points to Hours

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • 39 Comments [Read](#)

Epics, Features and User Stories

I've been getting more and more emails lately from people confused about the difference between ...

Nov 08, 2022 • 93 Comments [Re](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile

New to Agile and Scrum?

Scrum

User Stories

Planning Poker

Agile Software Development

Agile Project Management

Agile Presentations

Mountain Goat on YouTube

Agile Mentors Podcast

Elements of Agile

Assessment

Agile & Scrum Training

Online Certifications

Video Courses

In-Person Certifications

On-Site Courses

Training Schedule

Compare Courses

PDUs and SEUs

More...

About Us

Contact Us

Our Students

Blog

Books by Mike Cohn

Agile Tools

Now vs. Not-Now Prioritization Along with Medium-Term Goals

by
Mike Cohn Tagged:
product backlog, user stories, product owner, sprint planning,
prioritizing
[Comments](#)

In last month's newsletter I wrote about how we make personal financial decisions in a now vs. not-now manner. We don't map out must-haves, should-haves, could-haves, and won't haves. And I promised in this month's newsletter, I would cover a simple approach to now vs. not-now planning while still accommodating working toward a bigger vision for a product.

I always recommend having a medium-term vision for where a product is headed. I find a three-month horizon works well. At the

start of each quarter, a product owner should put a stake in the ground saying "Here's where we want to be in three months." This is done in conjunction with the team and other stakeholders, but the ultimate vision for a product is up to the product owner.

A product owner doesn't need to be overly committed to the vision—it can be changed. But, without a stake in the ground a few months out, prioritization decisions are likely to be driven by whatever emergencies erupt right before sprint planning meetings. Without a vision, the urgent always wins over the strategic.

For choosing between competing ideas for a medium-term vision, I like using a formal approach—that is, something I can explain to someone else. I want to be able to say, "I chose to focus on such-and-such rather than this-and-that" and then show some analysis indicating how I made that decision. I've written elsewhere about relative weighting, theme screening and theme scoring—and we have [tools on this website for performing those analyses](#).

But at the start of each sprint, a product owner needs to make the smaller now vs. not-now decisions of prioritizing user stories to be worked on in the next one sprint. Rather than using a formal, explainable approach for that, I advise product owners consider four things about the product backlog items they are evaluating:

1. how valuable the feature will be
2. the learning that will occur by developing the feature
3. the riskiness of the feature
4. any change in the relative cost of the feature

I do this by first sorting the candidate product backlog items on attribute one, the value of each feature. There's nothing special about this; it's how product people have prioritized features for years.

But I don't stop there. I use items two through four to adjust the now sorted backlog. For the most part, I will move product backlog items up rather than down based on the influence of these additional factors. Let me explain what each factor is about.

The second factor refers to how much learning will occur by developing the feature. Learning can take many forms—for example, a team might learn about the technology being used ("this vendor's library isn't as easy as we were told it would be") or a product owner might learn how well users respond to a new user interface. If the learning that will result from developing a particular product backlog item is significant, I will move that item up the product backlog so it is developed in the coming sprint.

As for riskiness, if a given risk cannot be avoided, I prefer doing that product backlog item sooner rather than later so that I can learn the impact of the risk. And so, I will move the product backlog item up into the current sprint. If, however, there is a chance of avoiding a risk entirely (perhaps by not doing the feature at all), I will

move that product backlog item out of the current sprint. I'll then hopefully continue to do that in each subsequent sprint, thereby avoiding that risk entirely.

Finally, some features can be cheap if they are done now or expensive if they are put off. When I see such an item on a product backlog, I will move it into the current sprint to avoid the higher cost of delaying the feature.

By combining the use of these four factors when selecting items for a sprint with a formal approach to establishing a medium-term, three-month vision, you'll be able to successfully prioritize in a now vs. not-now manner within the context of achieving a bigger goal.

Get Your Free Scrum Cheat Sheet

Sign up below to receive this FREE reference.

First Name

Email Address

[Privacy Policy](#)

Posted: August 19, 2014

Tagged: product backlog, user stories, product owner, sprint planning, prioritizing

About the Author

Mike Cohn specializes in helping companies adopt

and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of User Stories Applied for Agile Software Development, Agile Estimating and Planning, and Succeeding with Agile as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

Why I Don't Emphasize Sprint Goals

Sprint goals are considered a mandatory part of Scrum. Here's why I disagree.

Mar 21, 2023

[Read](#)

User Stories: How to Create Story Maps

Story maps help to create a shared understanding of the product, visualize user needs, and elicit ...

Mar 14, 2023

[Read](#)

How to Run a Successful User Story Writing Workshop

What you need to know to conduct a story-writing workshop with your team.

Feb 28, 2023 • 3 Comments [Read](#)

Be a Great Product Owner: Six Things Teams and Scrum Masters Need

Learn six ways effective product owners ensure their teams' success.

Jan 31, 2023 • 16 Comments [Read](#)

What Are Agile Story Points?

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

Epics, Features and User Stories

I've been getting more and more emails lately from people confused about the difference between ...

Nov 08, 2022 • 93 Comments [Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile **Agile & Scrum Training** **More...**

New to Agile and Scrum? Online Certifications About Us

Scrum Video Courses Contact Us

User Stories In-Person Certifications Our Students

Planning Poker On-Site Courses Blog

Agile Software Development Training Schedule Books by Mike Cohn

Development Compare Courses Agile Tools

Agile Project Management PDUs and SEUs

Agile Presentations

Mountain Goat on YouTube

Agile Mentors Podcast

Elements of Agile

Assessment

Simplify Prioritization into “Now” and “Not Now”

by

Mike

Cohn

6

Comments

Tagged:

product backlog, product owner, sprinting, customers, prioritizing

I think I'd like to buy a big-screen plasma television. And maybe after that, a new amplifier for my home theater.

I've also noticed that our home dishwasher doesn't dry as well as it used to. For some reason, this doesn't bother my wife, Laura, but it annoys me. So I'd like to replace the dishwasher.

The clothes washer is doing fine, but the dryer isn't working too well. I think we've had it 14 years, and I sometimes have to run the drying cycle three times before clothes are fully dried. So, I'll probably want to buy a clothes dryer, too. I've also lately become irritated by the toaster. It's great and works well, but bread doesn't fit. I have to angle the bread, with part sticking out of the toaster – so, I get partially toasted bread. Or, I flip the bread midway and then get the middle of the bread overly toasted. So, add a new toaster to the list.

After all that, a new car might be nice.

What is this list? Well, my personal *Things to Buy Backlog*, of course. It's a prioritized sequenced list of things I plan to buy. To facilitate long-term planning it includes everything I plan to buy for the next 12 months:

1. Groceries
2. Tickets to movies that haven't been released yet
3. A magazine that I'll buy in Newark airport next October because I'll find a cover story intriguing
4. A new pair of shoes
5. Big-screen plasma TV
6. Home theater amplifier
7. Dishwasher
8. Clothes dryer
9. Toaster
10. Car

Don't you have a list like this, with that level of detail? Of course not. Neither do I. We don't make lists like this because such a list is unnecessary. In everyday life, we don't map out the equivalent of a product backlog.

Instead, we make our purchase decisions in a much simpler fashion: “Now” or “not now.” Do I want that big television now, or not now? And this amount of prioritization is good enough.

In many cases, it is also good enough for our product development projects.

Sure, you'll always need to know what you're going to build this sprint. But what you build next sprint will likely depend on what the team delivers *this* sprint, and on how customers respond to it.

That means you might want to forgo prioritizing any further ahead unless you are in an environment that requires it, such as contract development.

Prioritizing features in a “now/not now” manner streamlines your planning process, allows learning to be incorporated into the planning process, and better matches how we prioritize outside the workplace.

Many projects will find this a very valuable shift.

In this [post](#) I write about how to do this within the context of establishing a longer-term vision for a product.

[Download my PDF](#)

Posted: July 15, 2014

Tagged: product backlog, product owner, sprinting, customers, prioritizing

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of User Stories Applied for Agile Software Development, Agile Estimating and Planning, and Succeeding with Agile as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

Be a Great Product Owner: Six Things Teams and Scrum Masters Need

Learn six ways effective product owners ensure their teams' success.

Jan 31, 2023 • 16 Comments [Read](#)

What Happens When During a Sprint

Succeeding with Scrum is easier when you know when and why to conduct each of the Scrum events ...

Jan 03, 2023 • 34 Comments [Read](#)

What Are Agile Story Points?

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

Epics, Features and User Stories

I've been getting more and more emails lately from people confused about the difference between ...

Nov 08, 2022 • 93 Comments [Read](#)

Relationship between Definition of Done and Conditions of Satisfaction

I'd like to clarify the relationship between two important concepts: a team's Definition of Done ...

Jun 28, 2022 • 38 Comments [Read](#)

Four Reasons Agile Teams Estimate Product Backlog Item

Estimating product backlog items provides benefits beyond predicting when a project will be finished.

May 17, 2022 [Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

Choose Backlog Items That Serve Two Purposes

by
Mike Cohn
Tagged:
product backlog, user stories, product owner, prioritizing
11 Comments

I've been playing a fair amount of Go lately. If you're not familiar with Go, it's a strategy game that originated in China 2,500 years ago. It's along the lines of chess, but it's about marking territory with black and white pieces played on the intersections of a grid of 19x19 lines.

Like Scrum, there are very few rules in Go. Also like Scrum, there are a fair number of principles, often called proverbs in the Go world. One of these Go principles is that a move that does two things is better than a move that does one. For example, a

single move may defend a group of the player's stones while threatening the opponent's stones.

Even if you don't know Go, I think this proverb is understandable--after all, something that does two things sounds like it would be better than something that achieves only one goal. But this isn't hard-and-fast advice. It's a principle or proverb because sometimes a move that does one thing--but does that thing very well--will be the better choice.

In addition to playing more Go over the past week or two, I've also spent a lot of time thinking about approaches to prioritizing product backlog items. I hope to share some new thoughts on that here in the coming months.

One thing that has become increasingly clear is that when prioritizing product backlog items, an item that can achieve two goals should often be higher priority.

Let's see how a product backlog item can help achieve two goals.

Normally, product backlog items are prioritized based on how desirable the new functionality is. This is often, of course, adjusted by how long that functionality will take to develop. That is, a product owner wants something pretty badly until finding out that feature will take the entire next quarter. So, desirability often tempered by cost (usually in development team time) determines priorities.

But there can be other important considerations. And prioritizing highly a product backlog item that is moderately desirable (given its cost) but that also achieves one of these secondary goals may make that item a better first choice overall.

One such consideration is how much learning will occur if the product backlog item is developed. If the team or product owner is going to learn from developing a feature, do it sooner.

Learning can occur in a variety of ways. Developers may learn from doing a product backlog item that the new technology they'd counted on being easy isn't. A product owner may learn by showing a new user interface built for a specific product backlog item is not something users are excited about as the product owner thought.

Another consideration is how much risk will be reduced or eliminated by developing a product backlog item. If a certain product backlog item needs to be developed and doing so will be risky, my preference is to do that product backlog item early so that I have time to recover from the risk if it hits.

Selecting product backlog items to work on that achieve two (or all three!) of these goals can be a very powerful prioritization strategy. Adding value while simultaneously reducing risk or accelerating learning is just as good as playing a Go stone that achieves two goals.

Get 200 Real Life User Stories Examples Written by Mike Cohn

See user stories Mike Cohn wrote as part of several real product backlogs.

First Name

Email Address

[Privacy Policy](#)

Posted: July 8, 2014

Tagged: product backlog, user stories, product owner, prioritizing

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

User Stories: How to Create Story Maps

Story maps help to create a shared understanding of the product, visualize user needs, and elicit ...

Mar 14, 2023

[Read](#)

How to Run a Successful User Story Writing Workshop

What you need to know to conduct a story-writing workshop with your team.

Feb 28, 2023 • 3 Comments [Read](#)

Be a Great Product Owner: Six Things Teams and Scrum Masters Need

Learn six ways effective product owners ensure their teams' success.

Jan 31, 2023 • 16 Comments [Read](#)

What Are Agile Story Points?

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

Epics, Features and User Stories

I've been getting more and more emails lately from people confused about the difference between ...

Nov 08, 2022 • 93 Comments [Read](#)

Advantages of User Stories over Requirements and Use Cases

At the surface, user stories appear to have much in common with use cases and traditional ...

Oct 05, 2022 • 41 Comments [Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile **Agile & Scrum Training** **More...**

New to Agile and Scrum? Online Certifications About Us

Scrum Video Courses Contact Us

User Stories In-Person Certifications Our Students

Planning Poker On-Site Courses Blog

Agile Software Development Training Schedule Books by Mike Cohn

Development Compare Courses Agile Tools

Agile Project Management PDUs and SEUs

Agile Presentations

Mountain Goat on YouTube

Agile Mentors Podcast

Elements of Agile

Assessment

Adding Decorated User Roles to Your User Stories

by
Mike Cohn 16
Tagged:
product backlog, user stories, user roles
[Comments](#)

When writing user stories there are, of course, two parts: user and story. The user is embedded right at the front of stories when using the most common form of writing user stories:

As a <user role> I <want/can/am required to> <some goal> so that <some reason>.

Many of the user roles in a system can be considered *first class users*. These are the roles that are significant to the success of a system and will occur over and over in the user stories that make up a system's

product backlog.

These roles can often be displayed in a hierarchy for convenience. An example for a website that offers benefits to registered members is as follows:

In this example, some stories will be written with “As a site visitor, …” This would be appropriate for something anyone visiting the site can do, such as view a license agreement.

Other stories could be written specifically for registered members. For example, a website might have, “As a registered member, I can update my payment information.” This is appropriate to write for a registered member because it’s not appropriate for other roles such as visitors, former members, or trial members.

Stories can also be written for premium members (who are the only ones who can perhaps view certain content) or trial members (who are occasionally prompted to join).

But, beyond the first class user roles shown in a hierarchy like this, there are also other occasional users—that is, users who may be important but who aren’t really an entirely separate type of user.

First-time users can be considered an example. A first-time user is rarely really a first-class role but they can be important to the success of a product, such as the website in this example.

Usually, the best way to handle this type of user role is by adding an adjective in front of the user role. That could give us roles such as:

First-time visitor
First-time member

The former would refer to someone on their absolute first visit to the website. The latter could refer to someone who is on the site for the first time as a subscribed member.

This role could have stories such as, “As a first-time visitor, additional prompts appear on the site to direct me toward the most common starting points so that I learn how to navigate the system.”

A similar example could be “forgetful member.” Forgetful members are very unlikely to be a primary role you identify as vital to the success of your product. We should, however, be aware of them.

A forgetful member may have stories such as “As a forgetful member, I can request a password reminder so that I can

log in without having to first call tech support.”

I refer to these as *decorated user roles*, borrowing the term from mathematics where decorated symbols such as a-bar (\bar{x}) and p-hat (\hat{x}) are common.

Decorated users add the ability to further refine user roles without adding complexity to the user role model itself through the inclusion of additional first-class roles. As a fan of user stories, I hope you find the idea as helpful as I do.

In the comments, let me know what you think and please share some examples of decorated user roles you've used or encountered.

Get 200 Real Life User Stories Examples Written by Mike Cohn

See user stories Mike Cohn wrote as part of several real product backlogs.

First Name

Email Address

[Privacy Policy](#)

Posted: July 1, 2014

Tagged: product backlog , user stories , user roles

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the *Better User Stories* video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

User Stories: How to Create Story Maps

Story maps help to create a shared understanding of the product, visualize user needs, and elicit ...

Mar 14, 2023

[Read](#)

How to Run a Successful User Story Writing Workshop

What you need to know to conduct a story-writing workshop with your team.

Feb 28, 2023 • 3 Comments [Read](#)

What Are Agile Story Points?

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

Epics, Features and User Stories

I've been getting more and more emails lately from people confused about the difference between ...

Nov 08, 2022 • 93 Comments [Read](#)

Advantages of User Stories over Requirements and Use Cases

At the surface, user stories appear to have much in common with use cases and traditional ...

Oct 05, 2022 • 41 Comments [Read](#)

Relationship between Definition of Done and Conditions of Satisfaction

I'd like to clarify the relationship between two important concepts: a team's Definition of Done ...

Jun 28, 2022 • 38 Comments [Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile Assessment		

4 Tips for Spring Cleaning Your Product Backlog

by

Mike Tagged:

Cohn product backlog, user stories, product owner, sprinting, meetings,

11 continuous improvement

Comments

It's May, and we're well into spring now. If you're like me, you haven't yet done your annual spring cleaning. But I'll do mine this week, if you promise to also do yours.

The type of spring cleaning I'm referring to is cleaning your product backlog. When not given sufficient attention, a product backlog can get messier than the shrubs I haven't trimmed the last three years. In this post, I want to share four things a product owner can do as part of spring cleaning the product backlog.

First, the product owner should go through the product backlog looking

for things to delete. Many product owners have a habit of adding to a product backlog but never removing from it features that are no longer wanted. If a feature is no longer wanted, it should be removed.

It's tempting to leave these low-value features in the product backlog under the premise that we'll get to them someday. If you feel you can't truly delete items, find a way to archive them somewhere.

If you're using software to manage your product backlog, it may have functionality for doing this. If you're using a spreadsheet, copy the unwanted items out of the main spreadsheet and into one named, "Long Term Product Backlog." That's what I do. And, if you're like me, you'll feel good having saved them, but you'll never look at them again.

My second spring cleaning tip is also about preventing the size of the product backlog from becoming unmanageable: Group related items. I've worked with product owners attempting to manage over 2,000 product backlog items. That's too many.

I find a good upper limit to be in the range of 100 to 150 items. When a product backlog contains more than that, it becomes difficult to work with the product backlog. It's harder to see relationships between items. It's harder to remember if product backlog items already exist for a certain thing, so new items are created, increasing the frequency of duplicates.

Grouping related items together into what is called a "theme" makes it easier to mentally work with those items. So much so that I could group as one item toward my upper limit of 100 to 150.

With low-priority items removed, and related items grouped into themes, it's time for my third tip for spring cleaning a product backlog: the Product Owner should check the priorities of the items on the product backlog.

An item's position in the product backlog is based on four factors:

- How valuable the feature is
- How much learning will occur by doing the feature
- How much risk is addressed by doing the feature
- Changes in relative cost

With the product backlog now prioritized, my final suggestion for spring cleaning is to make sure the top items are ready for the team to work on. Each of the top product backlog items should be sufficiently thought through so that the team can work on them in the next sprint with a reasonable expectation of being able to finish them.

This doesn't mean that each product backlog item is accompanied by a mini-spec detailing every issue or consideration around that item. It does mean, however, that the remaining open issues are ones that the team and product owner can quickly resolve during the sprint.

If delivering a top priority product backlog item requires three meetings of the "Stakeholder Steering Committee," that item isn't a good choice to be at the top of the product backlog.

So there you have it: Four tips for spring cleaning your product backlog. And, even though it's cleaning, this type of cleaning sure beats cleaning out the garage.

[Download my PDF](#)

Posted: June 17, 2014

Tagged: product backlog, user stories, product owner, sprinting, meetings, continuous improvement

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

User Stories: How to Create Story Maps

Story maps help to create a shared understanding of the product, visualize user needs, and elicit ...



Mar 14, 2023

[Read](#)

How to Run a Successful User Story Writing Workshop

What you need to know to conduct a story-writing workshop with your team.

Feb 28, 2023 • 3 Comments [Read](#)

Be a Great Product Owner: Six Things Teams and Scrum Masters Need

Learn six ways effective product owners ensure their teams' success.

Jan 31, 2023 • 16 Comments [Read](#)

What Happens When During a Sprint

Succeeding with Scrum is easier when you know when and why to conduct each of the Scrum events ...

Jan 03, 2023 • 34 Comments [Read](#)

What Are Agile Story Points?

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

Epics, Features and User Stories

I've been getting more and more emails lately from people confused about the difference between ...

Nov 08, 2022 • 93 Comments [Re](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

by
Roman **Tagged:**
Pichler product backlog, user stories, product owner, sprinting,
9 management, customers, lean
Comments

Many of you will know Roman Pichler as the author of [Agile Product Management with Scrum](#). For the last few years Roman has been working on various ideas to support envisioning and ideation. I've asked him to write a guest post here describing his Vision Board and how it connects to Scrum's product backlog. I'm sure you'll find his ideas very interesting.

—Mike

Vision and Backlog

Scrum is a great framework for building a product with the right features. It encourages the use of a vision shared by the product owner, the ScrumMaster, the development team, and the stakeholders, and it provides a product backlog that allows the team to move towards the vision by creating product increments.

Unfortunately, there is little guidance on how the backlog is derived from the vision. Once we have an idea for a new product or new features, how do we know which [user stories](#) we should include in the backlog? And how can we tell who should be at the sprint review meeting to help us understand if we are building the right product?

In the worst case, we could implement the wrong stories, get feedback from the wrong people, and build a product nobody wants and needs.

Four Questions

I have found it very valuable to answer the following four questions before stocking the product backlog:

- Who are the users and customers?
- Why would they use and buy the product?
- What makes the product special? What are its key features?
- What are the business goals the product should deliver and how are they met?

Without knowing who the users and customers are and why they would employ the product, it's impossible to write meaningful user stories. Similarly, we don't understand what should make the product special and stand out, it is difficult to make informed decisions about the product functionality, the user interaction, the UI design right, and even the key architecture and technology choices. And without being clear on the business goals and the product's business model, it will be hard to justify the investment and attract a budget.

Enter the Vision Board

The Vision Board is a simple yet effective tool that helps agile teams capture their vision and systematically answer the four questions above. The following picture shows the Vision Board, and I explain its sections below. You can download the tool for free at: romanpichler.com/tools/vision-board/

Vision

Statement is a concise summary of your idea that describes your intention and motivation.

Target Group describes the market or segment you want to address. You should state who the users and its customers are likely to be.

Needs describes the product's value proposition: the problems and pain points the product removes, and the benefits or gains it creates for the users and the customers. The section should make it clear why people will want to use and buy your product.

Product summarises the three to five features that make your product stand out. These are likely to correlate to its unique selling proposition, and they should address the needs identified.

Value explains **why** it's worthwhile for your company to invest in the product and **how** the business goals can be achieved. Sample goals are increase revenue, enter a new market, reduce cost, develop the brand, or gain a technological advantage.

Let's take a look at an example to make this more concrete.

A Sample Vision Board

Say we want to create a new game for children to help them enjoy more about music and dancing and learn about the topics. Then corresponding Vision Board could look like this:

The
board

above captures the overarching vision. It states children aged 8-12 as the intended users of the game and their parents as the customers. It lists three reasons why the kids would want to play the game, the key features of the product, and the business goal and how the business model could work.

The good news is the Vision Board above helped me to think through the market the value proposition and the business model of the new app. The bad news is it contains lots of untested assumptions that would make it a gamble to use the board to derive the product backlog. What's required instead is to systematically identify and validate its assumptions.

Validation with the Vision Board

Validating the Vision Board starts with identifying the most crucial assumption, the assumption that must be tested now to avoid late failure – finding out too late that the problem is not worthwhile solving. Once you have identified the appropriate assumption, decide which method is best to address it and who should be in your test group.

Some of my favourite techniques for testing Vision Board assumptions are direct observation – watching target users and customers get a job done, for instance, observing how children play music games on the computer today – and problem interviews – talking to members of the target group about how they accomplish a job today, for instance, what's good and bad about playing computer games, particularly those related to music and dancing.

Once you have run a test, collect the relevant data and analyse it. Then use the newly gained insights to change the board either radically or incrementally. An example of the latter is to adjust a need or to replace a key feature. But a radical change – called a “pivot” in Lean Startup – profoundly affects one or more sections of the Vision Board. The vision stays true but the path towards the vision changes. For instance, changing the product to an app that teaches children dancing by encouraging them to try out the moves themselves would be a pivot for the dance game.

From Vision Board to the Product Backlog

Once you have validated the key risks and critical assumptions on the Vision Board, use it to create the initial product backlog and discover the right stories, interactions, and UI design, as the following picture illustrates.

With your initial backlog and the Scrum team in place, you are ready to start sprinting.

Summary

Before writing user stories, coming up with user interface ideas or the user interaction, make sure you understand who the users and customers are, why they would use and buy your product. Find out what makes your product special, what your business goals are, and how you can meet them. As Joel A. Barker puts it: "Vision without action is merely a dream. Action without vision just passes the time. Vision with action can change the world." You can learn more about working with the Vision Board and download the template for free at: romanpichler.com/tools/vision-board/

Get Free User Stories Book Chapters

Please provide your name and email and we'll send you the sample chapters and we'll send a short weekly tip from Mike on how to succeed with agile.

[Get my chapters now!](#)

Posted: April 15, 2014

Tagged: product backlog, user stories, product owner, sprinting, management, customers, lean

About the Author

Roman Pichler is a leading agile product management and Scrum expert. He has more than 10 years experience in training and coaching product managers and product owners, and a long track record in helping companies apply agile practices to achieve business success. Roman is the author of "Agile Product Management with Scrum". He has created several product management tools, and he writes a popular blog on product ownership at: www.romanpichler.com/blog

You may also be interested in:

User Stories: How to Create Story Maps

Story maps help to create a shared understanding of the product, visualize user needs, and elicit ...

Mar 14, 2023

[Read](#)

[How to Run a Successful User Story Writing Workshop](#)

What you need to know to conduct a story-writing workshop with your team.

Feb 28, 2023 • [3 Comments](#)

[Read](#)

[Be a Great Product Owner: Six Things Teams and Scrum Masters Need](#)

Learn six ways effective product owners ensure their teams' success.

Jan 31, 2023 • [16 Comments](#)

[Read](#)

[What Happens When During a Sprint](#)

Succeeding with Scrum is easier when you know when and why to conduct each of the Scrum events ...

Jan 03, 2023 • [34 Comments](#)

[Read](#)

[What Are Agile Story Points?](#)

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • [122 Comments](#)

[Read](#)

Epics, Features and User Stories

I've been getting more and more emails lately from people confused about the difference between ...

Nov 08, 2022 • [93 Comments](#)

[Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

by

Mike Tagged:

Cohn product backlog, product owner, sprinting, velocity, defect

38 management, scrum master

Comments

When talking about it informally, I define velocity as simply a measure of how fast a team is going. And for most purposes, this definition works quite well. However, it creates confusion on some of the finer points of what should count in calculating a team's velocity. This confusion comes about because there are really two more precise ways of defining velocity. Let's see what they are.

- 1) Velocity measures how much functionality a team delivers in a sprint.
- 2) Velocity measures a team's ability to turn ideas into new functionality in a sprint.

Those may sound the same. They are subtly different. To see how, suppose you hop in a river and begin swimming. After an hour, you measure how far you've traveled, and you are 2 kilometers from where you began. In agile terms, we might want to call this your velocity and say you swim 2 kilometers per hour or per sprint, if a swimming sprint is an hour long.

But, what if the river had been flowing at the rate of one kilometer per hour against you while you swam? In that case, you really swam 3 kilometers. Measured against the banks of the river, you've only moved two kilometers of physical distance. But while going forward two kilometers, you overcame being pushed backward one kilometer by the river.

So, is your velocity two or three? If we use our first definition—velocity is how much a team delivers in a sprint—then velocity is two. This swimmer delivered 2 kilometers of progress.

If we use our second definition—velocity is the ability to turn ideas into new functionality—then velocity is three. This swimmer has the ability to deliver 3 kilometers of progress per sprint, and we'd see that he was swimming in a lake with no current instead of a river.

To see how this applies to an agile project, consider the issue of whether a team should earn velocity credit for fixing bugs during a sprint. A team that uses velocity to measure how much functionality is delivered in a sprint will not claim credit for bug fixes. No new functionality has been delivered. So no points are earned.

A team using defining velocity as the ability to turn ideas into functionality, on the other hand, will claim credit for bug fixes. Their logic is that the time spent on bug fixing could have been spent adding new functionality except the product owner prioritized different work for them.

For many teams, the two definitions will yield the same value. Values will differ most for teams doing work for which they are not taking velocity credit--usually teams doing things like a lot of bug fixing or doing large amounts of refactoring.

Neither of these subtle differences in the definition of velocity is always better than the other. The one you use should largely depend on what you hope to learn by measuring it and by your expectations about the future.

If you expect the future to be just like today—that is, the team will spend the same amount of time doing bug fixing, refactoring and the like as they do now, then using velocity as a measure of how much forward progress is made will be the right answer for you.

However, if you expect the future to be different—perhaps the large refactoring and time spent fixing bugs will soon be over—then you may want to define velocity as the team's ability to turn ideas into functionality, and would then add to velocity the points given to those activities.

The most important thing is to clarify with everyone on the team, including the product owner and ScrumMaster, is exactly what your team means when they use the term "velocity." Having a precise definition makes it very easy to answer questions that come up around what should be counted when measuring velocity.

[Download my PDF](#)

Posted: March 18, 2014

Tagged: product backlog, product owner, sprinting, velocity, defect management, scrum master

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile Assessment		

New Year's Resolutions for ScrumMasters and Product Owners

by
Mike Cohn **Tagged:**
product backlog, product owner, sprint planning, velocity,
continuous improvement, audio, scrum master
[Comments](#)

This article has an audio version: [Download Audio Version](#)

With the new year, it's time for some resolutions. I've got the same old ones (lose weight, eat better, get more sleep, help more old ladies across the street, stop calling every cat I see "Fajita," and such). But since I fail at those every year, I thought perhaps it would be better for all of us if we made some Scrum-related resolutions. And so here are a couple of

suggestions for both ScrumMasters and Product Owners. Each is a resolution I've made in the past during my time in each role.

For ScrumMasters

Let's start with two possible resolutions ScrumMasters may want to make:

1. **Always let team members speak before you do in meetings.** This was a hard one for me. I'm both opinionated and impatient. When an issue comes up during a meeting, it can be hard for me not to instantly blurt out my opinion. This often shuts down debate that might otherwise have occurred. I finally got over this problem (mostly) years ago when I resolved to always let two team members give their opinions before I (as a combination ScrumMaster / developer) would share my own.
2. **Praise the team more often.** I am very much a glass-is-half-empty type of guy. A team cuts their defect rate by 50 percent, and I want to know why not 100 percent. Velocity goes up by 5 points, and I think they could have gotten one more point. Part of this can be good; I am definitely always looking for ways to get better. And I put the same pressure on myself. But, I've learned that, of course, it can be depressing for a team that is making tremendous improvements to always hear about how there is still room to improve. Prevent this from becoming a problem by making it a priority to praise them more often.

For Product Owners

Here are a couple of resolutions Product Owners may want to make:

Redirect the team less often. It is extremely tempting to redirect the team every time you come up with a great new idea. Of course, you know how bad this can be as the team gets buffeted from one top priority to the next. Simply resolving to stop redirecting them, though, is unlikely to work. So here are two things you can do to help achieve this resolution:

1. Sit on changes for a day before introducing them to the team. Just commit to yourself that no matter how good or urgent an idea seems, you will hold off for one day before asking the team to work on it. Rarely is a change so critical that it must be started immediately. Stalling for even a day gives you time to reconsider. If the change seems as critical tomorrow, go ahead and interrupt the team.
2. Write it down somewhere. Often telling the team to work on something new is just a way for the Product Owner to get the item out of his or her head. You can also achieve this by making note of the desired change. If you're using a tool to manage your product backlog, add it so it'll be visible when you plan the next sprint. If you don't use a tool, note the item in a simple text file you can review before each sprint planning meeting.

Be more available. I've surveyed a few agile teams about what their Product Owners could do better; "Be more available" always ranks near the top of the results. If you're a Product Owner, resolve that 2014 will be the year in which you fully address this problem. Here are a couple of things you can do to make that happen:

1. Spend time in the team's area. If your desk is away from the team's shared workspace, do something like tell the team you will spend from 1 p.m. to 3 p.m. every day in their area. This time isn't for any specific meeting (although meetings can occur during this time). Rather, you just bring your laptop, find an empty desk or surface in their area, and do your normal work right there. If the team needs you, you're just a few steps away. When they don't, you just do your normal work but near them.
2. Share a secret code with the team. Establish a rule with the team that if they email you with something like "[Today]" in the subject line of an email, you will respond before going home for the day. (Or perhaps before going to bed at night, if working remotely from the team.) Discuss with the team what constitutes appropriate use of this. For example, they shouldn't email you 100 times per day with this secret code. (If they need to, see the item above about spending time in the team's work area. You've got a problem that can't be solved by email.)

Last, but Not Least

Resolve to Improve. Whatever you choose for 2014, resolve that by the end of the year, you and the team you're working with will be better than you are today. And it wouldn't hurt if you ate better and got more sleep this year, too.

In the comments below, please let me know what agile-related resolutions you've made for 2014.

[Download my PDF](#)

Posted: January 2, 2014

Tagged: product backlog , product owner , sprint planning , velocity , continuous improvement , audio , scrum master

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile **Agile & Scrum Training** **More...**

New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile Assessment		

How to Be Sure You've Thought of Everything

by
Mike Cohn
Tagged:
product backlog, product owner, customers, continuous improvement, audio
[Comments](#)

This article has an audio version: [Download Audio Version](#)

A common question I get is how can a product owner (or team) be sure they've thought of everything when writing a product backlog. The question is especially common from teams doing contract development where an upfront requirements specification is still expected. But I also get the question from teams in organizations that like to unnecessarily lock everything down way up front.

I've given this a great deal of thought and I've finally figured it out. Here's how you can be sure you've

thought of everything when writing a product backlog:

Wait.

Yes, wait until the end of the project. The only way to be sure you've thought of everything is to dive in, complete the project, get to the end and over time learn exactly what it was your customer wanted.

Every non-trivial project includes some emergent requirements. An emergent requirement is one that was not known in advance but that the customer figures out sometime during the project. No matter how hard a customer or product owner is asked to think, they will always come up with new requirements during the project. These result from looking at, and even better, touching, the system being built. Seeing what has been developed gives users new ideas. "Now that I've seen it, here are some new things I want," they tell us.

Customers and product owners should not use the existence of emergent requirements as an excuse for shoddy thinking. Development teams have the right to expect them to make reasonable efforts at knowing what they will eventually want. But, just as product owners and customers cannot possibly know everything in advance, development teams need to stop expecting them to.

Certainty is impossible. Avoid asking for or expecting it.

[Download my PDF](#)

Posted: November 19, 2013

Tagged: product backlog, product owner, customers, continuous improvement, audio

[About the Author](#)

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the *Better User Stories* video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

Be a Great Product Owner: Six Things Teams and Scrum Masters Need

Learn six ways effective product owners ensure their teams' success.

Jan 31, 2023 • 16 Comments [Read](#)

What Are Agile Story Points?

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

Epics, Features and User Stories

I've been getting more and more emails lately from people confused about the difference between ...

Nov 08, 2022 • 93 Comments [Read](#)

Elements of Agile: An Agile Assessment Tool for Iterative Improvements

New to agile, or needing an agile re-boot? We've got a new no-cost assessment and actionable ...

Sep 13, 2022 [Read](#)

Relationship between Definition of Done and Conditions of Satisfaction

I'd like to clarify the relationship between two important concepts: a team's Definition of Done ...

Jun 28, 2022 • 38 Comments [Read](#)

Four Reasons Agile Teams Estimate Product Backlog Item

Estimating product backlog items provides benefits beyond predicting when a project will be finished.

May 17, 2022 [Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile **Agile & Scrum Training** **More...**

New to Agile and Scrum? Online Certifications About Us

Scrum Video Courses Contact Us

User Stories In-Person Certifications Our Students

Planning Poker On-Site Courses Blog

Agile Software Development Training Schedule Books by Mike Cohn

Development Compare Courses Agile Tools

Agile Project Management PDUs and SEUs

Agile Presentations

Mountain Goat on YouTube

Agile Mentors Podcast

Elements of Agile

Assessment

Product Backlog Bankruptcy!

by
Ilan Goldstein Tagged:
product backlog, product owner, prioritizing
11 Comments

The following is a guest post by Ilan Goldstein, author of "Scrum Shortcuts without Cutting Corners: Agile Tactics, Tools, & Tips" which is full of advice on speeding up your adoption of Scrum. It's a great book I can highly recommend. You can [read my foreword](#) to it for more of my thoughts. Here are Ilan's thoughts on what to do with a large product backlog.

--Mike

You're sitting in class, really enjoying your agile training (that you've somehow made time to attend between all of the super-urgent tasks that keep popping up on your multitude of lists), when all of a sudden you hear words come out of the trainer's mouth that seem to make perfect sense yet also sound like crazy talk at the

same time...

Those words being: The Product Backlog is a single ordered list of **any** type work that needs doing (features, bugs, research requirements etc.) and only has:

ONE top priority item,
ONE priority two item,
ONE priority three item
So on so forth....

"What the? If we do that we'll have over [800] (insert your own massive number of choice here) items in this Backlog thingy and the chances of uniquely sequencing all of those is somewhere between zero and nil!".

I hear this type of concern all the time and for those reading this article feeling similarly hesitant, my message to you is to chill, as there is always an option for handling even the most gargantuan of Product Backlog mountains. That option: declare Backlog bankruptcy!

How does it work?

Our Product Owner puts together a short note (be it in email, skywriting or other) that goes a little something like this:

Hi All,

As I'm sure you're all well aware, the maintenance of our various, never-ending lists is becoming unmanageable. This is creating a major diversion from delivering the highest business value to you in the shortest time. These lists are overwhelming, distracting and frankly, many items on them are out-dated and no longer relevant. As such, we're taking the somewhat drastic approach of declaring 'Backlog bankruptcy'.

What this means for you:

1. *All items on all lists are being archived. Could be worse, at least we're not shift-deleting them...*
2. *Moving forward, we will be consolidating all requirements, be they features, bugs, research requirements (or other) into a single list that we will be calling our Product Backlog.*
3. *No doubt there will be some items on the now archived lists that should be resurrected and included within our new 'fresh-start' Product Backlog so please let me know about these. If an item is a feature or research requirement, please make it clear how it aligns with our **Product Vision** and if it's a defect of some kind, please let me know about the potential impact relative to other bugs that have recently been addressed.*
4. *To ensure that we don't need to declare bankruptcy again, we are going to take out less of a 'loan' this time around and limit the Product Backlog to [50] (insert manageable size) items. Once this quota is full, we can only add a new item by removing another or waiting for a space to open up.*

*Many thanks,
Your faithful Product Owner*

Clear benefits

Now this approach has several obvious benefits. Firstly, you are ensuring that the items in the Backlog are

current and relevant. Secondly, you ensure that all resurrected (and new) items go through the all-important *Product Vision* filter. Finally, you will find that this is a very manageable process, as new (or resurrected) items will be drip-fed to you while the various stakeholders take time to compile their cases for inclusion. This will ensure that the prioritizing process is progressive rather than all at once.

Wrapping Up

Humans are natural hoarders (you should check out my computer cable collection!) and that's why we often find ourselves in Backlog bedlam! That being said, our species also really appreciates the concept of a fresh-start. Backlog bankruptcy offers us that rare opportunity to start afresh, with the knowledge that our old lists are being archived (rather than destroyed) and thus avoiding scaring the natural hoarder in all of us!

[Download my PDF](#)

Posted: September 12, 2013

Tagged: product backlog, product owner, prioritizing

About the Author

Ilan Goldstein is an avid agilist with over a decade of practical hands-on experience.

He is a globally recognized Certified Scrum Trainer (CST) working with start-ups, market leaders, government agencies, and public companies around the world, helping them to improve their agility through the implementation of Scrum.

He is a regular conference speaker, guest university lecturer, and founder of both AxisAgile—a leading provider of agile training and consulting services—and Scrum Australia—a national not-for-profit

organization focused on growing and enriching the Scrum community Down Under.

He is the author of '*Scrum Shortcuts Without Cutting Corners*' the newest book to appear in my [signature Series](http://mikecohnsignatureseries.com), offering a broad selection of highly practical, real-world Scrum tactics, tools and tips.

You may also be interested in:

[Be a Great Product Owner: Six Things Teams and Scrum Masters Need](#)

Learn six ways effective product owners ensure their teams' success.

Jan 31, 2023 • 16 Comments

[Read](#)

[What Are Agile Story Points?](#)

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • [122 Comments](#)

[Read](#)

[Epics, Features and User Stories](#)

I've been getting more and more emails lately from people confused about the difference between ...

Nov 08, 2022 • [93 Comments](#)

[Read](#)

[Relationship between Definition of Done and Conditions of Satisfaction](#)

I'd like to clarify the relationship between two important concepts: a team's Definition of Done ...

Jun 28, 2022 • [38 Comments](#)

[Read](#)

[Four Reasons Agile Teams Estimate Product Backlog Items](#)

Estimating product backlog items provides benefits beyond predicting when a project will be finished.

May 17, 2022

[Read](#)

The Product Owner's Second Team

Successful product owners will see their stakeholders as a team, not merely a set of individuals.

Nov 02, 2021

[Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Product Backlog Bankruptcy!

Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

by
Mike Tagged:
Cohn product backlog, user stories, sprinting, meetings, daily scrum,
31 audio
Comments

This article has an audio version: [Download Audio Version](#)

I was recently interviewed for an upcoming agile textbook written by Sondra Ashmore and Kristin Runyan, and they asked me some questions about backlog refinement, such as who should attend, how to maximize the value and if the meetings can ever be fun. I'd like to share with you my thoughts on those questions today.

Who Should Attend?

Product backlog refining is not yet an official Scrum meeting. It's simply something that many have discovered as valuable, and can lead to a more productive sprint-planning meeting.

Last year, I wrote about when backlog refinement might be [accepted as a general process](#) in Scrum and the fact that it's probably still a few years away. As such, there's still no consensus on who should attend.

Although I'm generally a big believer of the whole team's involvement, that isn't really practical for this meeting. Here's a couple reasons why:

1. Product backlog refinement often happens two to three days before the end of a sprint. There is almost always someone on the team who is frantically busy two or three days before the end of a sprint. If we make that person attend another meeting, we could risk the delivery of whatever product backlog item the person is working on.
2. A good rule of thumb is that about 5 to 10 percent of the effort in each sprint should be spent on backlog refinement. While the whole team's involvement would be nice, not all team members may be able to participate.

How Do You Maximize the Value?

Maximizing the value of a backlog refinement meeting probably comes down to the same few things for any meeting:

- Keep it as short as possible.
- Show up prepared.
- Encourage everyone to participate.

Remember, conversations about the product backlog are not limited to a certain time or single meeting, so anyone can participate at any time. In the previous section, I mentioned that they often happen two to three days before the end of the sprint, but it's really whatever works for your team.

When you're refining the backlog, remember that it's not required that all product backlog items (usually in the form of user stories) are perfectly understood at the beginning of a sprint. The features only need to be sufficiently understood so the team has a reasonably strong chance of completing it during the sprint.

Can Backlog Refining Be Fun?

I have a hard time thinking of any meeting as truly fun, but I do think that when working with the right teammates, meetings can be viewed as welcomed breaks from the more intense work of the team.

Good teams can establish a rhythm where they alternate from very intense mental work either alone or in pairs, punctuated with occasional meetings. The meetings can have a social tone, and be an excuse for joking around with teammates or just taking a mental recess before diving back into the more intense work.

I've heard people come up with all sorts of ways to make Scrum meetings interesting. Most of the ideas I've heard are centered on when people are late to the daily scrum (for example, paying a jar, singing in front of teammates or telling a joke), but that doesn't mean those same creative ideas can't be inspiration for a less-embarrassing way to keep the refinement meeting light.

Spend a few minutes talking about what you've been up to, your family or anything else that makes the meeting seem like less of a daunting task.

In sum, refining the product backlog doesn't have to happen every sprint, but you should budget time for it to keep small, sprint-sized items at the top of the product backlog so you can defer an investment in items that will be worked on later.

Get 200 Real Life User Stories

Examples Written by Mike Cohn

See user stories Mike Cohn wrote as part of several real product backlogs.

First Name

Email Address

[Privacy Policy](#)

Posted: July 3, 2013

Tagged: product backlog, user stories, sprinting, meetings, daily scrum, audio

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

User Stories: How to Create Story Maps

Story maps help to create a shared understanding of the product, visualize user needs, and elicit ...

Mar 14, 2023

[Read](#)

How to Run a Successful User Story Writing Workshop

What you need to know to conduct a story-writing workshop with your team.

Feb 28, 2023 • 3 Comments [Read](#)

What Happens When During a Sprint

Succeeding with Scrum is easier when you know when and why to conduct each of the Scrum events ...

Jan 03, 2023 • 34 Comments [Read](#)

What Are Agile Story Points?

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

Epics, Features and User Stories

I've been getting more and more emails lately from people confused about the difference between ...

Nov 08, 2022 • 93 Comments [Read](#)

Advantages of User Stories over Requirements and Use Cases

At the surface, user stories appear to have much in common with use cases and traditional ...

Oct 05, 2022 • 41 Comments [Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile Assessment		

Names Should Not Be Needed for User Stories

by
Mike Cohn Tagged:
product backlog, user stories, audio
13 Comments

This article has an audio version: [Download Audio Version](#)

I've never been a fan of naming user stories. Stories should generally be short enough that naming them should not be necessary.

Yes, novels and movies get names. But we don't name the sentences in a novel. Since I still haven't been to see *The Great Gatsby* movie I started re-reading the novel recently so it's on my mind. Imagine Fitzgerald and his editor (Maxwell Perkins) discussing the book and Fitzgerald saying of the opening, "This is the

'remembering my father giving me advice' sentence. And this is the 'father gives that advice sentence.' And, Maxwell, what do you think of 'fundamental decencies' sentence?"

If I picture Fitzgerald and Perkins discussing a final draft of the sentence, I'm much more likely to picture Fitzgerald saying, "Maxwell, What do you think of this sentence?" and then pointing to it.

And that's just how I interact with user stories. I call them 'this' and I point to them. Or I simply read them aloud to the person I'm talking to--in the same way Fitzgerald might have read a sentence to Perkins in order to discuss it.

So, my advice is not to name your user stories. And definitely don't number them. That gets way too confusing. In [User Stories Applied](#) I advised against numbering stories, especially with numbers like 15, 15.1, and 15.1.1 to indicate relationships. Maybe a tool does that for you and you can ignore the numbers until the rare occasion on which you need the number. But I sure wouldn't number stories on cards manually. I have encountered one team though that used a Bates stamp to number their cards. ([Bates Numbering](#) refers to using a type of stamp that automatically increments each time the stamp is depressed. It's very common in the legal field and that's where I encountered the one team who did it.)

For themes, which are [groups of related user stories](#), I will usually put a separate index card on top of the theme and label it. I tend to use one- to three-word labels like "Reporting" or "Loan Processing" or "Shopping Cart Improvements" or "Performance" and so on. Within the theme, though, individual stories are not named.

Now, back to reading *The Great Gatsby*, quite happy that Fitzgerald just tells the story and didn't name each sentence.

[Download my PDF](#)

Posted: June 26, 2013

Tagged: product backlog, user stories, audio

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

Story maps help to create a shared understanding of the product, visualize user needs, and elicit ...

Mar 14, 2023

[Read](#)

[How to Run a Successful User Story Writing Workshop](#)

What you need to know to conduct a story-writing workshop with your team.

Feb 28, 2023 • [3 Comments](#)

[Read](#)

[What Are Agile Story Points?](#)

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • [122 Comments](#)

[Read](#)

Epics, Features and User Stories

I've been getting more and more emails lately from people confused about the difference between ...

Nov 08, 2022 • [93 Comments](#)

[Read](#)

Advantages of User Stories over Requirements and Use Cases

At the surface, user stories appear to have much in common with use cases and traditional ...

Oct 05, 2022 • [41 Comments](#)

[Read](#)

[Relationship between Definition of Done and Conditions of Satisfaction](#)

I'd like to clarify the relationship between two important concepts: a team's Definition of Done ...

Jun 28, 2022 • [38 Comments](#)

[Read](#)

The discussion here is closed but join us in the Agile Mentors Community to
further discuss this topic.

Names Should Not Be Needed for User Stories

Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile Assessment		

Three Tips for New ScrumMasters

by

Mitch **Tagged:**

Lacey product backlog, product owner, teams, continuous improvement,

13 transitioning to agile, scrum master

Comments

We're starting something new with this post--we will occasionally publish articles by guest authors. I'm very happy to have as our first guest post, "Three Tips for New ScrumMasters" by Mitch Lacey. Mitch is the author of [The Scrum Field Guide](#), which is full of great advice for new and experienced ScrumMasters.

One of the most common questions I get is "Now that I've taken a CSM class, what should I look out for when I return to the office?" While every situation is different, most new ScrumMasters should be aware of the following three issues.

First, remember the values and principles, the why-we-do-what-we-do portion of agile. Without a good

set of principles and values, people will often flail because they lack a clear understanding of the why, the meaning behind the practices.

Common indicators of a failure to grasp the principles behind the agile activities include:

- The team holding weekly "standup" meetings
- Teams not updating their task status on a Scrum board or tool
- Burndowns that are flat and suddenly drop to finish on time
- Not holding grooming meetings

For a refresher on the values and principles, watch my [Scrum for Managers video](#) - about the first 20 minutes - and reacquaint yourself with the why of Scrum and agile.

Another common issue has to do with architecture and design. In CSM classes, I discuss design and architecture but I don't go very deep on it. What I do say, however, is...

- Build incrementally
- Grow features over time
- Do the simplest thing possible
- Value feedback over "getting it right the first time"

The one people seem to struggle the most with is breaking the "getting it right the first time" mindset. One of my teams was able to build a system that supported millions of users and put it into production after a month by valuing feedback and throwing away the "getting it right the first time" mindset. How did we do this? Check out [this talk from Microsoft TechDays in Sweden in 2011 for more information](#).

Lastly, many teams lack a clear definition of done. The DoD is a list of sorts that includes the product owner's and team's criteria for declaring a product backlog item as potentially shippable, or done. It's important for several reasons.

- It sets a common understanding throughout the organization on what "done" means
- It is a communication tool for stakeholders
- It helps remove technical debt

Creating a DoD is not as easy as it might sound. I have a detailed chapter in my book, [The Scrum Field Guide](#), on this topic - and I have the chapter online as well. [Check out how to create your own Definition of Done here.](#)

They say an ounce of prevention is worth a pound of cure—this adage definitely applies to new ScrumMasters and teams. All ScrumMasters will encounter obstacles and challenges as they guide their teams. If you focus on resolving these three issues right away, however, you can avoid many of the common dysfunctions that plague agile efforts.

[Download my PDF](#)

Posted: June 11, 2013

Tagged: product backlog, product owner, teams, continuous improvement, transitioning to agile, scrum
master

About the Author

Mitch Lacey is an agile practitioner and trainer. Mitch has been managing projects for over fifteen years & is credited with many plan-driven & agile projects. He is the author of "[The Scrum Field Guide](#)", a book targeting teams adopting Agile and Scrum practices.

Mitch honed his agile skills at Microsoft Corporation, where he successfully released core enterprise services for Windows Live. Mitch's first agile team at Microsoft was coached by Ward Cunningham, Jim Newkirk & David Anderson.

As a Certified Scrum Trainer (CST) and a registered Project Management Professional (PMP), Mitch shares his experience in project and client management through Certified ScrumMaster courses, agile coaching engagements, conference presentations, blogs & white papers.

He has presented at Agile Alliance Agile 2006, 2007, 2008 and 2009 conferences, the 2008 Better Software Conference and the 2008 - 2013 SQE Agile Development Practices conferences. He was the stage producer for the Organization and Culture track for Agile 2009 and continued that trend by producing the Leadership and Organizations track for Agile 2010 and 2011, was the Agile2012 conference chair and is the Agile2014 conference chair. He has served on the board of directors for both the Scrum Alliance and the Agile Alliance.

You may also be interested in:

What Is Cross-Functional Collaboration in Agile?

Perhaps the most prevalent and persistent myth in agile is that a cross-functional team is one on ...

Feb 14, 2023 • 49 Comments [Read](#)

Be a Great Product Owner: Six Things Teams and Scrum Masters Need

Learn six ways effective product owners ensure their teams' success.

Jan 31, 2023 • 16 Comments [Read](#)

Six Attributes of a Great ScrumMaster

Scrum Masters exhibit a shared set of traits. Discover 6 in-demand qualities, plus a bonus ...

Jan 17, 2023 • 32 Comments [Read](#)

What Are Agile Story Points?

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

Don't Equate Story Points to Hours

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • 39 Comments [Read](#)

Epics, Features and User Stories

I've been getting more and more emails lately from people confused about the difference between ...

Nov 08, 2022 • 93 Comments [Re](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile

New to Agile and Scrum?

Scrum

User Stories

Planning Poker

Agile Software Development

Agile Project Management

Agile Presentations

Mountain Goat on YouTube

Agile Mentors Podcast

Elements of Agile

Assessment

Agile & Scrum Training

Online Certifications

Video Courses

In-Person Certifications

On-Site Courses

Training Schedule

Compare Courses

PDUs and SEUs

More...

About Us

Contact Us

Our Students

Blog

Books by Mike Cohn

Agile Tools

Using Scrum on an Analysis Project

by

Mike Cohn

14

Comments

Tagged:
product backlog, teams, sprinting, analysis project

Last week I wrote about "[Sprint Zero](#)" and made the point that on the rare occasion when this might be a good idea, Scrum teams would be better off thinking of that time as a "project before the project." Projects do not spring to life fully formed--that is, staffed and ready to be worked on. The vast majority of projects can, though, be instantly started and things like technical environments and an initial product backlog figured out during the first sprint--a good, old fashioned sprint number one.

But some projects should be preceded by an effort to decide what the project should entail and even whether there should be a project at all. This is the type of analysis work many teams put into a special Sprint Zero. While one sprint is often enough for this work, it's not hard to imagine a project large enough that it could

benefit from a couple of sprints (albeit, usually short, one-week sprints). Although I think this is rare, I'd like to offer further advice on conducting such a "project-before-the-project."

A project-before-the-project can be more simply thought of as an analysis project. It may answer questions such as:

- What features would be delivered?
- About how long would it take?
- How much investment will be necessary?
- Should the project be undertaken?

On a project like this, there is no "potentially shippable product increment" in the normal sense of working software delivered at the end of a sprint. So, what then does a project such as this deliver?

To answer questions like this about what to do when applying Scrum outside its home territory of product development, I find it helpful to consider the reasons why particular Scrum rules are in place. The requirement to deliver a potentially shippable product increment exists to help Scrum teams avoid analysis paralysis and to feel a sense of urgency to produce something within the sprint. Just the right amount of urgency can generate greater creativity. The rule also exists to help Scrum teams stay honest—it prevents them from claiming progress when none has been truly made.

To see how these rules can apply on an analysis project, suppose a team is considering developing a new system for hospitals and is running one week sprints to answer questions like those above. At the end of a sprint, the team needs to be truly *done* with something. They can't be truly done developing a feature since they are only analyzing the system. But one way the team could be done with something is for them to have interviewed all the pharmacists who are stakeholders on the project. The team now knows everything pharmacists will need including new functionality relating to prescribing medications, looking up patient records, and so on. The team will not need to talk to pharmacists about their basic requirements again. They can be said to be *done* interviewing pharmacists.

Alternatively, the team could instead have talked to all stakeholders regarding their needs for the prescriptions part of the system. In this case, the team spends the sprint interviewing not just pharmacists but also some medical doctors, nurses, and so on. They don't talk to these stakeholders about anything but the prescriptions part of the system. They'll come back and talk to stakeholders about other functionality in successive interviews. For now, though, this team can be said to be *done* analyzing needs of all stakeholders involved with prescriptions.

These examples illustrate how to apply the principles that underlie Scrum rules that can be a bit software- or product-development specific. While they allow a team to apply Scrum outside its home territory, I want to repeat my caution that I'd prefer not to undertake such an effort at all. Most projects can just start, with these issues resolved as the project begins.

[Download my PDF](#)

Posted: March 5, 2013

Tagged: product backlog, teams, sprinting, analysis project

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

[What Is Cross-Functional Collaboration in Agile?](#)

Perhaps the most prevalent and persistent myth in agile is that a cross-functional team is one on ...

Feb 14, 2023 • 49 Comments

[Read](#)

What Happens When During a Sprint

Succeeding with Scrum is easier when you know when and why to conduct each of the Scrum events ...

Jan 03, 2023 • [34 Comments](#)

[Read](#)

What Are Agile Story Points?

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • [122 Comments](#)

[Read](#)

Don't Equate Story Points to Hours

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • [39 Comments](#)

[Read](#)

Epics, Features and User Stories

I've been getting more and more emails lately from people confused about the difference between ...

Nov 08, 2022 • [93 Comments](#)

[Read](#)

Relationship between Definition of Done and Conditions of Satisfaction

I'd like to clarify the relationship between two important concepts: a team's Definition of Done ...

Jun 28, 2022 • [38 Comments](#)

[Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

Sprint Zero: A Good Idea or Not?

by
Mike Cohn 45
Tagged: product backlog, product owner, project management
[Comments](#)

In my previous post, I wrote about [alternatives to numbering sprints](#). In this post I want to deal with a very special number that some teams use in numbering their sprints--zero.

The concept of a "sprint zero" has become popular with some teams and so it is important to consider whether or not this is a good idea.

First, let's agree on the basic premise of "sprint zero." Sprint zero is usually claimed as necessary because there are things that need to be done before a Scrum project can start. For example, a team needs to be assembled. That may involve hiring or moving people onto the project. Sometimes there is hardware to

acquire or at least set up. Many projects argue for the need to write an initial product backlog (even if just at a high level) during a sprint zero.

One of the biggest problems with having a sprint zero is that it establishes a precedent that there are certain sprints or sprint types that have unique rules. Teams doing a sprint zero, for example, will dispense with the idea of having something potentially shippable at the end of that sprint. How can they have something potentially shippable after all if the goal of the sprint is to assemble the team that will develop the product?

I find that many of these things that can be used to argue for the need for a sprint zero are really best thought of as things that happen in what I call the *project before the project*. Before a development project begins, there is often a project to decide if there *should be* a development project. Before a company begins a major new initiative, someone has to think about whether that initiative should be undertaken at all.

Making that decision can be viewed as a project.

Since Scrum works well as a general purpose project management framework, it can be used to manage the work of this project-before-the-project. During this project-before-the-project, the early team members (perhaps just a future product owner) can work toward creating an initial product backlog, finding or hiring team members, setting up the technical environment, and so on.

I find it helpful to think of this work as a project of its own because it is not hard to imagine this work taking longer than one sprint, the special sprint zero. What does a team using a sprint zero call their second sprint if they need one to do whatever work they've used to justify the special type of sprint? Is it sprint 0.5?

A couple of cautions are necessary at this point:

Keep any "project before the project" as lightweight as possible. Most development projects do not need a project before they begin.

Remain true to the principles of Scrum. A team participating in a project before the project will not be able to have anything potentially shippable. That's fine. But understand why having something potentially shippable is a central tenet of Scrum and apply that in a honest way to the project before the project.

I wrote more about this last topic--staying true to the principles of Scrum on a project-before-the-project-[here](#).

Get Your Free Scrum Cheat Sheet

Sign up below to receive this FREE reference.

First Name

Email Address

[Privacy Policy](#)

Posted: February 26, 2013

Tagged: product backlog, product owner, project management

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

Be a Great Product Owner: Six Things Teams and Scrum Masters Need

Learn six ways effective product owners ensure their teams' success.

Jan 31, 2023 • 16 Comments [Read](#)

What Are Agile Story Points?

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

Epics, Features and User Stories

I've been getting more and more emails lately from people confused about the difference between ...

Nov 08, 2022 • 93 Comments [Read](#)

From Project Manager to Scrum Master -3 Tips for Making the Transition

What does a good project manager need to do to become a great Scrum Master?

Aug 23, 2022 [Read](#)

Relationship between Definition of Done and Conditions of Satisfaction

I'd like to clarify the relationship between two important concepts: a team's Definition of Done ...

Jun 28, 2022 • 38 Comments [Read](#)

Four Reasons Agile Teams Estimate Product Backlog Item

Estimating product backlog items provides benefits beyond predicting when a project will be finished.

May 17, 2022 [Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile **Agile & Scrum Training** **More...**

New to Agile and Scrum? Online Certifications About Us

Scrum Video Courses Contact Us

User Stories In-Person Certifications Our Students

Planning Poker On-Site Courses Blog

Agile Software Development Training Schedule Books by Mike Cohn

Development Compare Courses Agile Tools

Agile Project Management PDUs and SEUs

Agile Presentations

Mountain Goat on YouTube

Agile Mentors Podcast

Elements of Agile

Assessment

The Product Owner in a Sprint Retrospective

by
Mike Cohn 8
Tagged: product backlog, product owner, teams, sprinting
[Comments](#)

I've decided we should kick all the testers out of sprint retrospectives. We don't need them. They aren't really part of the team, anyway. They think differently and their issues aren't really the same issues faced by the real team.

How'd that sound? It was hard to even type so hopefully it took you pretty far aback.

However, I occasionally hear exactly that same message except *about product owners*. I can't think of a single good reason why a team would want to hold a retrospective without the product owner present. I can think of many *bad reasons* why they may want to--but

each of those should be the topic of a retrospective!

If you're tempted to run a sprint retrospective without your product owner, I strongly suggest you think about why. Product owners are full, first-class team members. It's critical that they participate in retrospectives and they are as open as everyone else to hearing things they can do to improve. Teams that don't include their product owner tend to suffer from us vs. them thinking that is almost always harmful to the project.

I hope your next retrospective is successful. To help that happen, make sure the product owner is there.

[Download my PDF](#)

Posted: February 6, 2013

Tagged: product backlog, product owner, teams, sprinting

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the *Better User Stories* video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

[What Is Cross-Functional Collaboration in Agile?](#)

Perhaps the most prevalent and persistent myth in agile is that a cross-functional team is one on ...

Feb 14, 2023 • 49 Comments

[Read](#)

[Be a Great Product Owner: Six Things Teams and Scrum Masters Need](#)

Learn six ways effective product owners ensure their teams' success.

Jan 31, 2023 • [16 Comments](#)

[Read](#)

[What Happens When During a Sprint](#)

Succeeding with Scrum is easier when you know when and why to conduct each of the Scrum events ...

Jan 03, 2023 • [34 Comments](#)

[Read](#)

What Are Agile Story Points?

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • [122 Comments](#)

[Read](#)

Don't Equate Story Points to Hours

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • [39 Comments](#)

[Read](#)

Epics, Features and User Stories

I've been getting more and more emails lately from people confused about the difference between ...

Nov 08, 2022 • [93 Comments](#)

[Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile Assessment		

by
Mike Cohn 24
Tagged: product backlog, user stories, teams, sprinting, customers
[Comments](#)

A Scrum trainer recently asked for a couple of good, real examples of large user stories (epics) being split into smaller stories. I thought it would be useful to share those examples here as well since the companies where these came from gave me permission to share them.

Example 1: Selecting Marketing Campaigns

This first example is from a company that sells software to large retailers--think of WalMart or a company such as that. The target user is a vice president of marketing who is trying to figure out how to spend the ad budget. The user's big goal is this first story, an epic:

1: As a VP Marketing, I want to review the performance of historical promotional campaigns so that I can identify and repeat profitable campaigns.

The team working on this said it was clearly an epic and so I asked them to split it. They turned it into:

1a: As a VP Marketing, I want to select the timeframe to use when reviewing the performance of past promotional campaigns, so that ...

1b: As a VP Marketing, I can select which type of campaigns (direct mail, TV, email, radio, etc.) to include when reviewing the performance of past so that ...

[Note I'm numbering these stories in a way to make it easy to see which story came from which parent story. On a real project I wouldn't necessarily do that unless I were using a tool that did it automatically.]

I didn't know if 1a or 1b were large and should be split again. So I asked the team, "If I were to ask you to estimate these, what unit instantly pops into your minds--hours, days, weeks, months or years?" For 1a they said days, so we left it alone. For 1b they said weeks. We split it up into:

1b1: As a VP Marketing, I want to see information on direct mailings when reviewing historical campaigns so that...

1b2: As a VP Marketing, I want to see information on TV ads when reviewing historical campaigns so that...

1b3: As a VP Marketing, I want to see information on email ads when reviewing historical campaigns so that...

and so on for each type of ad campaign.

Each of the stories above was reasonably small ("our estimate of these would be in days"). However, each needed one more level of detail, called a story's "conditions of satisfaction," which are essentially high-level acceptance tests. For 1b2 (the television story above), they wrote the following on the back of that story's index card:

See how many viewers by age range.

See how many viewers by income level.

Example 2: Maximizing Hotel Revenue

Let's look at another example. This one is from a company I worked with that was in the revenue maximization business. The idea was that a hotel wants to charge as much as it can for its rooms. They set room prices based on a large number of factors such as what other hotels are charging, the time of year, any known local events (e.g., if the Super Bowl or World Cup are in town, the rates go up). Here was the initial user story, an epic:

1: As a hotel operator, I want to set the optimal rate for rooms in my hotel.

To keep this example a little simpler while still using real stories from this product backlog, let's assume the hotel has only one type of room. As I've said, a number of data elements can affect the desired price of a hotel room. The general formula for pricing a hotel room then becomes this:

$$\text{RoomRate} = f(a,b,c\dots)$$

RoomRate is a function of a variety of factors. For each factor there was a story such as these:

1a: As a hotel operator, I want to set the optimal rate for rooms based on prior year pricing.

1b: As a hotel operator, I want to set the optimal rate for rooms based on what hotels comparable to mine are charging.

and so on...

Story 1a is really just saying that prior year pricing information is used in the RoomRate function. Story 1b says that what comparable hotels are charging is also a factor in the RoomRate equation.

These stories are interesting in that each was fairly large (an epic) on its own. It would be impossible to implement all of them in one sprint as there were many more than the two above. As each was implemented, the RoomRate function above would calculate the wrong price because it was being built first as:

```
RoomRate = f (a)  
then  
RoomRate = f (a,b)  
then  
RoomRate = f (a,b,c)  
and so on.
```

So if after one sprint the calculation is RoomRate = f (a), the system is generating the wrong RoomRate (so it can't be shown to customers) but it is being calculated with the right math as far as can be done. Maybe values for (b) and (c) are hard coded or maybe they're just ignored. But working this way allows complex algorithms like that to be built up incrementally.

But because story 1b above is too big, here's how it was split:

- 1 b1:** As a hotel operator, I can define a 'Comparable Set' of hotels. [Comparable Set was a term used widely in the company.]
- 1b2:** As a hotel operator, I can add a hotel to a Comparable Set.
- 1b3:** As a hotel operator, I can remove a hotel from a Comparable Set.
- 1b4:** As a hotel operator, I can delete a Comparable Set I no longer wish to use.
- 1b5:** As a hotel operator, rates charged at hotels in a Comparable Set are used to determine rates at my hotel.

Here were a couple of additional stories:

- 1c:** As a hotel operator, I want to set the optimal rate for rooms based on current projected occupancy.
- 1d:** As a hotel operator, I can set parameters related to optimal rates such as target occupancy, minimum acceptable occupancy and maximum acceptable occupancy [could be more than 100%].

I find that learning how to split user stories is something nearly all teams struggle with at first. My experience is that sometime within the first year of doing it, things start to click into place and team members discover all the little local patterns for story splitting unique to their domain and types of projects. So, if you're relatively new to agile or user stories, stick with--it will get easier.

[Take the Assessment](#)

Posted: December 17, 2012

Tagged: product backlog, user stories, teams, sprinting, customers

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

User Stories: How to Create Story Maps

Story maps help to create a shared understanding of the product, visualize user needs, and elicit ...



Mar 14, 2023

How to Run a Successful User Story Writing Workshop

What you need to know to conduct a story-writing workshop with your team.

Feb 28, 2023 • 3 Comments [Read](#)

What Is Cross-Functional Collaboration in Agile?

Perhaps the most prevalent and persistent myth in agile is that a cross-functional team is one on ...

Feb 14, 2023 • 49 Comments [Read](#)

What Happens When During a Sprint

Succeeding with Scrum is easier when you know when and why to conduct each of the Scrum events ...

Jan 03, 2023 • 34 Comments [Read](#)

What Are Agile Story Points?

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

Don't Equate Story Points to Hours

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • 39 Comments [Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

by Andy Crowe



Published Feb 2012

Publisher: Velociteach

I was looking forward to reading this book. I haven't paid much attention to the PMI-ACP initiative and thought this book would help me see what the test covered. The book is extremely short: there are less than 140 pages between the introduction and the glossary. The entire book is double-spaced so it is a very quick read. After the main book is a glossary and then two practice tests with 100 questions each.

Unfortunately, the book is self-published and is a showcase of all that can go wrong with self-publishing. I can get past the repeated promotional offers in the text, on the back cover and on a business card taped to the inside back. They were a bit annoying but theoretically well intentioned. What I can't get past is the book's need for good technical and copy editing.

The writing is generally crisp and very understandable but the lack of copy editing left me shaking my head at too many places for a \$60 book. For example, consider this quote:

"One way this is accomplished is through Cause and Effect diagrams. These are also known as Ishikawa diagrams or fishbone diagrams. These charts are used to illustrate how different factors might relate together and to identify a root problem. Cause and effect diagrams, or Ishikawa Diagrams, are also known as fishbone diagrams. (p. 77)"

OK, maybe I'm just more sensitive to copy editing errors because I'm a writer. And I can get past the galley proofreading problems like paragraphs that truncate randomly like on page 93. But these are examples of the dangers of self-publishing. (And are one reason I'm hesitating about self-publishing my next book, even though I already produce camera-ready pages for my publisher.)

Unfortunately the book is full of a number of technical errors:

"Customer" is called a formal role on Scrum teams (pp. xxii-xxiii)
The iteration backlog "shows all the functionality that the team will complete during the current iteration." (p. 106) This means the iteration backlog is the subset of product backlog items selected for the iteration, not the tasks (or "sprint backlog items").
Story points are assigned to iteration backlog items. (p. 189 and 230) This stems from the fundamental misunderstanding above.
Software cannot be released during an iteration. (page 194 and 233) The book takes the 1995 view of Scrum that software is only releasable at the end of a sprint. Happily, this isn't 1995. (Perhaps I shouldn't be too harsh on the book on this point. Maybe the PMI-ACP tests people on vintage 1995 Scrum. However, if so, the book should certainly point out the fallacy.)

What I found most fascinating is that the author learned about Agile in the late 1990s:

My journey with Agile began in the late 1990s when I sat through a presentation on a new methodology known as the Rational Unified Process... During this presentation, the facilitator mentioned that he had been working with teams that worked with no actual project manager. He referred to this as "Agile." I peppered the instructor with questions and left with my head spinning. (p.. x)

Given that the Agile Manifesto was the first documented use of "agile" in relation to software development and was written in 2001, I find it shocking that the author heard another speaker call it Agile in the late 1990s. I was doing Scrum before that but had certainly never heard the term "agile" anywhere near a software conversation before. Perhaps it was this slight, ahem, exaggeration in the first paragraph of the book that poisoned my view of the rest of the book.

So, while I can't recommend this book very highly, I do want to say a few good things about it:

The two sample tests at the end seem good. (I assume they accurately reflect what is covered on the test.) I very much liked that the answers were explained rather than just provided as a list of the correct answer for each question.

The book is well-written (despite its need for some editors). It was, for the most part, a quick and pleasant read--no excessively long paragraphs, no places where I got lost wondering what the author's point was, etc.

The book is short. In an era where too many books exceed my attention span or interest level, it was no problem quickly reading this one. Of course, some topics are covered in exceedingly short sections (Agile Modeling gets four sentences; Minimal Marketable Feature gets one paragraph). But I assume that's all that's necessary to pass the test.

So, the flaws in the book prevent me from going so far as to recommend it. But if you want to get a feel for what's on the PMI-ACP test, or if you have solid agile experience and want a quick book to read in a couple of hours to fill in gaps, or if you're looking for some good sample tests, you'll probably like this book.

Tagged: product backlog, teams, sprinting, story points, backlog, book reviews

You may also be interested in:

[What Is Cross-Functional Collaboration in Agile?](#)

Perhaps the most prevalent and persistent myth in agile is that a cross-functional team is one on ...

Feb 14, 2023 • 49 Comments

[Read](#)

[What Happens When During a Sprint](#)

Succeeding with Scrum is easier when you know when and why to conduct each of the Scrum events ...

Jan 03, 2023 • 34 Comments

[Read](#)

What Are Agile Story Points?

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • [122 Comments](#)

[Read](#)

Don't Equate Story Points to Hours

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • [39 Comments](#)

[Read](#)

Epics, Features and User Stories

I've been getting more and more emails lately from people confused about the difference between ...

Nov 08, 2022 • [93 Comments](#)

[Read](#)

Relationship between Definition of Done and Conditions of Satisfaction

I'd like to clarify the relationship between two important concepts: a team's Definition of Done ...

Jun 28, 2022 • [38 Comments](#)

[Read](#)

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile Assessment		

Handling Requirements from Architects Outside the Team

by
Mike Cohn Tagged:
18 product backlog, product owner, teams, agile requirements
[Comments](#)

I was recently asked what I thought about using the "Wise Architects" in a company to provide technical oversight to the multiple teams on a project.

A common objection to this is that the architects are outside the team and should not, therefore, have any say in how the team builds whatever it is that they are building.

This argument doesn't hold water, though, as there are other outsiders who provide requirements to the

team. But those requirements are always filtered through a product owner who decides whether they are important or not.

An organization's Wise Architects often provide requirements to a team in the form of non-functional requirements. I think of non-functional requirements as "constraints" on how a team solves a problem. So the Wise Architect cannot tell a team *how* they solve a problem, but can provide constraints on how it's solved--the system must scale to a certain number of concurrent users, it must process this many transactions per minute, it must run on Linux, it must integrate with our such-and-such, etc. Non-functional requirements become product backlog items and can be prioritized by the product owner based on how important the product owner views compliance with each. For example, if a product owner decides that running on Linux is not critical and the product could be just as successful on a different server OS, the product owner would remove that non-functional requirement or perhaps place it low on the product backlog so the team is at least aware the architect wants it.

[Take the Assessment](#)

Posted: July 10, 2012

Tagged: product backlog, product owner, teams, agile requirements

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the *Better User Stories* video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

What Is Cross-Functional Collaboration in Agile?

Perhaps the most prevalent and persistent myth in agile is that a cross-functional team is one on ...

Feb 14, 2023 • 49 Comments [Read](#)

Be a Great Product Owner: Six Things Teams and Scrum Masters Need

Learn six ways effective product owners ensure their teams' success.

Jan 31, 2023 • 16 Comments [Read](#)

What Are Agile Story Points?

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

Don't Equate Story Points to Hours

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • 39 Comments [Read](#)

Epics, Features and User Stories

I've been getting more and more emails lately from people confused about the difference between ...

Nov 08, 2022 • 93 Comments [Read](#)

Relationship between Definition of Done and Conditions of Satisfaction

I'd like to clarify the relationship between two important concepts: a team's Definition of Done ...

Jun 28, 2022 • 38 Comments [Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

The Rules vs. The Generally Accepted Practices of Scrum

by
Mike Cohn
Tagged:
product backlog, user stories, product owner, teams, sprinting,
agile practices
[Comments](#)

In a [post back in March](#) I introduced a term on this blog that I'd been trying out in discussions and a few classes. The term was GASP and it stood for a *Generally Accepted Scrum Practice*. What I'm interested in right now, and I'm hoping everyone here will help with, is creating a list of all the GASPs we can think of.

But first, we need to more formally define what a GASP is. Here's my definition:

A Generally Accepted Scrum Practice (GASP) is an activity performed by many, but not necessarily all,

Scrum teams. A team that does not perform the practice can still be considered to be doing Scrum. So, something like "work in short, timeboxed iterations no longer than a calendar month" is not a GASP. It is more of a Scrum rule. To be considered a GASP, the practice must be something "generally accepted" as a good idea. As a working definition of that, I've been using the idea that "every Scrum team should be aware of the practice but some teams may justifiably choose not to perform that practice, often choosing to do something similar instead."

For example, I'm going to suggest that "Conduct a sprint review meeting at the end of the sprint" is a GASP not a Scrum rule. Can a team really be considered to be doing Scrum if they forego a sprint review meeting? Yes, and I think it's becoming more common. We are seeing more teams who do as-needed mini-reviews rather than a big sprint review only at the end. For example, a team doing web development may conduct a short review with their product owner after each user story is complete and then release the new functionality to the website immediately. Is this better or worse than the single at-end sprint review? I can't say, but in some contexts I'm sure it is. And I sure wouldn't tell a team doing multiple mini-reviews that they "aren't doing Scrum."

What makes "Conduct a sprint review meeting at the end of the sprint" a GASP in this example is that the team should know about the idea of the end-of-the-sprint review. It's a good idea. Lots of teams do it. And the team in our example is free to reject it in favor of something that works better for them but they should know about it.

So, to be a GASP, a practice has to have moved beyond just a promising good new idea. It has to be something that enough Scrum teams are doing that we can call the practice "generally accepted." I'm going to suggest that "Consider user stories as your product backlog items" is a GASP. I think stories have certainly proven a worthy approach to the product backlog. Does a team have to work with user stories to do Scrum? Of course not. But I do think stories are in such common use that a team should know what they are before rejecting them in favor of something better.

In contrast to a GASP, a rule is an inviolable thing that if a team isn't doing, they aren't doing Scrum.

So: What do you think are the generally accepted practices of Scrum? What do you think are its rules?

Download Scrum Master Guide

Get a free copy of Situational Scrum Mastering: Leading an Agile Team

[Get my free guide now!](#)

Posted: June 28, 2012

Tagged: product backlog, user stories, product owner, teams, sprinting, agile practices

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

User Stories: How to Create Story Maps

Story maps help to create a shared understanding of the product, visualize user needs, and elicit ...

Mar 14, 2023

[Read](#)

How to Run a Successful User Story Writing Workshop

What you need to know to conduct a story-writing workshop with your team.

Feb 28, 2023 • [3 Comments](#)

[Read](#)

What Is Cross-Functional Collaboration in Agile?

Perhaps the most prevalent and persistent myth in agile is that a cross-functional team is one on ...

Feb 14, 2023 • [49 Comments](#)

[Read](#)

[Be a Great Product Owner: Six Things Teams and Scrum Masters Need](#)

Learn six ways effective product owners ensure their teams' success.

Jan 31, 2023 • [16 Comments](#)

[Read](#)

[What Happens When During a Sprint](#)

Succeeding with Scrum is easier when you know when and why to conduct each of the Scrum events ...

Jan 03, 2023 • [34 Comments](#)

[Read](#)

What Are Agile Story Points?

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • [122 Comments](#)

[Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

GASPing About the Product Backlog

by

Mike Tagged:

Cohn product backlog, sprinting, backlog, daily scrum, continuous

81 improvement

Comments

I've been wondering lately if Scrum is on the verge of getting a new standard meeting--the Backlog Grooming Meeting, which is a meeting an increasing number of teams are doing each sprint to make sure the [product backlog](#) is prepared and ready for the start of the next sprint.

a product backlog

To see why a Backlog Grooming Meeting may be a few years away from becoming a Generally Accepted Scrum Practice, or what I call a GASP, let's revisit the early 2000s.

Back then, Scrum didn't have a formal [Sprint Retrospective Meeting](#). Prevailing wisdom at the time was, in fact, fairly opposed to such a meeting. The logic was that a good Scrum team should be always on the lookout for opportunities to improve; they should not defer opportunities to discuss improvement to the end of a sprint.

That argument was a very good one. However, what was happening on teams was that day-to-day urgencies took precedence and opportunities to improve often went either unnoticed or unacted upon. And so what most teams eventually realized was that of course we should improve any time we notice an opportunity, but at a minimum each team should set aside a dedicated time each sprint for doing so--and thus the retrospective became a standard part of Scrum. This was helped along tremendously by the great book, [Agile Retrospectives: Making Good Teams Great](#), by Esther Derby and Diana Larsen.

I've had more CSM course attendees recently asking questions about a Backlog Grooming Meeting as though it were a GASP. Many are surprised when I tell them that not every Scrum team has such a meeting each sprint. I still don't advocate every team conduct a Backlog Grooming Meeting each sprint--as with the early arguments against retrospectives, I'd prefer backlog grooming to happen in a more continuous, as-needed way--but so many teams are successfully using a Backlog Grooming Meeting, arguments against it may be on their last gasps.

Share what you think below. Will a Product Backlog Grooming meeting become so common it becomes a Generally Accepted Scrum Practice (GASP)?

[Watch Now](#)

Posted: March 14, 2012

Tagged: product backlog, sprinting, backlog, daily scrum, continuous improvement

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the *Better User Stories* video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

What Happens When During a Sprint

Succeeding with Scrum is easier when you know when and why to conduct each of the Scrum events ...

Jan 03, 2023 • 34 Comments [Read](#)

What Are Agile Story Points?

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

Epics, Features and User Stories

I've been getting more and more emails lately from people confused about the difference between ...

Nov 08, 2022 • 93 Comments [Read](#)

Elements of Agile: An Agile Assessment Tool for Iterative Improvements

New to agile, or needing an agile re-boot? We've got a new no-cost assessment and actionable ...

Sep 13, 2022 [Read](#)

Relationship between Definition of Done and Conditions of Satisfaction

I'd like to clarify the relationship between two important concepts: a team's Definition of Done ...

Jun 28, 2022 • 38 Comments [Read](#)

Four Reasons Agile Teams Estimate Product Backlog Item

Estimating product backlog items provides benefits beyond predicting when a project will be finished.

May 17, 2022 [Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile **Agile & Scrum Training** **More...**

New to Agile and Scrum? Online Certifications About Us

Scrum Video Courses Contact Us

User Stories In-Person Certifications Our Students

Planning Poker On-Site Courses Blog

Agile Software Development Training Schedule Books by Mike Cohn

Development Compare Courses Agile Tools

Agile Project Management PDUs and SEUs

Agile Presentations

Mountain Goat on YouTube

Agile Mentors Podcast

Elements of Agile

Assessment

by
Mike Cohn **Tagged:**
product backlog, user stories
77 Comments

I want to show a real easy way to put user stories in a spreadsheet-based [product backlog](#). I wrote this after seeing someone tweet a screen capture of a product backlog I made nine years ago and thought to myself, "Yikes, that's out of date for how I do it today ..." So in this post, we'll look at an agile product backlog template in Excel.

As you probably know, I'm a big fan of writing the product backlog in the form of [user stories](#) and of writing user stories in the form: "As a _____, I _____, so that _____. " An example being: "As a frequent flyer, I really want to be able to connect to the internet while flying so that I can update my blog while traveling rather than having to save this as a text file and updating my blog later." (Can you guess where I am while writing this?)

What I've found makes a user story in this format very easy to work with in an agile Excel spreadsheet is to take the boilerplate parts and put them into column headings. So we'll have column headings like "As a" and "I" and "so that". The meat of each story is then clearly visible in each row. Additional columns can be added for things like a unique identifier, notes, status and such. In this example, I've also included a column for the theme or grouping of which the story is a part. You can see this in the screen capture below.

You can click the image for a larger view of the agile Excel spreadsheet.

[the product backlog in Excel](#)

Get Your Free Scrum Cheat Sheet

Sign up below to receive this FREE reference.

First Name

Email Address

[Privacy Policy](#)

Posted: July 1, 2011

Tagged: product backlog, user stories

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the *Better User Stories* video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

User Stories: How to Create

Story Maps

Story maps help to create a shared understanding of the product, visualize user needs, and elicit ...

Mar 14, 2023

[Read](#)

How to Run a Successful User Story Writing Workshop

What you need to know to conduct a story-writing workshop with your team.

Feb 28, 2023 • 3 Comments [Read](#)

What Are Agile Story Points?

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

Epics, Features and User Stories

I've been getting more and more emails lately from people confused about the difference between ...

Nov 08, 2022 • 93 Comments [Read](#)

Advantages of User Stories over Requirements and Use Cases

At the surface, user stories appear to have much in common with use cases and traditional ...

Oct 05, 2022 • 41 Comments [Read](#)

Relationship between Definition of Done and Conditions of Satisfaction

I'd like to clarify the relationship between two important concepts: a team's Definition of Done ...

Jun 28, 2022 • 38 Comments [Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile

New to Agile and Scrum?

Scrum

User Stories

Planning Poker

Agile Software Development

Agile Project Management

Agile Presentations

Mountain Goat on YouTube

Agile Mentors Podcast

Elements of Agile Assessment

Agile & Scrum Training

Online Certifications

Video Courses

In-Person Certifications

On-Site Courses

Training Schedule

Compare Courses

PDUs and SEUs

More...

About Us

Contact Us

Our Students

Blog

Books by Mike Cohn

Agile Tools

by

Mike **Tagged:**

Cohn product backlog, user stories, product owner, continuous

21 improvement, lean

[Comments](#)

The weather turned nice about two weeks ago, which meant it was time for spring cleaning about the Cohn home, affectionately known as the Cohnderosa (which will only mean something if you're old enough to remember "Bonanza"). While washing the windows around the outside of the house I had plenty of time to think about spring cleaning I'd also just helped a couple of clients with--we cleaned up their product backlogs.

In order to clean up the product backlog, I want to introduce you to a new Scrum artifact--the Long-Term Product Backlog. The Long-Term Product Backlog is maintained by the product owner and is usually round, black and sits next to or under the product owner's desk. Left unattended, a product backlog can become large and hard to work with. If your backlog has reached this point, take some time to do some spring cleaning--review the backlog and delete / throw away user stories that you can finally admit you're never going to get to.

Some product owners or teams feel this is an admission of failure. It's not. In most cases it's an admission of success--we're throwing away feature ideas that we once thought important because since writing them down we've discovered even more important features. So celebrate the fact that you've got newer, bigger, better feature ideas and delete the less valuable ones from your product backlog.

If permanently deleting such ideas is too big of a step, perhaps you really do want to introduce a new artifact into your process. Create a spreadsheet and paste the user stories there for safe keeping. Or print a report from your product

backlog tool and file the report somewhere safe just in case. In my experience, though, you won't miss them. And just like the bright new view through the windows of the Conderosa, you'll be able to see the rest of your product backlog much more clearly.

[Download my PDF](#)

Posted: May 6, 2011

Tagged: product backlog, user stories, product owner, continuous improvement, lean

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

User Stories: How to Create Story Maps

Story maps help to create a shared understanding of the product, visualize user needs, and elicit ...

Mar 14, 2023

[Read](#)

How to Run a Successful User Story Writing Workshop

What you need to know to conduct a story-writing workshop with your team.

Feb 28, 2023 • 3 Comments [Read](#)

Be a Great Product Owner: Six Things Teams and Scrum Masters Need

Learn six ways effective product owners ensure their teams' success.

Jan 31, 2023 • 16 Comments [Read](#)

What Are Agile Story Points?

Story points are perhaps the most misunderstood topic in agile.

Story points are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

Epics, Features and User Stories

I've been getting more and more emails lately from people confused about the difference between ...

Nov 08, 2022 • 93 Comments [Read](#)

Advantages of User Stories over Requirements and Use Cases

At the surface, user stories appear to have much in common with use cases and traditional ...

Oct 05, 2022 • 41 Comments [Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile

New to Agile and Scrum?

Scrum

User Stories

Planning Poker

Agile Software Development

Agile Project Management

Agile Presentations

Mountain Goat on YouTube

Agile Mentors Podcast

Elements of Agile

Assessment

Agile & Scrum Training

Online Certifications

Video Courses

In-Person Certifications

On-Site Courses

Training Schedule

Compare Courses

PDUs and SEUs

More...

About Us

Contact Us

Our Students

Blog

Books by Mike Cohn

Agile Tools

by
Mike Cohn
Tagged:
product backlog, continuous improvement, agile projects
75 Comments

I was recently asked what kind of project is most suited for the [agile process](#). In my view, the most appropriate projects for agile are ones with aggressive deadlines, a high degree of complexity, and a high degree of novelty (uniqueness) to them. We want to use agile when we are doing something that is new, or at least new to the team building it. If it's something the team has done before over and over then the team probably doesn't need an agile approach.

To my mind, this is where some of the manufacturing analogies come in. If we are building the same car day after day, we learn pretty quickly all the nuances of building that car. We don't need an agile approach because the novelty of the situation is low. Novelty alone does not mean we should use an agile process.

I went to my favorite Chinese restaurant for lunch recently. I ordered an entree "triple extra spicy and with jalapenos." It was probably the first time they cooked this particular dish that way and so it was a novel or unique order. The cook prepared it wonderfully though and because I could see into the kitchen I'm sure they didn't need a daily standup or even TDD to make my lunch. (I might have noticed a kanban back there, however. So, in addition to novelty, the project needs a certain amount of complexity. One final element I believe is required in making a project appropriate for agile is urgency. The timeboxes and iterations of an agile approach are devised to keep the intensity and focus going on a project.

If there's no urgency to the project, those are unneeded. So let's see how these three factors--urgency, complexity, and novelty--mix on various projects, starting of course with software projects. There couldn't be a better fit. Software projects are notoriously complex. Each software project is largely a new endeavor. And in today's world, there is almost

always a sense of urgency.

But let's look at one other situation where we commonly hear about Scrum (in particular) being applied: getting married. At least a couple of times a year I hear about a couple who planned their wedding using Scrum. There is always a wedding backlog--buy cake, pick photographer, send invitations, pick dress, etc. How does planning a wedding do against the three factors I'm proposing. Sense of urgency? Check. There's always a deadline and it's usually pretty fixed. Complexity? Well, it's not like a software project but it does have its own complexities often enhanced by non-functional requirements such as a fixed budget, who sits next to whom, the type of food to be served, the need to let Cousin Ira's band perform at the reception, etc. Novelty? Yep. Most people don't get married enough times with large celebrations that planning the event becomes second hand.

So, agile is most appropriate on any urgent project with significant complexity and novelty--and that includes software development and weddings. It does raise the question though of whether a couple's first kiss at the end of the ceremony is a product backlog item or part of the done criteria for the overall product.

[Download my PDF](#)

Posted: January 15, 2011

Tagged: product backlog , continuous improvement , agile projects

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com.

If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

What Are Agile Story Points?

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

Epics, Features and User Stories

I've been getting more and more emails lately from people confused about the difference between ...

Nov 08, 2022 • 93 Comments [Read](#)

Elements of Agile: An Agile Assessment Tool for Iterative Improvements

New to agile, or needing an agile re-boot? We've got a new no-cost assessment and actionable ...

Sep 13, 2022

[Read](#)

Relationship between Definition of Done and Conditions of Satisfaction

I'd like to clarify the relationship between two important concepts: a team's Definition of Done ...

Jun 28, 2022 • 38 Comments [Read](#)

Four Reasons Agile Teams Estimate Product Backlog Items

Estimating product backlog items provides benefits beyond predicting when a project will be finished.

May 17, 2022

[Read](#)

Needs, Wants, and Wishes on Your Product Backlog

To prioritize a backlog, think about what users need, what they want, and what they merely wish for.

Oct 12, 2021

[Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

Agile Teamwork: 3 Ways to Minimize Handoffs

by Tagged:

Mike product backlog, product owner, teams, sprinting, backlog,
Cohn programming, teamwork, testing, customers, scrum master, lean,
8 articles
Comments originally published in Better Software on 2010-04-05

Agile teamwork can really help to minimize handoffs. Teams using a sequential development process have become accustomed to handoffs between specialists. Analysts hand their work to designers, who hand it to programmers, who pass it on to testers. Teams that are new to Scrum often do not go far enough in eliminating these handoffs, wrongly assuming, for instance, that the programmers should finish programming a product backlog item before handing it off to the testers or that analysts should work at least one sprint ahead of the rest of the team.

High-performing Scrum teams, however, have learned to do a little bit of everything all the time during a sprint, thereby eliminating large handoffs. To do this effectively, scrum teams must make three changes: favor talking over writing, make handoffs very small and very often, and mix the size of items that are brought into each sprint.

Agile Teamwork - Stop Writing and Start Talking

There's a grand myth about requirements: If you write them down, users will get exactly what they want. That's not true. At best, users will get exactly what was written down, which may or may not be anything like what they really want. As such, Scrum teams forego a lengthy, up-front requirements phase and the resulting product specification in favor of a dynamic product backlog.

Because Scrum teams shift focus from writing about requirements to talking about them, conversations with the product owner—rather than a product spec—become the primary way of finding out how a new feature should behave. Team members talk with the product owner about how a feature should work, how quickly it should perform, what acceptance criteria must be passed, and so on. Scrum team members also talk with users, customers, and other stakeholders as appropriate and, perhaps most importantly, with each other.

On traditional projects, analysts often act as intermediaries, communicating customer needs to programmers. On Scrum teams, however, analysts function as facilitators, ensuring that appropriate discussions between team and product owner take place. For instance, an analyst might steer the product owner and team toward talking about a particular user story because it has more risk than another of going astray. At other times, an analyst might convey a top-level understanding of a new feature to the team before bringing the team and product owner together to discuss the details. In all cases, analysts on Scrum teams foster communication rather than distill information through a document.

All of this is not to say we should abandon written requirements documents. Rather, we should use documents where appropriate. Experienced Scrum teams lean heavily on cleanly written code and automated test cases. They then augment these forms of documentation with a written requirements document only to the extent that such a document is helpful or required for regulatory, contractual, or legal purposes. As Tom Poppendieck, coauthor of books on lean software development, once told me, "When documents are mostly to enable handoffs, they are evil. When they capture a record of a conversation that is best not forgotten, they are valuable."

Agile Teamwork - Transfer Small Tasks Frequently

On Scrum teams, the unit of transfer between disciplines should be smaller than an individual product backlog item. That is, although there will always be some handoffs, the amount of work being transferred from one person to the next should generally be as small as possible.

As an example, suppose a team is developing a new eCommerce application. The team chooses to work on this user story from its product backlog: "As a shopper, I can select how I want items shipped based on the actual costs of shipping to my address so that I can make the best decision." At the beginning of the sprint, those who are interested in or who will be involved in developing this feature begin to discuss it. Let's suppose this group includes the product owner, a business analyst, a tester, and a programmer. They begin by talking about some of the general requirements implicit in this feature: Which shipping companies (FedEx, DHL, etc.) do we support? Do we want to support overnight delivery? Two-day delivery? Three-day delivery?

As these discussions occur, the individuals involved naturally will begin thinking about how to get started. On a traditional project, each person would be able to start however he wanted (after the work was handed over). On a Scrum team, however, how to get started should be a collaborative discussion among those who will work on this feature. For this example, let's assume that the programmer for some reason makes the case that it will be easier to start with FedEx. The tester agrees. The analyst states an intention to investigate DHL and learn more about the parameters that affect DHL shipping costs. The analyst's goal is to have that information available by the time the programmer and tester finish with FedEx.

When the programmer knows enough to get started coding, she does so. The product owner, analyst, and tester discuss high-level tests (Will our site ship any odd-sized items like skis?). After that discussion, the tester turns the high-level list of tests into concrete tests (boxes of this size and weight going to that destination). The tester creates test data and automates the tests. Some automation may be possible without any interim deliverables from the programmer, while full automation may require getting an early version from the programmer. While the tester is thinking of the concrete tests, he also should inform the programmer of any test cases that she may not be considering while she's programming.

When the programmer and tester have finished, they add support for calculating FedEx shipping costs into the build, complete with automated tests. Graphically, this can be depicted as shown in figure 1, where four individuals work together on one product backlog item rather than handing it off to each other.

Next, the programmer and tester check in with the business analyst, who hopefully has learned more about calculating DHL shipping costs. The process is repeated, and support for DHL shipping calculations is added to the application when the programming and testing are complete. The key element in figure 1 is that the team has learned to work by doing a little of everything all the time. Rather than an analysis phase (done without the programmer and tester) followed by a programming phase followed by a testing phase, a little of each of those activities is happening at all times.

"A symptom of continuing to hand off work in overly large chunks is a tendency for no product backlog items to be finished until the last few days of the sprint."

A symptom of continuing to hand off work in overly large chunks is a tendency for no product backlog items to be finished until the last few days of the sprint. Testers on teams that work this way often complain that they are given nothing to test until two days before the end of a sprint and then are expected to test everything that quickly. The best way to expose this problem is to create a chart of the number of product backlog items finished as of each day in the sprint. An example can be seen in figure 2a. As the ScrumMaster on a team, I often just hang this chart in the team area with no fanfare or explanation. Team members soon figure out the problem exposed by a chart like this and, hopefully, start to find ways to finish product backlog items sooner. The result often will be similar to figure 2b, which shows a much smoother

flow through the sprint.

Agile Teamwork - Create a Manageable Mix

When planning a sprint, teams should pay attention to the sizes of the product backlog items to which they are committing. Some product backlog items are more complex than the FedEx/DHL example given. It's possible that some product backlog items might require a week or more of programming before a programmer can provide a tester with something even initially testable. That's OK. Not everything can be split as small as we might like. The key is to avoid bringing a bunch of items like this into the same sprint. Doing so will shift too much testing work to the end of the sprint.

Instead of planning a sprint with, for example, three very large items that cannot be partially implemented, bring one or two such items into the sprint along with two or three smaller items. Some of the programmers can work on the large items, handing them over to testers whenever possible. The remaining programmers can work on the smaller items, ensuring a somewhat smoother flow of work to testers early in the sprint.

Creating the right sense of teamwork can be challenging. ScrumMasters can help by ensuring that the team embraces the concept of whole-team responsibility and whole-team commitment to deliver working software at the end of each sprint. Though the team might struggle at first to break longheld habits of specialization and handoffs, increasing communication, decreasing the size of handoffs, and mixing the size of backlog items brought into a sprint will help individuals make the shift from sequential development to working as a team.

[Download my PDF](#)

Posted: April 4, 2010

Tagged: product backlog, product owner, teams, sprinting, backlog, programming, teamwork, testing, customers, scrum master

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile

New to Agile and Scrum?

Scrum

User Stories

Planning Poker

Agile Software Development

Agile Project Management

Agile Presentations

Mountain Goat on YouTube

Agile Mentors Podcast

Elements of Agile

Assessment

Agile & Scrum Training

Online Certifications

Video Courses

In-Person Certifications

On-Site Courses

Training Schedule

Compare Courses

PDUs and SEUs

More...

About Us

Contact Us

Our Students

Blog

Books by Mike Cohn

Agile Tools

by Roman Pichler



Published Apr 2010

Publisher: Addison-Wesley Professional

As a project management framework, Scrum introduces many changes. One of the biggest is the role of the product owner who represents the users or customers of a product or system. The product owner is responsible for making sure the right product is being built and in the right order. This forces the product owner to think iteratively and incrementally about the product--rather than a small set of large decisions made at the outset of a project, the Scrum product owner makes many more but smaller decisions throughout the course of a typical development project.

This excellent book provides new and experienced product owners with the guidance they will need to work in this new way. The book focuses on precisely what you need to know in order to be a great product owner. Author Roman Pichler assumes that the reader is either an experienced traditional product manager learning Scrum or will pick up an additional book on traditional product management. This allows him to focus specifically on the unique product management challenges of using Scrum. He covers how to create a shared vision of the product, which is more difficult on Scrum as its iterative nature avoids a prolonged upfront specification phase. Pichler covers thorough coverage of creating a product backlog, planning a release, and collaborating with the team during the sprints ("iterations") of the project. He also provides advice on how to transition into the new role of product owner.

There is a shortage of fantastic product owners in the world. This book will help fix that problem.

Tagged: product backlog, product owner, sprinting, backlog, management, customers, project management,
book reviews

You may also be interested in:

Be a Great Product Owner: Six Things Teams and Scrum Masters Need Learn six ways effective product owners ensure their teams' success. Jan 31, 2023 • 16 Comments Read	What Happens When During a Sprint Succeeding with Scrum is easier when you know when and why to conduct each of the Scrum events ... Jan 03, 2023 • 34 Comments Read	What Are Agile Story Points? Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ... Dec 06, 2022 • 122 Comments Read	Epics, Features and User Stories I've been getting more and more emails lately from people confused about the difference between ... Nov 08, 2022 • 93 Comments Read	From Project Manager to Scrum Master -3 Tips for Making the Transition What does a good project manager need to do to become a great Scrum Master? Aug 23, 2022 Read	Relationship between Definition of Done and Conditions of Satisfaction I'd like to clarify the relationship between two important concepts: a team's Definition of Done ... Jun 28, 2022 • 38 Comments Read
--	--	--	--	--	---

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		

Distributed Teams: Build Trust through Early Progress

by

Mike Cohn

Tagged: product backlog, product owner, sprinting, agile project

Cohn management, scrum master, release planning, distributed teams

Comments

Critical to creating a coherent team is building trust among team members. This is much more difficult on a distributed team. Unable to rely on repeated, frequent face-to-face communication, distributed teams need to take other measures to build trust. Traveling ambassadors, starting meetings with casual conversations, occasional in-person meetings of the full team, working agreements, and similar activities all help. What also helps is early pressure for the team to produce working software by the end of each sprint, even the earliest ones. Unfortunately, many projects schedule too much time for teambuilding exercises and discussions too

early in the project. This is a common and dangerous [agile project management](#) mistake, as shown by research from Professor Lynda Gratton and her coauthors, as published in the *MIT Sloan Management Review*.

Guiding these diverse teams to success requires some counterintuitive management practices.

In particular, team leaders should focus on tasks at the early stages, rather than on interpersonal relationships, and then switch to relationship building when the time is right. (Gratton, Voigt, and Erickson. "Bridging faultlines in diverse teams." *MIT Sloan Management Review*, Summer 2007, 22–29.)

Though this research suggests that the early focus should be on tasks, I do not mean to suggest that product owners or ScrumMasters should be assigning tasks to developers. Rather, I mean that these leaders should emphasize the need for the team to make demonstrable progress even in the early sprints.

The problem with an early emphasis on relationship building is that it encourages less-than-ideal subgroups to form. Any large group will inevitably split into subgroups. If these subgroups are allowed to form too early they will form around surface-level attributes—Americans, Swedes, C++ programmers, Java programmers, female database engineers, male programmers, and so on.

As Gratton and her coauthors write, "Simply put, in a team's early going, the more people interact with one another, the more likely they are to make snap judgments and to emphasize their differences" (26). What we'd like to do is defer relationship building until team members have learned more significant things about each other, such as specific skills, competencies, approaches to work, and so on. This is done through early emphasis on progress rather than relationship building. The subgroups that form at that time will be based on the mutual need to work together to develop the product.

To develop a particular user story from the product backlog, you and I need to work together. In doing so we learn each other's skills and specific competencies. I come to know you not just as a Java programmer but as a Java programmer with a real passion and strength for automated unit testing. You find that I am not just a DBA, but one who is strong at optimizing SQL statements. Teams with subgroups formed around compatible skills, attitudes, approaches to work, and so on are less likely to lead to a later breakdown in trust than subgroups formed on superficial attributes (such as American, Swede, programmer, tester, and so on).

Think back to one or more troubled teams you were a member of. Odds are that conflict on those teams was of an us-versus-them nature based on superficial attributes: this office versus that office, programmer versus DBA, Linux fanatics versus Windows fanatics. When teams feel an immediate need to make progress, those types of subgroups do not have time to form. After a team has worked together for a few sprints, shift the emphasis toward relationship building by incorporating more social activities and shared downtime into the sprints.

A team needs to have a sufficient amount of shared experience before social activities and relationship building can be useful. But when that has been achieved, "instilling confidence in the team and creating opportunities to socialize at that point helps the development of new abilities and allows the team to grow" (29). I want to be clear that I am not saying that no time for socialization should be included at the start of a project. Seeding visits and whole-team, in-person get-togethers at the start of a project for its initial release planning can be very useful. Three points are key here:

First, the project should start with intensity and a focus on early demonstrations of progress.

Second, the entire "budget" for socialization should not be spent in the first couple of sprints.

Third, early social activities should tie into the work of the project, such as bringing a team together for

release planning.

[Download my PDF](#)

Posted: February 22, 2010

Tagged: product backlog, product owner, sprinting, agile project management, scrum master, release planning, distributed teams

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the *Better User Stories* video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

Be a Great Product Owner: Six Things Teams and Scrum Masters Need

Learn six ways effective product owners ensure their teams' success.

Jan 31, 2023 • 16 Comments [Read](#)

Six Attributes of a Great ScrumMaster

Scrum Masters exhibit a shared set of traits. Discover 6 in-demand qualities, plus a bonus ...

Jan 17, 2023 • 32 Comments [Read](#)

What Happens When During a Sprint

Succeeding with Scrum is easier when you know when and why to conduct each of the Scrum events ...

Jan 03, 2023 • 34 Comments [Read](#)

What Are Agile Story Points?

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

Epics, Features and User Stories

I've been getting more and more emails lately from people confused about the difference between ...

Nov 08, 2022 • 93 Comments [Read](#)

From Project Manager to Scrum Master -3 Tips for Making the Transition

What does a good project manager need to do to become a great Scrum Master?

Aug 23, 2022 [Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile **Agile & Scrum Training** **More...**

New to Agile and Scrum? Online Certifications About Us

Scrum Video Courses Contact Us

User Stories In-Person Certifications Our Students

Planning Poker On-Site Courses Blog

Agile Software Development Training Schedule Books by Mike Cohn

Development Compare Courses Agile Tools

Agile Project Management PDUs and SEUs

Agile Presentations

Mountain Goat on YouTube

Agile Mentors Podcast

Elements of Agile

Assessment

by
Mike Cohn **Tagged:**
teamwork, design, ui design, phases
33 [Comments](#)

One of the [12 principles in the Agile Manifesto](#) is "Working software (or product) is the primary measure of progress." That's why agile teams (e.g. [Scrum teams](#)) whether developing software or any other product, work together to deliver something of value to their customer each and every iteration.

For this to happen, agile teams embrace concurrent engineering. Concurrent engineering (or simultaneous engineering) is a way of working where tasks overlap, rather than happening in a series of phased handoffs.

The Main Benefit of Concurrent Engineering

Contrast overlapping work with sequential engineering, where product development work happens in phases. For example, on a software project, we might have an analysis phase followed by a design phase followed by a coding phase and ultimately a testing phase. At the end of each phase, work is handed off to the next person or team with the skills to complete the next phase of work.

If that happened on an agile software development project, it might take 4 iterations before a team could deliver something to the customer!

So [cross-functional agile teams](#) instead [collaborate to complete all activities necessary](#) to deliver a product increment inside a time-bound iteration (sometimes called a sprint). The various kinds of work overlap.

Using the previous example, on an agile team, as one developer is designing a user interface, a second team member begins coding the functionality, and a third developer begins to create tests for the functionality. All inside the same iteration!

At the end of the iteration, [high-performing agile teams](#) are able to deliver a fully conceptualized, designed, created, and tested increment of value to their customer.

Concurrent engineering speeds up product development and time to market—not because teams are working faster or harder, but because they are able to get small chunks of completed functionality into the hands of their users sooner. This gives organizations a tremendous competitive advantage, and is one of the many reasons companies [adopt an agile methodology](#) to begin with.

To make concurrent engineering work, agile teams must do three things: Avoid finish-to-start relationships, embrace uncertainty, and start separately but finish together.

Avoid Finish-to-Start Project Management Practices

When some teams first begin implementing agile, they cling to their sequential mindset and finish-to-start activities with a series of activity-focused sprints. They use one iteration for analysis and design, a second for coding, and a third for testing. The team is split in thirds, with the analysts working one sprint ahead of the programmers and the testers working one sprint behind them.

This can be a very alluring approach for teams who are [new to agile or Scrum](#). Not only does it seemingly solve the problem of how to overlap work but it also allows each type of specialist to work mostly with others of their own kind, which many are used to doing.

Unfortunately, the same disadvantages apply to activity-specific sprints as apply to activity-specific teams: too many hand-offs and a lack of whole-team responsibility.

Activity-specific sprints create what are known as finish-to-start relationships. In a finish-to-start relationship, one task must finish before the next can start.

For example, a Gantt chart on a sequential project may show that analysis must finish before coding can start and that coding must finish before testing can start.

Experienced agile teams learn that this is not true; many activities can be overlapped.

In agile projects what is important is not when tasks start but when they finish. Coding cannot finish until analysis finishes and testing cannot finish until coding finishes. These are known as finish-to-finish relationships.

Embrace Uncertainty

To start a task while a related task is not yet finished, the team needs to become willing to work around uncertainty and open issues temporarily.

The key thing for team members to understand is that while they eventually need an answer to those issues, they do not always need the answer before starting work on a [product backlog](#) item. Instead, they can share enough information (for example at the [daily scrum](#)) for other team members to get started.

As an example, suppose a team is working on a [user story](#), "As a user, I am logged out after n minutes of inactivity."

Before that story can be considered complete, someone is going to need to decide how long n is--30 minutes? 12 hours? But, someone could absolutely begin work on that story without the answer.

Once team members fully grasp that some answers can be learned during the iteration, they become much more willing to live with the uncertainty that is needed to practice the overlapping work of concurrent engineering.

This is an [iterative and incremental approach](#) to project management: Get a few details from the users about what they need and then build a little of it; build a little and then test what you've built.

The goal should be to start the sprint with just enough information that the team can barely finish each product backlog item. Team members should feel that if they had to resolve even one more open issue during the sprint, they could not have finished that product backlog item.

Start Separately But Finish Together

Some people argue that to maintain a holistic point of view, certain activities (e.g., user experience design, database design, and architecture) must happen upfront.

I argue that we should think holistically but work iteratively. One way to do that is to stagger when certain activities start.

With an agile approach to work, it doesn't so much matter when each team member starts on a product backlog item. What matters is that they all finish together, or as close to it as practical. In a 10-day iteration, one team member may start coding a user story on day six and another begins creating tests on day eight.

Their goal, however, is to finish together on day ten.

I equate this to running a race around a 400-meter track as in the Olympics. Because outside lanes are progressively longer, runners in those lanes begin the race further ahead physically on the track. This ensures that each person runs the same distance and that they finish at the same place, making it possible to judge the winner.

An agile team can think of certain specialists (analysts, graphic designers, product designers) being in the outside tracks in the development process. They need a bit of a head start. (Yes, I know in a real running race everyone starts running at the same time. But the starting line for the outside track is "ahead" of the inside track.)

The analyst needs to identify what's even being built. And the designer may need to provide wireframes, mockups or similar starting points to the team if the team is to have any chance to build and test the feature within a sprint. So giving them a bit of a head start by having them look ahead towards the work of the next iteration is a good idea.

Note that I said "a bit of a head start." The designer does not work apart from the team or work three sprints ahead. The designer is absolutely on the team and the designer's primary responsibility is helping the team in any way possible to finish the committed work of the current sprint. They just leave enough capacity to look ahead to the next sprint.

A good [product owner](#) does the same thing. A team is in trouble if the product owner is so buried by the work of the current sprint that the product owner arrives at the next sprint planning meeting without having given any thought to what to work on next.

Certain roles on an agile team need to be looking somewhat ahead. They should look ahead only as far as necessary, but some peering forward by product owners, analysts, or designers is helpful.

How Do You Do It?

How does your team achieve the goal of overlapping work using elements of concurrent engineering? Please share your thoughts in the comments section below.

[Take the Assessment](#)

Posted: December 19, 2017

Tagged: teamwork, design, ui design, phases

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

Scrum, Remote Teams, & Success: Five Ways to Have All Three
In a remote-first world, how does an agile team thrive working in a virtual, distributed way?

Jun 07, 2022

[Read](#)

What Is a High-Performing Agile Team?

How to build and sustain a Scrum team that exceeds the sum of its parts.

Apr 19, 2022

[Read](#)

What Does Scrum Mean by Cross Functional Teams?

What exactly does cross functional mean and why does it matter?

Apr 05, 2022

[Read](#)

Retrospecting with a Quiet Team

How to run a Retrospective when your team won't talk.

Feb 01, 2022

[Read](#)

25 Questions that Will Help You Know Your Teammates Better

One of the benefits of being on a team is the friendships you can make with your teammates. Many of ...

Jan 21, 2020 • 26 Comments [Read](#)

Six Agile Product Development Myths - Busted

Pervasive myths about agile get in the way of success. It's time to bust six of those myths.

Feb 19, 2019 • 47 Comments [Read](#)

Sorry, the browser you are using is not currently supported. Disqus actively supports the following browsers:

[Firefox](#)
[Chrome](#)
[Internet Explorer 11+](#)
[Safari](#)

[Tony Bresse](#) • 5 years ago

Hi Mike

I noticed this statement "There is no analysis phase followed by a design phase followed by a coding phase and ultimately a testing stage. "

Problem 1_ How can we code without design in mind

Problem 2_ How can we design without some kind of analysis in mind

Problem 3_ Not a best practice not to test (Like my profile quote.. not really)

Here is what I see if this statement is practiced with "No Analysis":

1_ We will go back to the swing thing that was supposed to be built based on specific design. you remember that?

[https://www.tamingdata.com/...](https://www.tamingdata.com/)

If that is followed, then that will cause rework and possible sprint delays

[Learn About Agile](#) [Agile & Scrum Training](#) [More...](#)

New to Agile and Scrum? [Online Certifications](#) [About Us](#)

Scrum [Video Courses](#) [Contact Us](#)

User Stories [In-Person Certifications](#) [Our Students](#)

Planning Poker [On-Site Courses](#) [Blog](#)

Agile Software Development [Training Schedule](#) [Books by Mike Cohn](#)

Compare Courses [Agile Tools](#)

Agile Project Management [PDUs and SEUs](#)

Agile Presentations

Mountain Goat on YouTube

Agile Mentors Podcast

Elements of Agile Assessment

Mix the Sizes of the Product Backlog Items You Commit To

by
Mike Cohn Tagged:
product backlog, teams, sprint planning, programming, testing,
agile project management
[Comments](#)

Teams used to a sequential development process have become accustomed to hand-offs between specialists. Analysts hand their work to designers who hand it to programmers who pass it on to testers. Each of these hand-offs includes some overhead in the form of meetings, documents to read and perhaps sign, and so on.

In part because of this overhead, the hand-offs tend to be of large amounts of functionality. In the purest meaning of a waterfall process, the entire application is handed from group to group. Teams that are new to

agile project management often do not go far enough in eliminating these hand-offs. They often make the assumption that the programmers should finish programming a [product backlog](#) item before handing it off to the testers.

This results in lengthy delays at the start of a sprint, when the testers are waiting for a first product backlog item to be handed to them. On a Scrum project, the unit of transfer between disciplines should be smaller than an individual product backlog item. That is, although there will always be some hand-offs (not everyone can be working on everything all the time), the amount of work being transferred from one person to the next should generally be as small as possible.

To facilitate these small transfers, Scrum teams learn to work by doing a little of everything all the time. Rather than an analysis phase (done without the programmer and tester) followed by a programming phase followed by a testing phase, a little of each of those activities is happening at all times. Even with an emphasis on minimal hand-offs, there will be some product backlog items that will require a week or more of programming time before the programmer can give something even beginning to be testable to a tester. That's OK. Not everything can be split as small as we might like.

To avoid a flurry of testing activity toward the end-of-sprint, avoid bringing a bunch of these complex items into the same sprint. Instead of planning a sprint with, for example, three very large items that cannot be partially implemented, bring one or two large items into the sprint along with two or three smaller items.

Some of the programmers can work on the large items, handing them over to testers whenever possible. The remaining programmers can work on the smaller items, ensuring a somewhat smoother flow of work to testers early in the sprint.

While your goal in each sprint should be to do a little bit of everything all the time, some backlog items are complex by nature and will not be completed until near or on the last day of a sprint. To avoid having multiple product backlog items wrapping up the last few days of the sprint, mix the size of the product backlog items you choose to bring into each sprint.

Posted: December 30, 2009

Tagged: product backlog, teams, sprint planning, programming, testing, agile project management

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the *Better User Stories* video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

[Why I Don't Emphasize Sprint Goals](#)

[What Is Cross-Functional Collaboration in Agile?](#)

[What Are Agile Story Points?](#)
Story points are perhaps the most

[Don't Equate Story Points to Hours](#)

[Epics, Features and User Stories](#)

[Relationship between Definition of Done and Conditions of](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile **Agile & Scrum Training** **More...**

New to Agile and Scrum? Online Certifications About Us

Scrum Video Courses Contact Us

User Stories In-Person Certifications Our Students

Planning Poker On-Site Courses Blog

Agile Software Development Training Schedule Books by Mike Cohn

Development Compare Courses Agile Tools

Agile Project Management PDUs and SEUs

Agile Presentations

Mountain Goat on YouTube

Agile Mentors Podcast

Elements of Agile

Assessment

Make the Product Backlog DEEP

by
Mike **Tagged:**
Cohn product backlog, user stories, product owner, sprinting, agile
15 product management
[Comments](#)

Roman Pichler, author of "*Agile Product Management with Scrum: Creating Products That Customers Love*" and I use the acronym DEEP to summarize key attributes of a good [product backlog](#):

Detailed Appropriately. User stories on the product backlog that will be done soon need to be sufficiently well understood that they can be completed in the coming sprint. Stories that will not be developed for awhile should be described with less detail.

Estimated. The product backlog is more than a list of all work to be done; it is also a useful planning tool. Because items further down the backlog are not as well understood (yet), the estimates associated with them will be less precise than estimates given items at the top.

Emergent. A product backlog is not static. It will change over time. As more is learned, user stories on the product backlog will be added, removed, or reprioritized.

Prioritized. The product backlog should be sorted with the most valuable items at the top and the least valuable at the bottom. By always working in priority order, the team is able to maximize the value of the product or system being developed.

Product backlogs are discussed in much more detail in [Succeeding with Agile](#).

Get Free User Stories Book Chapters

Please provide your name and email and we'll send you the sample chapters and we'll send a short weekly tip from Mike on how to succeed with agile.

[Get my chapters now!](#)

Posted: December 14, 2009

Tagged: product backlog, user stories, product owner, sprinting, agile product management

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

User Stories: How to Create Story Maps

Story maps help to create a shared understanding of the product, visualize user needs, and elicit ...

 Mar 14, 2023 [Read](#)

How to Run a Successful User Story Writing Workshop

What you need to know to conduct a story-writing workshop with your team.

Feb 28, 2023 • 3 Comments [Read](#)

Be a Great Product Owner: Six Things Teams and Scrum Masters Need

Learn six ways effective product owners ensure their teams' success.

Jan 31, 2023 • 16 Comments [Read](#)

What Happens When During a Sprint

Succeeding with Scrum is easier when you know when and why to conduct each of the Scrum events ...

Jan 03, 2023 • 34 Comments [Read](#)

What Are Agile Story Points?

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

Epics, Features and User Stories

I've been getting more and more emails lately from people confused about the difference between ...

Nov 08, 2022 • 93 Comments [Re](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile **Agile & Scrum Training** **More...**

New to Agile and Scrum? Online Certifications About Us

Scrum Video Courses Contact Us

User Stories In-Person Certifications Our Students

Planning Poker On-Site Courses Blog

Agile Software Development Training Schedule Books by Mike Cohn

Development Compare Courses Agile Tools

Agile Project Management PDUs and SEUs

Agile Presentations

Mountain Goat on YouTube

Agile Mentors Podcast

Elements of Agile

Assessment

Agile Design: Intentional Yet Emergent

by
Mike Cohn
Tagged:
product backlog, product owner, teams, sprinting, product management
13 Comments

The [agile process](#) favors an incremental, just-in-time approach to design. As such, Scrum projects do not have an upfront analysis or design phase; all work occurs within the repeated cycle of sprints. This does not mean, however, that design on a Scrum project is not intentional. An intentional design process is one in which the design is guided through deliberate, conscious decision making.

The difference on a Scrum project is not that intentional design is thrown out, but that it is done (like everything else on a Scrum project) incrementally. Scrum teams acknowledge that as nice as it might be to make all design decisions up front, doing so is impossible. This means that on a Scrum project, design is both intentional and emergent. The design emerges because there is no up-front design phase (even though

there are design activities during all sprints). Design is intentional because product backlog items are deliberately chosen with an eye toward pushing the design in different directions at different times.

As an example of how the product backlog items can be sequenced to influence the architecture of the system, consider a workflow system I worked on. The system supported a fund-raising company that produced specialized T-shirts and similar products. School-age children would go door to door selling these items. The sales revenue would be split between the company and the organization the kids represented, such as a school, sports team, or other group.

For each sale, the kid would complete a form and send it to the company, where it was scanned, sent through an optical character recognition (OCR) process, and converted into an order. To keep shipping costs down, orders from the same organization were batched together and sent back to the organization, after which the kids would hand-deliver the items.

Our software handled the entire process—from when the paper was received by the company until the shipment went out the door. Kids have notoriously bad writing and are bad spellers, so our system had to do more than just scan forms and prepare packing lists. There were various levels of validation depending upon how accurately we thought each order form had been read.

Some forms were routed to human data-entry clerks who were presented the scanned form on one side of the screen, the system's interpretation on the right, and an additional space to make corrections. Because thousands of shirts were processed on the busiest days, this process needed to be as automated as possible. I worked with the product owner, Steve, to write the product backlog.

After that, I met with the development team to discuss which areas of the system were the highest risk or we were the most uncertain about how to develop. We decided that our first sprint would focus on getting a high-quality document to run through the system from end to end. It would be scanned, go through OCR, and generate a packing list. We would bypass optional steps such as deskewing crooked pages, despeckling pages, and so on but would prove that the workflow could be completed from start to finish.

This wasn't highly valuable but it was something that needed to be done, and it let the developers test out the general architecture. After we accomplished this, we had a basic database in place and could move documents from state to state, triggering the correct workflow steps. Next the developers asked the product owner if they could work on the part of the system that would display a scanned document to a human who would be able to override the scanned and interpreted values. This was chosen as the second architectural goal of the project for three reasons:

It was a manual step, making it different from the workflow steps handled already.

Getting the user interface right was critical. With the volume of documents flowing through this system, saving seconds was important. We wanted to get early feedback from users to allow time to iterate on usability.

After this feature was added, users could start processing shirt orders.

The project continued in this way for a few months and was ultimately tremendously successful, meeting all of the prerelease targets for reliability and throughput. A key to the success was that the product owner and technical personnel worked together to sequence the work. The closest the team got to a design phase was the first afternoon in the conference room when we identified risky areas and dark corners and decided which one we wanted to tackle first.

From there the design emerged sprint by sprint, yet was intentionally guided by which product backlog items were selected to illuminate the dark corners and risks of the project. *Succeeding with Agile* goes into much more detail on agile design, including how the roles of architect and user experience designer change with Scrum, the concept of emergent design, and how teams work together and with the product owner to deliver increments of functionality that guide the design of the final product.

[Download my PDF](#)

Posted: December 4, 2009

Tagged: product backlog, product owner, teams, sprinting, product management

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the *Better User Stories* video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

[What Is Cross-Functional Collaboration in Agile?](#)

Perhaps the most prevalent and persistent myth in agile is that a cross-functional team is one on ...

Feb 14, 2023 • 49 Comments

[Read](#)

[Be a Great Product Owner: Six Things Teams and Scrum Masters Need](#)

Learn six ways effective product owners ensure their teams' success.

Jan 31, 2023 • [16 Comments](#)

[Read](#)

[What Happens When During a Sprint](#)

Succeeding with Scrum is easier when you know when and why to conduct each of the Scrum events ...

Jan 03, 2023 • [34 Comments](#)

[Read](#)

What Are Agile Story Points?

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • [122 Comments](#)

[Read](#)

Don't Equate Story Points to Hours

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • [39 Comments](#)

[Read](#)

Epics, Features and User Stories

I've been getting more and more emails lately from people confused about the difference between ...

Nov 08, 2022 • [93 Comments](#)

[Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

How Do You Get from Here to Agile? Iterate.

by

Mike Tagged:

Cohn product backlog, teamwork, continuous improvement, project
20 management, transitioning to agile, iterations, iterating
[Comments](#)

Historically, when an organization needed to change, it undertook a “change program.” The change was designed, had an identifiable beginning and ending, and was imposed from above. This worked well in an era when change was necessary only once every few years. But in today’s fast-paced, ever changing environment, it makes more sense to create agile organizations, ready to adapt to whatever comes their way.

But how do you manage the effort of moving from the point you are today—whether that’s just starting to adopt an [agile project management](#) approach or fine-tuning your implementation—to a place where you can readily react and respond to the vagaries of the market? By following an iterative transition process. Making small changes on a continual basis is a logical way to adopt a development process that is itself iterative.

Doing so will be much more likely to result in a successful and sustainable transition.

This is why I believe that the effort of adopting Scrum is best managed using Scrum itself. With its iterative nature, fixed timeboxes, and emphasis on teamwork and action, it seems best suited to manage the enormous project of becoming and then growing agile with Scrum. Just as Scrum development projects use product backlogs, you should use an improvement backlog to track the effort of adopting Scrum in your organization. An improvement backlog lists everything that the organization could do better in its use of Scrum.

If you're just starting with Scrum, your improvement backlog will emphasize creating awareness and desire. If the transition is already well underway, your improvement backlog may contain more items around developing the ability to do Scrum well, to promote successes, or to transfer it to other groups. A small department or single-project transition may involve a single improvement backlog. But when Scrum is being adopted across a large site, department, or organization, the transition effort becomes large enough that multiple improvement backlogs are used, each of which is created by a community of individuals who are passionate about improving the organization in a particular way. Many corporate improvement initiatives fail because plans are not made specific and actionable.

Using Scrum to manage the effort of becoming agile allows you to divide the work into stories and tasks with concrete deliverables and definite end dates. At the end of every iteration, the organization will have improved in measurable and visible ways. Eventually, you will have successfully transformed the organization into one that not only is agile but also that seeks to become more agile each day. Additional advice on using Scrum to manage your transition effort is provided in Chapter 4, "Iterating Toward Agility" of *Succeeding with Agile*.

[Download my PDF](#)

Posted: November 19, 2009

Tagged: product backlog, teamwork, continuous improvement, project management, transitioning to agile,

iterations, iterating

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

What Are Agile Story Points?

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

Epics, Features and User Stories

I've been getting more and more emails lately from people confused about the difference between ...

Nov 08, 2022 • 93 Comments [Read](#)

8 Reasons Scrum Is Hard to Learn (but Worth It)

Transitioning to Scrum is worth it, but some aspects are challenging.

Oct 11, 2022 [Read](#)

Elements of Agile: An Agile Assessment Tool for Iterative Improvements

New to agile, or needing an agile re-boot? We've got a new no-cost assessment and actionable ...

From Project Manager to Scrum Master - 3 Tips for Making the Transition

What does a good project manager need to do to become a great Scrum Master?

Aug 23, 2022 [Read](#)

Relationship between Definition of Done and Conditions of Satisfaction

I'd like to clarify the relationship between two important concepts: a team's Definition of Done ...

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile **Agile & Scrum Training** **More...**

New to Agile and Scrum? Online Certifications About Us

Scrum Video Courses Contact Us

User Stories In-Person Certifications Our Students

Planning Poker On-Site Courses Blog

Agile Software Development Training Schedule Books by Mike Cohn

Development Compare Courses Agile Tools

Agile Project Management PDUs and SEUs

Agile Presentations

Mountain Goat on YouTube

Agile Mentors Podcast

Elements of Agile

Assessment

by

Mike

Cohn

44

Tagged:

product backlog, user stories, product owner, defect management

Comments

A common [agile project management](#) question is whether bugs belong on the [product backlog](#). Before I address that question, let me clarify that the question refers to bugs that are unrelated to functionality being coded during the sprint. If someone finds a bug during a sprint that is related to the features being worked on, the best thing to do is yell, "Hey, Mike, the boojum code is broken when I..." The second best thing to do is to stick a new task card on the task board saying "Fix the boojum code; it breaks when..." But what about a bug found today in something the team wrote a month or a year ago? And what about the existing bug database that a team new to agile is almost sure to bring with them?

The ideal situation is to put the bugs right onto the product backlog. To see why, consider the situation from a user's perspective: Do you think a user cares whether something is considered a New Feature or a Bug? No. The user just wants the system to behave in some way different from how it behaves today. They don't care what it's called.

But notice I called this the *ideal situation*. It isn't always practical--and sometimes for reasons out of the team's control or influence. Bug databases have a way of becoming embedded into organizations. The tech support group uses them as a primary way of communicating with the developers. The marketing group uses the bug database in a similar way. This makes it quit the bug database cold turkey.

In these cases, the most common solution team's use is to have two backlogs

- a product backlog of features
- a bug backlog

This approach is a bit harder on the team and the product owner but allows an agile team to work more easily with existing processes in the organization.

Sometimes when we make such accommodations to the overall organization, the accommodation can damage or destroy the agile adoption. Allowing everyone to work on five concurrent projects is an example of a crippling accommodation. Accommodating the organization's use of a bug database is not crippling. It's just a bit more work.

The product owner takes on most of the additional work by having to prioritize two separate work queues and then present them to the team in a more or less consolidated manner ("My top priorities are the first three items on the product backlog, then bugs #12403 and 12415, then these next two items on the product backlog..."). This isn't too onerous as the items would need to be prioritized even if all were in one backlog.

So: Ideally bugs belong on the product backlog just like any feature request. But, that would often necessitate a significant change for the rest of the organization so two backlogs are used.

Get 200 Real Life User Stories Examples Written by Mike Cohn

See user stories Mike Cohn wrote as part of several real product backlogs.

First Name

Email Address

[Privacy Policy](#)

Posted: July 6, 2009

Tagged: product backlog, user stories, product owner, defect management

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

User Stories: How to Create Story Maps

Story maps help to create a shared understanding of the product, visualize user needs, and elicit ...

Mar 14, 2023

[Read](#)

How to Run a Successful User Story Writing Workshop

What you need to know to conduct a story-writing workshop with your team.

Feb 28, 2023 • [3 Comments](#)

[Read](#)

Be a Great Product Owner: Six Things Teams and Scrum Masters Need

Learn six ways effective product owners ensure their teams' success.

Jan 31, 2023 • [16 Comments](#)

[Read](#)

[What Are Agile Story Points?](#)

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • [122 Comments](#)

[Read](#)

[Epics, Features and User Stories](#)

I've been getting more and more emails lately from people confused about the difference between ...

Nov 08, 2022 • [93 Comments](#)

[Read](#)

Advantages of User Stories over Requirements and Use Cases

At the surface, user stories appear to have much in common with use cases and traditional ...

Oct 05, 2022 • [41 Comments](#)

[Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

Using a Task Board with One Remote Team Member

by
Mike Cohn
Tagged:
product backlog, daily scrum, scrum master, taskboard, distributed teams
[Comments](#)

I want to address a question I was emailed yesterday but that I receive frequently. What should we do in the case when the team really likes having a [task board](#), but when one team member is remote? My first answer is always: Try to get the one person to move to where the rest of the team is.

I don't expect to see any moving trucks roll out when I ask this, but I have to ask. I figured if I keep asking, some team somewhere will eventually have the person move. Having one remote is a cost that must be borne by the full team. For the right person, it's easily worth it.

But sometimes, the person who is remote has not special skills, knowledge or experience to justify the added hassle. So, assuming that we've tried the "move here" solution and it didn't work, what can a team do that likes the physical task board but that has one remote member? My recommendation is to continue to use the physical task board -- it is simply too beneficial to the collocated team members to give it up in favor of a [product backlog tool](#), especially if team members are already used to it and like it.

How the team conveys information on the task board to the remote team member depends on what that person does. Sometimes the remote person works remotely because they have a very specialized skill that couldn't be filled in the office where the rest of the team is. This would be the case, for example, if our remote person is an expert in Windows internals and is writing boot-time code in C++. It would also be the case if our remote person has twenty years of experience in the domain and with some of our really old code.

In these cases, the remote person usually (not always) works for a day or two relatively alone. The remote person in this case could identify during the daily Scrum what he or she will work on and not need further interaction with the task board until selecting the next task.

Here, the remote person doesn't really interact with the task board at all and interacts only with the team. Not ideal as I'd like the person to see the tasks, but this can work in some situations.

What is more common is for the ScrumMaster to take on the responsibility of updating an electronic version that mirrors the physical task board. A shared spreadsheet is normally sufficient for this, but some teams opt for a more specialized tool, which is fine. Many tools offer 5-10 user free licenses and since the tool is only needed by the ScrumMaster and one remote employee, the free license is adequate.

Normally the ScrumMaster updates the electronic task board once a day, usually right after the daily scrum meeting. Of course, reading the physical task board and updating the electronic one can be quite time-consuming because the ScrumMaster has to look at each task in both places to see if that item needs to be updated.

One good way of minimizing the time the ScrumMaster spends doing this is to mark the cards on the physical task board. The mark indicates "I've updated this task. Please update it in the online task board." I like to use [Post-It flags](#) for this. As team members do their daily scrum, they stick one of these flags (of any color) on the card. If the estimate changes, if the task is done, if a new task is added, those cards are tagged with a flag. When the meeting is over, the ScrumMaster can very easily see which items need to be updated.

The ScrumMaster removes the flags once the online task board is updated. This approach also works in situations where the ScrumMaster updates the board more than once a day. Any time someone changes the board, flag the task.

Other teams do something similar using [color-coded dots](#). Anything touched on Monday is blue, Tuesday is red, and so on. Two problems occur though: First, the dot packs usually come with four colors to a pack so you have to buy a fifth color. Second, it's a hassle to make sure we have the right color on hand.

[Watch Now](#)

Posted: May 31, 2009

Tagged: product backlog, daily scrum, scrum master, taskboard, distributed teams

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the *Better User Stories* video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

[Six Attributes of a Great ScrumMaster](#)

Scrum Masters exhibit a shared set of traits. Discover 6 in-demand qualities, plus a bonus ...

Jan 17, 2023 • 32 Comments [Read](#)

[What Are Agile Story Points?](#)

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

[Epics, Features and User Stories](#)

I've been getting more and more emails lately from people confused about the difference between ...

Nov 08, 2022 • 93 Comments [Read](#)

[From Project Manager to Scrum Master -3 Tips for Making the Transition](#)

What does a good project manager need to do to become a great Scrum Master?

Aug 23, 2022 [Read](#)

[Relationship between Definition of Done and Conditions of Satisfaction](#)

I'd like to clarify the relationship between two important concepts: a team's Definition of Done ...

Jun 28, 2022 • 38 Comments [Read](#)

[Four Reasons Agile Teams Estimate Product Backlog Item](#)

Estimating product backlog items provides benefits beyond predicting when a project will be finished.

May 17, 2022 [Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile

New to Agile and Scrum?

Scrum

User Stories

Planning Poker

Agile Software Development

Agile Project Management

Agile Presentations

Mountain Goat on YouTube

Agile Mentors Podcast

Elements of Agile

Assessment

Agile & Scrum Training

Online Certifications

Video Courses

In-Person Certifications

On-Site Courses

Training Schedule

Compare Courses

PDUs and SEUs

More...

About Us

Contact Us

Our Students

Blog

Books by Mike Cohn

Agile Tools

The Ideal Agile Workspace

by Mike Cohn 28 Comments Tagged: product backlog, product owner, teams, sprinting, project management, scrum master, taskboard, communication, sprint burndown chart, musing

As you may know, I am working on a new book, which will be called [Succeeding with Agile](#). I recently finished writing a chapter for it on the impact of the human resources and facilities groups on an organization that is transitioning to an [Agile project management](#) approach. While writing that chapter, I put together a list of all the things that I think should be visible within the ideal agile workspace:

Big Visible Charts. Alistair Cockburn coined the term “Big Visible Charts” to describe the charts that agile teams like to hang on their walls. One of the most common of these is the sprint burndown chart, showing the number of hours remaining as of each day of the current sprint. Charts like these provide a strong visual reminder of the current state of the project. What is shown on these charts will get the

attention of team members so display charts showing the most important information for that sprint.

Ron Jeffries suggests considering big visible charts showing the number of passing customer acceptance tests, the pass/fail status of tests by day, sprint and release burndown charts, number of new stories introduced to the product backlog per sprint, and more.

Additional feedback devices. In addition to big, visible charts, it is common for an agile team to use additional visual feedback devices in their workspace. One of the most common is a lava lamp that is turned on whenever the automated build is broken. I've also worked with teams that use flashing red traffic lights to indicate exceptional conditions such as an issue on a production server. Also popular are [ambient orbs](#) and Nabaztag rabbits, which are wireless programmable devices that can also be configured to change colors, speak messages, or wiggle their ears as a team desires. Devices like these make a workspace more dynamic, unobtrusively bringing into it information the team may find helpful.

Everyone on your team. Each person on the team should ideally be able to see each other person on the team. This absolutely includes the ScrumMaster and ideally includes the product owner. I do understand, however, that product owners often have responsibilities to other groups outside the development team and so may sit near them instead. Still, in an ideal world the product owner would be visible to everyone in the team workspace.

The sprint backlog. One of the best ways to ensure that everything necessary is completed in the sprint is to make the sprint backlog visible. The best way to do that is by displaying the sprint backlog on a wall, ideally in the form of a [task board](#). A task board is usually oriented in rows and columns with each row containing a particular user story and one index card or sticky note for each task involved in that story. Task cards are organized in columns, minimally including "To Do" "In Process," and "Done." In this way, team members are able to see work progressing across the task board during the sprint and all work to be done is visible at all times.

The product backlog. One problem with running an endless series of sprints is that each can feel disconnected or isolated from the whole of a planned released or related set of new capabilities. A good way to reduce the impact of this problem is by displaying the product backlog somewhere clearly visible. This can be as simple as keeping the shoebox full of user stories written on index cards on a table in the middle of the team's space. Even better, tack the index cards with those upcoming user stories on a wall where all can see them. This allows team members to see how the user stories they are working on in the current sprint relate to others that are coming soon.

At least one big white board. Every team needs at least one big whiteboard. Locating this in the team's common workspace encourages spontaneous meetings. One developer may start using the board to think through a problem; others may notice and offer to help.

Someplace quiet and private. As important as open communication is, there are times when someone needs some peace and quiet. Sometimes this is for something as simple as a private phone call. Other times it can be to think through a particularly challenging problem without being interrupted.

Food and drink. It's always a good idea to have food and drink available. These don't need to be fancy, and they don't even need to be provided by the organization. I've worked with plenty of teams that buy a small under-desk refrigerator and share the expense of buying water bottles or soda for it. Other teams buy a coffee machine, depending on team preferences. Some teams rotate bringing in snacks, both healthful and not.

A window. Windows are often a scarce commodity and are doled out to an organization's favored employees. One of the nice things about an open workspace is that windows are shared. Even if the view is only of our parking lot and can only be seen across three messy desks, at least I can see the window and some natural light.

It's unlikely that every one of these will be visible from your workspace, but the more of them visible, the better. Let me know what else you think should be visible from within the ideal agile workspace.

Download Scrum Master Guide

Get a free copy of Situational Scrum Mastering: Leading an Agile Team

[Get my free guide now!](#)

Posted: March 5, 2009

Tagged: product backlog, product owner, teams, sprinting, project management, scrum master, taskboard, communication, sprint burndown chart, musing

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

What Is Cross-Functional Collaboration in Agile?

Perhaps the most prevalent and persistent myth in agile is that a cross-functional team is one on ...

 Feb 14, 2023 • 49 Comments [Read](#)

Be a Great Product Owner: Six Things Teams and Scrum Masters Need

Learn six ways effective product owners ensure their teams' success.

Jan 31, 2023 • 16 Comments [Read](#)

Six Attributes of a Great ScrumMaster

Scrum Masters exhibit a shared set of traits. Discover 6 in-demand qualities, plus a bonus ...

Jan 17, 2023 • 32 Comments [Read](#)

What Happens When During a Sprint

Succeeding with Scrum is easier when you know when and why to conduct each of the Scrum events ...

Jan 03, 2023 • 34 Comments [Read](#)

What Are Agile Story Points?

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

Don't Equate Story Points to Hours

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • 39 Comments [Re](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile

New to Agile and Scrum?

Scrum

User Stories

Planning Poker

Agile Software Development

Agile Project Management

Agile Presentations

Mountain Goat on YouTube

Agile Mentors Podcast

Elements of Agile

Assessment

Agile & Scrum Training

Online Certifications

Video Courses

In-Person Certifications

On-Site Courses

Training Schedule

Compare Courses

PDUs and SEUs

More...

About Us

Contact Us

Our Students

Blog

Books by Mike Cohn

Agile Tools

Should a Team Swarm on to One Backlog Item at a Time?

by

Mike Tagged:

Cohn product backlog, teams, sprinting, backlog, teamwork, testing,

26 iterations, iterating

Comments

This week I want to address the question of whether a team should work on one product backlog item at a time or whether it's OK to work on multiple items. In general, a team should feel comfortable working on multiple product backlog items at the same time during an iteration.

A typical seven person team will plan between five and ten items into an iteration. They'll usually be closer to the low end of that range with a one- or two-week iteration and closer to the high end with a four-week iteration. Perhaps surprisingly, though, iteration length has less influence on the number of product backlog

items selected than you might think. It seems that teams with longer sprints simply have larger product backlog items. A seven-person team will rarely be at its most efficient when all team members swarm onto a single product backlog item. There's too much opportunity for them to get in each other's way for this to be a good long-term strategy. However, an entire team swarming onto a single product backlog item can be a very effective learning technique and one that I often encourage.

If you are part of a team that hasn't yet learned how all disciplines can work well together, limiting the entire team to one product backlog item in progress at a time is something you might want to try. This forces people to quickly learn ways to work in small batches so that work can, for example, be transferred from programming to testing in multiple tiny increments. Similarly, if you are on a team where each developer grabs a product backlog item and works independently on it throughout an iteration, a rule of "only one item in progress at a time" is a good way to experience the benefits of working in smaller batches.

So, while I think swarming in this way is an excellent technique to use sometimes, I don't think it should be the default way of working for very many teams. Some may benefit from it over the long term, but most will find that it introduces too many opportunities to be in someone else's way as they try to make progress. I consider it equivalent to a drill that a team may do to improve a skill--good to use for practice but not the way to do something forever.

Download Scrum Master Guide

Get a free copy of Situational Scrum Mastering: Leading an Agile Team

[Get my free guide now!](#)

Posted: October 20, 2008

Tagged: product backlog, teams, sprinting, backlog, teamwork, testing, iterations, iterating

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and*

Planning, and Succeeding with Agile as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

What Is Cross-Functional Collaboration in Agile?

Perhaps the most prevalent and persistent myth in agile is that a cross-functional team is one on ...

Feb 14, 2023 • 49 Comments [Read](#)

What Happens When During a Sprint

Succeeding with Scrum is easier when you know when and why to conduct each of the Scrum events ...

Jan 03, 2023 • 34 Comments [Read](#)

What Are Agile Story Points?

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

Don't Equate Story Points to Hours

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • 39 Comments [Read](#)

Epics, Features and User Stories

I've been getting more and more emails lately from people confused about the difference between ...

Nov 08, 2022 • 93 Comments [Read](#)

Relationship between Definition of Done and Conditions of Satisfaction

I'd like to clarify the relationship between two important concepts: a team's Definition of Done ...

Jun 28, 2022 • 38 Comments [Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile **Agile & Scrum Training** **More...**

New to Agile and Scrum? Online Certifications About Us

Scrum Video Courses Contact Us

User Stories In-Person Certifications Our Students

Planning Poker On-Site Courses Blog

Agile Software Development Training Schedule Books by Mike Cohn

Development Compare Courses Agile Tools

Agile Project Management PDUs and SEUs

Agile Presentations

Mountain Goat on YouTube

Agile Mentors Podcast

Elements of Agile

Assessment

by
Mike Tagged:
Cohn product backlog, user stories, sprinting, daily scrum, teamwork,
48 communication
Comments

I want to address a question that I got recently and that comes up every month or two. I was recently emailed the following:

Most of our teams complete 10 or more user stories per sprint. When answering the 3 questions in the daily Scrum, it was clear what each person was working on, but it wasn't as clear how each story was doing or when a story was in trouble. For example, if no one worked on a story, problems with that story aren't visible because no one mentions that story during the standup. What we've started doing is conducting the daily standup story-by-story rather than person-by-person. Now it's very clear how each story is progressing, but difficult to figure out what each person is doing. Some people work on multiple stories and others may not even speak up in a daily Scrum.

One possible solution to this common problem is that these teams are doing too many product backlog items per sprint. Based on data I analyzed on successfully finished sprints, I determined that a team should average around 1 to 1-1/2 user stories (product backlog items of any sort, really) per person per sprint. So, a six-person team should get somewhere around 6-9 user stories done per sprint. Teams doing shorter sprints (one or two weeks) should be at the low end of the range; teams doing three- or four-week sprints should be at the upper end. This means that coincidentally teams with longer sprints do slightly larger user stories in their sprints. I hate that the answer I got was around one user story per person in a sprint because that makes it sound like each person grabs one user story and works on it alone for the duration of the sprint.

Nothing could be further from the truth.

Expressing the number of items in terms of the number of people on the team is just an obvious way to state the result: more people, more user stories. I want to reiterate though that the result is not for each person to do one user story per sprint alone. Another solution to the problem above is to conduct the daily standup in front of a physical task board and have people point to whatever they are working on. I always prefer to do this whenever possible. I frequently ask the speaker to "Point to what you're working on." Assuming a well setup task board this will show me the user story that the task relates to. Another solution is to designate a "point person" for each user story the team plans to work on in the sprint. This person is responsible for knowing if the user story is moving along appropriately. The person is essentially a "story owner" but we're talking about 2-5 minutes a day of extra work. This isn't a heavyweight new responsibility. The person may or may not be the primary contributor on the story; it doesn't seem to matter.

Another possible solution could be to look at the sizes of the teams involved. I find the ideal team size is 5-7 people. Standard agile advice seems to be 5-9 is the right size. When possible I try to stay on the low end of the range. If the team is more than 9, it is easy to lose track of what people are doing. Finally, most teams do the daily standup per-person, but some encounter the problem described here and discuss things story-by-story. Not all teams need to do it the same. If you're in an organization with even a handful of teams, I would randomly split them and tell some to try person-by-person and others to try story-by-story for a full sprint. I'd get everyone together afterwards for a short cross-team retrospective and let people say how it went. Hopefully teams could hear the results of other teams and then make a good decision about what to do next.

[Get your first lesson now!](#)

Posted: September 27, 2008

Tagged: product backlog, user stories, sprinting, daily scrum, teamwork, communication

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the *Better User Stories* video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

User Stories: How to Create Story Maps

Story maps help to create a shared understanding of the product, visualize user needs, and elicit ...

Mar 14, 2023

[Read](#)

How to Run a Successful User Story Writing Workshop

What you need to know to conduct a story-writing workshop with your team.

Feb 28, 2023 • 3 Comments [Read](#)

What Happens When During a Sprint

Succeeding with Scrum is easier when you know when and why to conduct each of the Scrum events ...

Jan 03, 2023 • 34 Comments [Read](#)

What Are Agile Story Points?

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

Epics, Features and User Stories

I've been getting more and more emails lately from people confused about the difference between ...

Nov 08, 2022 • 93 Comments [Read](#)

Advantages of User Stories over Requirements and Use Cases

At the surface, user stories appear to have much in common with use cases and traditional ...

Oct 05, 2022 • 41 Comments [Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile Assessment		

Rolling Lookahead Planning

by Mike Cohn
13 Comments
Tagged:
product backlog, product owner, teams, backlog, velocity, agile projects, iterations, agile processes, iterating, articles
originally published in Iterations on 2008-07-09

Agile processes replace some of the upfront planning of a traditional process with a just-in-time approach in which plans are progressively refined. For the often-used example of a team working in isolation, progressive refinement usually takes the form of creating a high-level release plan and then planning each iteration as it starts. While this is often sufficient for the lone team working on a desert island, it will generally be inadequate on a project being built by multiple teams. When a team needs to integrate its work with the work of other teams, all teams involved will usually benefit from what is known as rolling lookahead planning.

A team doing rolling lookahead planning will begin its iteration planning meetings just like any other team. Usually this is by doing commitment-driven iteration planning, which takes most teams two to five hours to

plan a two- to four-week iteration. This is because commitment-driven iteration planning requires the team to determine which product backlog items it will commit to deliver during the iteration by carefully considering each. Product backlog items are discussed and considered in roughly the priority order established by the product owner or customer. The tasks necessary to complete a product backlog item are identified and estimated. When the team finishes discussing the work involved to complete a product backlog item, team members decide whether they can commit to completing it in the coming iteration.

Once the current iteration has been planned, the team remains in the iteration planning meeting and plans two more iterations. Wait, before you panic let me point out that while it often takes two to five hours to plan the first iteration, this is because the second and third iterations are planned at the user story or product backlog item level rather than with tasks as was done with commitment-driven iteration planning for the current iteration. Using data such as its historical average velocity, or a projection of future velocity if there's a good reason to deviate from history, the product owner and team identify the product backlog items that are likely to be developed in the next two or so iterations.

The team is not committing to deliver these more distant items in the same way it is committing to complete the product backlog items of the current iteration. Specific commitments will be made to those iterations when each rolls forward to become the current iteration. Until then, each is planned only at a high level and only by identifying the work likely to be done in each iteration.

A team does not perform rolling lookahead planning to identify design considerations for the current iteration, although that may occur and be a side-benefit. The chief purpose of rolling lookahead planning is the coordination of work to be done by multiple teams. As new iterations roll into the planning horizon, teams look ahead to the functionality likely to be developed in those iterations. In doing so, teams on a large project are likely to find dependencies upon one another.

For example, suppose both your team and mine are ready to plan the fourth iteration of what will be a seven-month project. We conduct separate meetings, committing to features to be developed in iteration four. We separately identify and estimate the tasks to be performed in that iteration. Before concluding our separate meetings, our teams each use historical velocities to forecast what they will do in iterations five and six.

After both iteration planning meetings are over, you tell me that your team has identified a dependency on my team. You need me to develop something and pass it along to you so that you can complete a particular product backlog item. I tell you that my team cannot do it during iteration four as we just finished planning that iteration. That's fine with you because you don't need it until iteration six as it just now rolled into view. We agree that my team will develop the requested functionality in iteration five, delivering it to you at the start of iteration six. By looking ahead two iterations we have successfully avoided the emergency of "my team needs this from you this iteration" that can otherwise be common on large agile projects.

Rolling lookahead planning, and specifically looking ahead two iterations, is a useful technique that belongs in the toolbox of any agile team working on a large project.

Get 200 Real Life User Stories Examples Written by Mike Cohn

See user stories Mike Cohn wrote as part of several real product backlogs.

First Name

Email Address

[Privacy Policy](#)

Posted: July 8, 2008

Tagged: product backlog, product owner, teams, backlog, velocity, agile projects, iterations, agile processes, iterating, articles

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

What Is Cross-Functional Collaboration in Agile?

Perhaps the most prevalent and persistent myth in agile is that a cross-functional team is one on ...

Feb 14, 2023 • 49 Comments [Read](#)

Be a Great Product Owner: Six Things Teams and Scrum Masters Need

Learn six ways effective product owners ensure their teams' success.

Jan 31, 2023 • 16 Comments [Read](#)

What Are Agile Story Points?

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

Don't Equate Story Points to Hours

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • 39 Comments [Read](#)

Epics, Features and User Stories

I've been getting more and more emails lately from people confused about the difference between ...

Nov 08, 2022 • 93 Comments [Read](#)

Advantages of User Stories over Requirements and Use Cases

At the surface, user stories appear to have much in common with use cases and traditional ...

Oct 05, 2022 • 41 Comments [Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile **Agile & Scrum Training** **More...**

New to Agile and Scrum? Online Certifications About Us

Scrum Video Courses Contact Us

User Stories In-Person Certifications Our Students

Planning Poker On-Site Courses Blog

Agile Software Development Training Schedule Books by Mike Cohn

Development Compare Courses Agile Tools

Agile Project Management PDUs and SEUs

Agile Presentations

Mountain Goat on YouTube

Agile Mentors Podcast

Elements of Agile

Assessment

Visualizing a Large Product Backlog With a Treemap

by
Mike Cohn
Tagged:
product backlog, product owner, story points, project management, release planning, large teams
[Comments](#)

In the early days we promoted [agile project management](#) only for small teams because that was where it originated. We had plenty of experience to say that agile worked well on seven- to 10-person teams. We were also quick to learn the techniques that allowed agile project management methodologies to scale up to around 20 to 40 people. These days, though, there are many truly large projects being done with agile methodologies.

In preparing for this posting I started counting on my fingers the number of 100+ person projects I'm aware

of. I quickly ran out of fingers. I've been involved in a couple of 500+ person projects and am aware of three projects that each have over 1,000 people. We are truly at the point of doing large scale agile. Unfortunately, the charts and tools we use for such large projects have not entirely caught up with us.

For example, the time-tested Scrum burndown chart is absolutely wonderful but is really limited to showing only one thing: a single team's rate of progress through their [product backlog](#). This month I want to write about visualizing large product backlogs using a technique I've been advocating with my clients for three years but will be writing about here for the first time. Suppose you have a very large product backlog--such as one with lots of themes (groups of related user stories) or being worked on by multiple agile teams. The best way to visualize this product backlog is with a *treemap*. Treemaps were invented in the 1990s by Dr. Ben Shneiderman as a way of visualizing hierarchical (that is, tree-structured) data. The following is a simple treemap of a two-story house:

[treemap of two floors of a house](#)

In this treemap, each rectangle is sized to represent the relative area of the room. From it you can see that the master bedroom is about twice the size of either kid's room and a little larger than the downstairs family room. The combined green area of the downstairs rooms is slightly larger than the area of blue upstairs rooms. From this very simple treemap you can get a feel of certain aspects of this house. To visualize a product backlog with a treemap we need to conceptualize it as hierarchical data. We can do this a couple of different ways. For example, Rental Car Theme

As a traveler, I can rent a car
As a traveler, I can get collision insurance on a car rental policy.
As a traveler, I can request a baby car seat be inside my car.

Airplane Theme

As a traveler, I can request an aisle seat.
As a frequent flyer, I can request an upgrade to first class.

Or we could create a product backlog as a tree with levels for

Team

Product Backlog Items Being Worked On By That Team

The following figure shows a product backlog as a treemap. There are five themes in the treemap. (Theme 4 is in the top left; Theme 1 is in the bottom right. You can click on it to enlarge it but be sure to come back.) Each theme is made up of a number of individual user stories. Story 4-28 (indicating the 28th story of theme 4) is in the top left. I've used this "theme-dash-story" notation for simplicity only for this example. I wouldn't do that on a real project, of course.

[a treemap](#)

The size of each story in the preceding figure represents the number of story points that the story was assigned when estimated. Colors can be used on a treemap to represent an additional attribute of the data. On agile projects I've used colors to represent whether a user story has been developed or not. One color coding we used was:

1. Done
2. Started but not done
3. Not started but not planned to have been started yet
4. Not started but it should have been started by now
5. Blocked

I've also used color to indicate which team would work on a user story / product backlog item or whether the item was ready for development or not. Treemaps are very flexible in this regard. Check out the link to the stock market as a treemap at the end to see a great example of using color. A good treemap is interactive--you can mouse over, click on regions of it, and so on. For example, in the treemap above you'll notice that some of the user stories of Theme 4 are so small you can't read them. Clicking on a part of Theme 4 should zoom the treemap in so it displays only Theme 4, meaning more room is available and more detail can be shown as below:

Zoomed in on Theme 4

Sometimes zooming isn't enough. A good treemap implementation will also display additional detail when mousing over part of it as shown in this example:

[A treemap with an active popup](#)

Treemaps are an excellent way of visualizing large product backlogs. To date, I've made use of various open source or relatively inexpensive general treemapping tools to create these on my projects or for my clients. Hopefully now some of the leading agile tool vendors begin to incorporate this type of visualization technique into their products. I would love to be able to visualize a large product backlog and interact with directly from within some of those tools. Until then, check out some of the links below for useful ways to create treemaps. The ones in this posting were created with IBM's free Many Eyes tool.

[Get Your Free Scrum Cheat Sheet](#)

Sign up below to receive this FREE reference.

First Name

Email Address

[Privacy Policy](#)

Posted: July 2, 2008

Tagged: product backlog, product owner, story points, project management, release planning, large teams

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

Be a Great Product Owner: Six Things Teams and Scrum Masters Need Learn six ways effective product owners ensure their teams' success. <small>Jan 31, 2023 • 16 Comments Read</small>	What Are Agile Story Points? Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ... <small>Dec 06, 2022 • 122 Comments Read</small>	Don't Equate Story Points to Hours I've been quite adamant lately that story points are about time, specifically effort. But that does ... <small>Nov 22, 2022 • 39 Comments Read</small>	Epics, Features and User Stories I've been getting more and more emails lately from people confused about the difference between ... <small>Nov 08, 2022 • 93 Comments Read</small>	From Project Manager to Scrum Master -3 Tips for Making the Transition What does a good project manager need to do to become a great Scrum Master? <small>Aug 23, 2022 • 10 Comments Read</small>	Relationship between Definition of Done and Conditions of Satisfaction I'd like to clarify the relationship between two important concepts: a team's Definition of Done ... <small>Jun 28, 2022 • 38 Comments Read</small>
---	---	---	---	---	--

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

How To Fail With Agile: Twenty Tips to Help You Avoid Success

by

Mike Cohn Tagged:

Cohn product backlog, product owner, teams, sprinting, backlog, & meetings, daily scrum, teamwork, management, customers, agile Clinton projects, scrum master, agile processes, agile practices, articles, Keith agile books
44 originally published in Better Software on 2008-07-01

Comments

The [agile process](#) is now accepted as valid alternatives to traditional software development processes. Most people who adopt agile do so to realize the benefits of faster delivery, higher quality, products that more closely match user needs, and so on.

Not everyone is so enamored of agile. Some teams and individuals balk when a mandate to “become agile” is passed down from some “higher-up” in the organization or when some young go-getter decides to start an idealistic grassroots movement to effect change. A switch to agile often conflicts with personal goals such as

maintaining the status quo, avoiding career risk, working no harder than necessary, or maintaining a large fiefdom of direct reports.

It is to these individuals—those who have to become agile but don't want to—that we would like to direct our advice. Don't worry. We're not going to try to seduce you into trying agile, convince you of its merits, or tell you how to succeed. No, we're going to help you ensure agile failure. Then you'll be done with it and can go back to your comfort zone.

Although there are many ways to sabotage your agile project, for convenience we have grouped them into four categories: management issues, team issues, product owner issues, and process issues. In each instance, we will cite an example of someone who successfully caused agile failure, list the general guidelines for failure that the example is meant to demonstrate, and then list alternative techniques you can try to help you replicate the process. We hope this approach will allow you to fail quickly and avoid potential success.

Management Issues

Drew had seen management fads come and go. In his mind, agile was no different. A quick learner, he read a number of books and even took a class on agile. He didn't trust it, but, as a team player, it was his obligation to give it a try.

Drew picked team members and told them to "be agile." He told them that they would need to meet daily, estimate their work, and produce versions of their product (a database tool for storing artwork) every month.

Since Drew didn't trust agile or his team's ability, he attended every daily scrum, paying close attention and pointing out what the team was doing right and what it was doing wrong. Soon the daily meetings became a model of brevity and procedural correctness. As a bonus, no one spoke up about problems—especially in front of Drew. Drew had successfully followed our first guideline to agile failure.

GUIDELINE 1: Don't trust the team or agile. Micromanage both your team members and the process.
To no one's surprise, the team did not produce impressive results. It didn't meet all of its iteration goals and was no more productive than it had been before. Drew conducted retrospectives that did not reveal any problems that he could fix. As a result, Drew threw away all the books he had read and directed the team to return to the old way of developing the project. Drew was following guideline 2.

GUIDELINE 2: If agile isn't a silver bullet, blame agile.

While Drew went to one extreme by micromanaging his team, it is equally effective to go to the opposite extreme and not provide any guidance at all. Remember: While self-organizing agile teams are also self-managing, they are not self-leading. An agile team needs the type of leadership that provides a vision to work toward and motivation for achieving that vision. A strong agile leader, often in the form of a product owner, knows how to motivate a team with a description of an extremely desirable product that is just beyond what the team may think it can do. Freed to pursue that goal and provided with ongoing guidance from a product owner, an agile team can become truly high performing. Don't give your team that opportunity! If micromanagement isn't your style, follow guideline 3.

GUIDELINE 3: Equate self-managing with self-leading and provide no direction to the team whatsoever.

While support for using agile may come from the highest levels of a company, often the adoption of agile will

be driven by the agile team itself. Don't worry. You still have plenty of opportunities to create failure in those cases, especially if you are the manager. You may want to start by undermining the evangelist on the team—the one who has read all the agile books and is taking the chance to promote agile. Brush off the rules he is asking you to follow. Interrupt the daily scrum with new directions. Change the priority of the iteration goals. It works well and is encapsulated in guideline 4.

GUIDELINE 4: *Ignore the agile practices.*

They don't apply to management. If you want to be sure that agile doesn't take root, go straight to the agile team members themselves and let them know you think agile is a fad. Some of them will be skeptical to begin with, so it won't take much to convince them to ignore the practices. Remember, like Barney Fife, you have the power to nip this thing in the bud. Just follow guideline 5.

GUIDELINE 5: *Undermine the team's belief in agile.*

Team Issues

Not all of us are managers. Don't worry, non-managers can wreak havoc at the team level, as well. Just take the case of the NotQuite agile team, tasked with developing inventory-management software. This team shows the power of consistency in bringing down an agile project. For its first iteration, NotQuite committed to completing six items from the product backlog; it finished four. Because it was the first iteration and most teams overcommit in their first iteration, the product owner cut the team a little slack. This didn't faze the NotQuite team.

For the second iteration the team again planned to finish six items; it finished five. The slight improvement only lulled the product owner into a false sense of security. NotQuite continued to chronically overcommit, falling short in the third, fourth, and fifth iterations. Soon, the product owner learned not to trust the team, and this undermined any success it may have had with agile—a fantastic implementation of guideline 6 for agile failure.

GUIDELINE 6: *Continually fail to deliver what you committed to deliver during iteration planning.*

When falling short, don't make the mistake of going all-out on every iteration, reaching the last day panting with exhaustion time after time. A team like that could almost be forgiven for never quite delivering what it planned. A key to NotQuite's failure was its cavalier attitude toward missed commitments. Team members made it clear that it really didn't matter if something was finished on the last day of the iteration (as had been committed) or a few days into the next iteration. What's a few days between friends? Remember, a few days here and there can add up to quite a lot. If an agile team continually misses its commitments, it makes it impossible for the product owner to make plans and external commitments. This leads to guideline 7 for how to fail at agile.

GUIDELINE 7: *Cavalierly move work forward from one iteration to the next. It's good to keep the product owner guessing about what will be delivered.*

Perhaps the best way to cause an agile project to fail is to follow guideline 8.

GUIDELINE 8: *Do not create cross-functional teams. Put all the testers on one team, all the programmers on another, and so on.*

Merrilynn was able to use this guideline to kill her company's pilot agile project. Her organization was developing an application that would have separate Windows and Web-based clients. As a development director, Merrilynn had control over team composition and was able to create three separate teams: a

Windows team, a Web team, and a test team. This team structure worked against the goals of agile. If Merrilynn had wanted to succeed, she would have instead created three teams that each included Windows, Web, and test skills. Because Merrilynn kept the teams separate, she made it impossible for any team to deliver the working software that an agile team is expected to deliver at the end of each iteration. Nicely done, Merrilynn.

Another option open to Merrilynn was putting all twenty of her people on one team. This would have violated the standard agile advice of creating teams of five to nine people. She could have justified it to anyone who questioned the decision by stressing the unique two-client nature of her team's product. If she had chosen to create one large team instead of three reasonably sized teams, Merrilynn would have substantially increased the communication overhead of her team. This would have slowed progress and created complaints about how all the conversations in agile were a tremendous burden. If separating teams is too hard to justify, you can bog down a project very easily by following guideline 9.

GUIDELINE 9: *Large projects need large teams. Ignore studies that show productivity decreases with large teams due to increased communication overhead. Since everyone needs to know everything, invite all fifty people to the daily standup.*

Product Owner Issues

As you can see, agile failure at the product owner level is easily achieved through miscommunication, general ignorance of the team's progress, and lack of education.

If the management and team guidelines aren't available to you, there is another route to take: Consider a takedown from the product owner angle. A product owner has many options at her disposal to bring an agile project to its knees.

Take Kathy, for example, who was the product owner for a team working on a video game. The team was making great progress on features with every iteration and showing more player "fun" every time. Kathy let team members keep thinking that this was all they needed to do. She never attended reviews, rarely tried the game, and requested stories that were meant to steer the game toward the product she imagined. If that weren't enough, Kathy didn't share her own vision with the team or the other customers to whom she reported (such as marketing).

A year into development, the game was demonstrated to a group of executives who were shocked at the direction the game had taken. It was not what they wanted to market. The disconnect between Kathy, the team, and senior management caused the project to lose six months of progress. Well done, Kathy!

Kathy demonstrated several guidelines for how an agile project can fail at the hands of a product owner.

GUIDELINE 10: *Don't communicate a vision for the product to the agile team or to the other stakeholders.*

GUIDELINE 11: *Don't pay attention to the progress of each iteration and objectively evaluate the value of that progress.*

GUIDELINE 12: *Replace a plan document with a plan "in your head" that only you know.*

One of the tenets of iterative development is the discovery of the value of features being added as part of the

whole. This is the reason that every iteration produces a potentially shippable release of the product. This is in stark contrast to plan-driven projects, which attempt to predict the utilization of resources so the product emerges complete from all the separate parts only at the end. When a product owner does not make that change, the agile team can quickly fail. It is just as critical to educate the product owner as it is to educate the team. Generally speaking, if you want to ensure agile failure quickly, avoid training at all.

The crucial role of product owner often is balanced by someone else who acts as the team's ScrumMaster or coach. On many successful projects, a certain amount of naturally occurring tension exists between product owner and ScrumMaster. A product owner always desires more, more, more features. The coach, by contrast, is responsible for monitoring the health of the team. If the team is being pushed too hard and is beginning to get sloppy due to fatigue, the coach pushes back against the product owner's desires for more. A good way to fail at agile is to eliminate this push-pull tension between the coach and product owner by following guideline 13.

GUIDELINE 13: *Have one person share the roles of ScrumMaster (agile coach) and product owner. In fact, have this person also be an individual contributor on the team.*

As you can see, agile failure at the product owner level is easily achieved through miscommunication, general ignorance of the team's progress, and lack of education. You can compound that, if necessary, by having one person act in roles that are designed to balance each other. Following these guidelines to the letter is a great way to fail.

Process Issues

Rather than align pay, incentives, job titles, promotions, and recognition with agile, create incentives for individuals to undermine teamwork and shared responsibility.

If all else succeeds, careful misapplication of process issues can bring down almost any agile project. Jon is a terrific example of a process nightmare, and he did most of his best sabotage without even knowing he was doing it. Jon was the lead developer for a Chicago-based team developing software designed to approve or reject loan applications. In addition to being the lead developer, he was also the ScrumMaster (note how he began by embracing guideline 13, which by itself can wreak havoc).

Jon and his team were new to agile and were anxious to get rid of its unneeded parts. They immediately dispensed with daily standup meetings, reasoning that since the team sat in the same general area, most conversations could be heard over the six-foot-high cubicle walls.

They also decided that having automated unit tests was unnecessary. Since theirs was a new application, there was no chance of breaking old code, and since all new code would be fresh in everyone's minds there would be little chance of accidentally breaking it. However, Jon and his coworkers did embrace refactoring and collective code ownership.

Their new rule was that any programmer could change the code of any other programmer at any time. They soon learned that refactoring and collective code ownership can be very dangerous without the safety net of automated unit tests to make sure you aren't breaking things while improving them. Jon and his team had unwittingly stumbled on these two guidelines for causing agile failure.

GUIDELINE 14: *Start customizing an agile process before you've done it by the book.*

GUIDELINE 15: Drop and customize important agile practices before fully understanding them.

An alternative to these guidelines is to dive into the practices without understanding why you're doing them. As coaches, we encounter many teams who have learned a technique or been told to do something by someone and who then continued to do it even when they'd outgrown the technique (a subtle, yet effective, subterfuge). This brings to mind the story of the newlywed wife who cuts a quarter inch off both ends of every roast she cooks.

When her husband asks why she's trimming the roast that way, she has no ready answer; she does it that way because it's the way her mother always did it. Curious as to her mother's rationale, the wife calls her mother and asks why she taught her to cut the ends of the roast. Her mother says she only does it that way because her own mother taught her to do so. The young wife next calls her grandmother and asks why she cut a quarter inch off the end of every roast. Her grandmother tells her, "Because my roasting pan was too small. The roast wouldn't fit any other way." We capture this as our next guideline for agile failure.

GUIDELINE 16: Slavishly follow agile practices without understanding their underlying principles.

If you haven't been able to implement guidelines 14 through 16 and your agile project is succeeding despite your best efforts, you can bring even a successful project to a halt simply by changing nothing. What, you say? Change nothing? Follow the example of the StatusQ team. StatusQ, assigned to build a new Web-based reservation system, got off to a good start. Team members were new to agile but did a good level of research and sent a few of their people off to become certified ScrumMasters.

The project quickly benefited from the new practices. In a month, StatusQ had a simple Web site up and running and was able to demonstrate a few key interfaces that gave its customers a lot of confidence that their vision of an accessible and powerful reservation system would work.

StatusQ never held a retrospective at the end of each sprint. The ScrumMaster didn't push for it because he didn't see the need. Everything was already working wonderfully well. You can encourage this behavior on your own team by complaining loudly to your ScrumMaster and team members if they try to hold a retrospective. Tell them that it's a waste of time to sit and talk about a project that's going well. Tell them that retrospectives only make sense when things are going wrong.

Over time, things at StatusQ started to slow down. The rate of change and the growing code base were creating maintenance problems. Changes to the system from the customers were supposed to be a benefit to them, but the code couldn't keep up. Code stability became so bad that the project seemed to be moving backward. Finally company management stepped in, put a halt to the changes, and finished the contract at half price.

This team had unknowingly demonstrated our next two guidelines for agile failure.

GUIDELINE 17: Don't continually improve.**GUIDELINE 18: Don't change the technical practices.**

Company process issues that at first seem unrelated to the project can have a negative impact on morale and motivation. Consider the case of Dave, an up-and-coming artist who worked for a mobile phone game developer. He welcomed his company's adoption of agile, as it made a lot of sense to him. The new agile teams consisted of programmers, artists, designers, and a number of other people from numerous disciplines. Everyone relied on each other to create iterations of their game. If the artists did a good job, then the entire team looked good. Dave often dropped what he was doing to help his teammates iterate on the art to improve the game. This often came at the expense of his own work, but, for Dave, team goals came first.

Dave's team did a great job and produced a hit title that sold many thousands of copies and earned the company substantial profits.

At the end of the year, Dave had his first performance review with the company's lead artist. Dave was shocked to learn that his yearly bonus was small. Dave was told that he was judged by the senior artist in the company to have missed his art production goals. As it turns out, the senior artist was counting the number of iteration task cards that Dave completed and based his judgment on that rather than on the real amount of work Dave completed.

Chagrined, Dave returned to his team vowing to make sure his task cards took priority over the needs of the team. Guideline 19 can be your backup plan for any enthusiastic agilists in your company.

GUIDELINE 19: *Rather than align pay, incentives, job titles, promotions, and recognition with agile, create incentives for individuals to undermine teamwork and shared responsibility.*

A tenet shared by all agile processes is that work is prioritized based on the value provided by each feature. Other factors, such as risk and knowledge creation, are considered, but the amount of value delivered remains the dominant factor. A sure fire way to ensure agile failure is to ignore this tenet and instead follow guideline 20.

GUIDELINE 20: *Convince yourself that you'll be able to do all requested work, so the order of your work doesn't matter.*

There are, of course, other ways to fail in addition to those collected here. In your effort to undermine a successful agile project, you may have already discovered some on your own. Of course, you are probably keeping quiet about them because it's critical that the sabotage not be detected until the bridge is blown. We are confident that, through diligent application of the guidelines here or those that only you know—or both—you will be able to plot the downfall of your next agile project and ensure agile failure.

Get Your Free Scrum Cheat Sheet

Sign up below to receive this FREE reference.

First Name

Email Address

[Privacy Policy](#)

Posted: June 30, 2008

Tagged: product backlog, product owner, teams, sprinting, backlog, meetings, daily scrum, teamwork, management, customers

About the Author

Over the course of twenty years, Clinton Keith has gone from programming avionics for advanced fighter jets and underwater robots to overseeing programming for hit video game titles such as "Midtown Madness," "Midnight Club," and "The Bourne Conspiracy." He introduced the video game industry to agile development is now helping creative teams adopt agile. Clinton is the author of "Agile Game Development with Scrum." His web site is <http://www.clintonkeith.com>

Mike Cohn, the founder of Mountain Goat Software, a process and project management consultancy that specializes in helping companies adopt and improve their use of Agile processes and techniques. He is the author of User Stories Applied for Agile Software Development, Agile Estimating and Planning, and Succeeding with Agile. Mike is a co-founder of the Agile Alliance. He is also a co-founder and current board member of the Scrum Alliance. He can be reached at mike@mountaingoatsoftware.com

You may also be interested in:

[What Is Cross-Functional Collaboration in Agile?](#)

Perhaps the most prevalent and persistent myth in agile is that a cross-functional team is one on ...

Feb 14, 2023 • 49 Comments [Read](#)

[Be a Great Product Owner: Six Things Teams and Scrum Masters Need](#)

Learn six ways effective product owners ensure their teams' success.

Jan 31, 2023 • 16 Comments [Read](#)

[Six Attributes of a Great ScrumMaster](#)

Scrum Masters exhibit a shared set of traits. Discover 6 in-demand qualities, plus a bonus ...

Jan 17, 2023 • 32 Comments [Read](#)

[What Happens When During a Sprint](#)

Succeeding with Scrum is easier when you know when and why to conduct each of the Scrum events ...

Jan 03, 2023 • 34 Comments [Read](#)

[What Are Agile Story Points?](#)

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

[Don't Equate Story Points to Hours](#)

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • 39 Comments [Re:](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Sorry, the browser you are using is not currently supported. Disqus actively supports the following browsers:

- [Firefox](#)
- [Chrome](#)
- [Internet Explorer 11+](#)
- [Safari](#)

[Skip Pletcher](#) • 9 years ago

because no list should ever be simple,

GUIDELINE 21. Use flexible scheduling to your advantage.

Make sure some people arrive and leave early while others arrive and leave late. Arrange work schedules so that the chance of one person being in the building at the same time as another team member is 50:50. This minimizes the available 'core time' available for collaboration, assuring that mismatched assumptions creep into the product.

GUIDELINE 22. Maintain discipline integrity.

If someone joins the team from business analysis, make sure that person is never asked to do anything except requirements. Likewise with developers or testers. We may be cross-functional as a group, but we don't have to help each other as a team. No one should be expected to do anything she doesn't have years of experience perfecting; no one should ever be asked to share any of his skills or techniques with someone from another

Learn About Agile **Agile & Scrum Training** **More...**

New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

Improving On Traditional Release Burndown Charts

by
Mike Cohn Tagged:
product backlog, user stories, product owner, story points,
velocity, metrics, release planning, sprint burndown chart
[Comments](#)

I want to use this month's blog posting to introduce a type of burndown (and burnup) chart that I find useful. I've been drawing this style of burndown chart for years and have coached many of my clients to do the same. Unfortunately, we've had to draw it either by hand or in tools like Visio and OmniGraffle because the agile tool vendors haven't (to my knowledge) hit on this idea yet. I'm hopeful that some of them will see this posting, decide this is a good visualization, and incorporate it into their products. The classic Scrum release burndown chart is good at showing whether a team will finish "on time" as can be seen in the following example burndown chart:

A release burndown chart such as this one shows sprints on the horizontal axis and can show story points or ideal days on the vertical. It is updated once per sprint to show the team's net progress that sprint. A team's net progress is the amount of work they finished net of any changes in scope. So a team that completes 30 points of work but that has 10 points added to their product backlog will show net progress of 20. But while a traditional release burndown chart excels at showing whether a team is on pace to finishing on time, it is not very good at showing what will be included in that "on time" delivery.

To see this, imagine two teams that each start with 200 story points of work. The first team finishes twenty points of work for each of ten sprints. The second team is incompetent and rather than completing twenty points each sprint they drop twenty points of scope each sprint. The two burndown charts will be identical--perfect lines descending over ten sprints from 200 to 0. At one level this is OK: the burndown chart shows whether a team will be finished (or by when the will be finished). The simplicity of the standard release burndown chart has much in its favor. It isn't hard, though, to extend a release burndown chart to also show what will be in the product by the final sprint. Look at the next figure, which is a hypothetical example of an eCommerce product.

In this figure, you can see the burndown is tracked in the normal way through the end of the current sprint, the seventh. The company desires to release this product after the fourteenth two-week sprint. The right side of the burndown chart shows the team's product backlog with the highest priority theme ("Returns") at the top. This top block represents some "must-have" user stories related to returning purchased items. Below that is a theme for gift wrapping purchased items, followed by some "nice to have" aspects of returning items.

At the bottom right is the Coupons theme. Extending out from the team's current position at the end of the seventh sprint are four lines. These lines represent the following:

1. The team's current position, drawn as a horizontal line from the current burndown position over to the product backlog. This tells us what is in the product so far. We can see that the mandatory return user stories and the gift wrap user stories are finished and that the team is partially into the nice-to-have return user stories.
2. A black, dashed line showing the team's most likely finish. This is the first of three trend lines meant to show the likely range of work the team might deliver. To draw a team's most likely finish use the team's long-term average velocity. You can define "long-term average velocity" in whatever way you want but my preference is to use the average velocity of the last 8-12 sprints. Pick the number of historical sprints that is most suitable for your team based on how long the sprints are and how long the team stays together.
3. A pessimistic forecast of the amount of functionality that may be delivered. I recommend forecasting this based on a team's worst-case but likely velocity. Calculate this by averaging the worst three or so velocities chosen among the same 8-12 iterations you looked back at to determine the team's long-term average velocity.
4. An optimistic forecast of the amount of functionality that may be delivered. Calculate this in the same was as in the pessimistic case but use the three (or so) best velocities of the team.

The figures in this blog are static images I've cut from a presentation. If you apply this technique for your team, the backlog items on the right should be clickable, allowing users to drill down into a product backlog theme to see specifically which items (typically user stories) make up the backlog. By producing a single chart that shows both a team's rate of progress (its burndown) and the product backlog, we have a single visualization that shows both when a team is likely to finish and what features will be in the product by that time. This makes it easier for product owners to make scope vs. schedule tradeoff decisions. Check back in a few weeks when I'll show an even more powerful technique for visualizing large product backlogs.

Get Your Free Scrum Cheat Sheet

Sign up below to receive this FREE reference.

First Name

Email Address

[Privacy Policy](#)

Posted: June 17, 2008

Tagged: product backlog, user stories, product owner, story points, velocity, metrics, release planning, sprint
burndown chart

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of User Stories Applied for Agile Software Development, Agile Estimating and Planning, and Succeeding with Agile as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

User Stories: How to Create Story Maps

Story maps help to create a shared understanding of the product, visualize user needs, and elicit ...

Mar 14, 2023

[Read](#)

How to Run a Successful User Story Writing Workshop

What you need to know to conduct a story-writing workshop with your team.

Feb 28, 2023 • 3 Comments

[Read](#)

Be a Great Product Owner: Six Things Teams and Scrum Masters Need

Learn six ways effective product owners ensure their teams' success.

Jan 31, 2023 • 16 Comments

[Read](#)

[What Are Agile Story Points?](#)

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • [122 Comments](#)

[Read](#)

[Don't Equate Story Points to Hours](#)

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • [39 Comments](#)

[Read](#)

Epics, Features and User Stories

I've been getting more and more emails lately from people confused about the difference between ...

Nov 08, 2022 • [93 Comments](#)

[Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

Working with “Storyless Tasks”

by
Mike **Tagged:**
Cohn product backlog, story points, velocity, defect management,
32 taskboard
[Comments](#)

A question I get frequently is what to do with tasks that do not belong to a particular user story or product backlog item. Common examples I'm asked about include tasks like “update the build server to use the latest version of FitNesse.” Fixing bugs from prior iterations are another common example. I've sometimes heard of these tasks that don't belong to a particular product backlog item or user story referred to as “storyless tasks.”

If you've been to one of my classes, heard me speak at a conference, or just poked around my website you very likely know that I have a strong preference for [task boards](#) whenever possible. Usually this means whenever a team is collocated or a few special distributed teams who want to do it.

A taskboard will look something like this:

To answer the questions about storyless tasks, I want to answer in the context of using a task board. But the essentials of my answer will remain the same even if you are using a tool. Most teams find it useful to include two additional rows on their task boards:

1. Miscellaneous
2. Bugs

Each other row on those boards represents a user story and the cards are the tasks to implement the story. The Miscellaneous and Bugs rows are essentially the storyless tasks. The Miscellaneous row holds things like “update the build server.”

I do not normally recommend that a team estimate these items in story points such that they would earn any points toward velocity by completing these tasks. Tasks in the Miscellaneous row are usually tasks that enable the team to perform the other work (or to perform it better or faster). It is reasonably safe that over the long-term any story points you’d consider assigning here will average out. This is yet another reason why I always stress that velocity is a useful long-term predictor rather than a predictor of the short-term such as what a team may complete in exactly the next iteration. I often use a Bugs row on the task board for bugs that are either:

discovered during the iteration but unrelated to the stories of the current iteration (bugs related to stories already on the board go into the To Do column of the story’s row)
planned at the start of the iteration to be done during the iteration

If the bugs were planned into the iteration at its start, often a team does put a story point estimate on those bugs. Usually the bugs are estimated together--that is, “if we fix these 6 bugs we’ll get two story points.” Lots of teams with large defect backlogs plan to do something like bring in 5 points of bugs each iteration. So they estimate those a bit “backwards.” Rather than grab a stack of bugs and ask, “how many points do these 8 bugs represent in total” they keep adding bugs into the iteration until they feel they’ve brought in the desired number of points worth of bugs.

[Download my PDF](#)

Posted: May 21, 2008

Tagged: product backlog , story points , velocity, defect management , taskboard

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of User Stories Applied for Agile Software Development, Agile Estimating and Planning, and Succeeding with Agile as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

What Are Agile Story Points?

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

Don't Equate Story Points to Hours

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • 39 Comments [Read](#)

Epics, Features and User Stories

I've been getting more and more emails lately from people confused about the difference between ...

Nov 08, 2022 • 93 Comments [Read](#)

Relationship between Definition of Done and Conditions of Satisfaction

I'd like to clarify the relationship between two important concepts: a team's Definition of Done ...

Jun 28, 2022 • 38 Comments [Read](#)

Four Reasons Agile Teams Estimate Product Backlog Items

Estimating product backlog items provides benefits beyond predicting when a project will be finished.

May 17, 2022 [Read](#)

Estimating with Story Points Course Available for One Week

Registrations are now open to Estimating with Story Points.

Nov 17, 2021 [Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile **Agile & Scrum Training** **More...**

New to Agile and Scrum? Online Certifications About Us

Scrum Video Courses Contact Us

User Stories In-Person Certifications Our Students

Planning Poker On-Site Courses Blog

Agile Software Development Training Schedule Books by Mike Cohn

Development Compare Courses Agile Tools

Agile Project Management PDUs and SEUs

Agile Presentations

Mountain Goat on YouTube

Agile Mentors Podcast

Elements of Agile

Assessment

by
Mike Cohn
Tagged:
product backlog, user stories, product owner
184 Comments

In my [user stories book](#) and in all my training and conference sessions on [user stories](#) I advocate writing user stories in the form of:

"As a , I want so that ." While I consider the so-that clause optional, I really like this template. At a conference, someone asked me why. Because I get that question fairly often, I want to give three reasons why here:

Reason 1

Something significant and I'm tempted to say magical happens when requirements are put in the first person. Obviously by saying "As a such-and-such, I want ..." you can see how the person's mind goes instantly to imagining he or she is a such-and-such. As for the magic, Paul McCartney was interviewed and asked about why the Beatles songs were so amazingly popular. One of his responses was that their songs were among the first to use a lot of pronouns. Think about it: *She Loves You, I Wanna Hold Your Hand, I Saw Her Standing There, I Am The Walrus, Baby You Can Drive My Car*, etc. His point was that these helped people more closely identify with the songs.

You can read more about the [Beatles' use of pronouns](#) in this article.

Reason 2

Having a structure to the stories actually helps the product owner prioritize. If

the product backlog is a jumble of things like:

- Fix exception handing
- Let users make reservations
- Users want to see photos
- Show room size options

... and so on, the product owner has to work harder to understand what the feature is, who benefits from it, and what the value of it is.

Reason 3

I've heard an argument that writing stories with this template actually suppresses the information content of the story because there is so much boilerplate in the text. If you find that true, then correct it in how you present the story. I've seen backlogs in Word that present the boilerplate in grayed text with the unique parts in black. I've created product backlogs in Excel that use column headings to filter out the common text.

Look here, and you'll see what I mean:

AS A/AN	I WANT TO...	SO THAT...
moderator	create a new game by entering a name and an optional description	I can start inviting estimators
moderator	invite estimators by giving them a url where they can access the game	we can start the game
estimator	join a game by entering my name on the page I received the url for	I can participate
moderator	start a round by entering an item in a single multi-line text field	we can estimate it
estimator	see the item we're estimating	I know what I'm giving an estimate for
estimator	see all items we will try to estimate this session	I have a feel for the sizes of the various items
moderator	see all items we try to estimate this session	I can answer questions about the current story such as "does this include _____"
moderator	select an item to be estimated or re-estimated	the team sees that item and can estimate it

Seriously take a look at the table above before continuing ... I'll wait ... Please, really look at it before reading further.

OK, here's how I bet you read the spreadsheet. I bet that as you read most of the rows you added in the "As a ...," "I want ...," and "so that ..." text, possibly even by looking at the heading as you read across each row. I've experimented by asking people, and this is what most people do. If that text is unnecessary, why do we

mentally mouth the words to ourselves or even glance at the heading while reading the row? Perhaps it's not so non-essential after all.

For now, and probably a long time to come, I'll be sticking with the, "As a , I want so that " template for these and other reasons.

[Reserve a Spot](#)

Posted: April 25, 2008

Tagged: product backlog, user stories, product owner

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of User Stories Applied for Agile Software Development, Agile Estimating and Planning, and Succeeding with Agile as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

User Stories: How to Create Story Maps

Story maps help to create a shared understanding of the product, visualize user needs, and elicit ...

Mar 14, 2023

[Read](#)

How to Run a Successful User Story Writing Workshop

What you need to know to conduct a story-writing workshop with your team.

Feb 28, 2023 • 3 Comments [Read](#)

Be a Great Product Owner: Six Things Teams and Scrum Masters Need

Learn six ways effective product owners ensure their teams' success.

Jan 31, 2023 • 16 Comments [Read](#)

What Are Agile Story Points?

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

Epics, Features and User Stories

I've been getting more and more emails lately from people confused about the difference between ...

Nov 08, 2022 • 93 Comments [Read](#)

Advantages of User Stories over Requirements and Use Cases

At the surface, user stories appear to have much in common with use cases and traditional ...

Oct 05, 2022 • 41 Comments [Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile Assessment		

Prioritizing Tasks Within a Sprint

by
Mike Cohn
Tagged:
product backlog, user stories, product owner, sprint planning,
prioritizing
[Comments](#)

In the discussion that ensued [from another post here](#), Brian Bailey [asked a great set of questions](#). The questions were big enough that I've moved them here. I'm going to intersperse Brian's questions and comments with my responses. Brian started with:

If the product owner is onsite and part of the same company, what role should she have in prioritizing tasks within a sprint? To clarify, I assume that a 30-day sprint does not mean that new features that are completed are held until the end of the sprint. If a feature is finished and ready to move to production after the first week, it can be released, correct? If that's the case, should a PO be allowed to prioritize the stories and tasks the team committed to ("Please start with this

critical item") or should the team have full control over the order things are worked on?

First, the concept of prioritizing tasks within a sprint is one that doesn't make sense. When a team plans a sprint they make a commitment to complete the user stories they select from the product backlog. It's not a come-hell-or-high-water commitment, but the team is expected to do its best to complete the work they select. And averaging things out over the course of many sprints, the team should generally meet their commitment. The idea of prioritizing items within a commitment doesn't make sense.

Back in 1988 I got married and made a commitment to my wife, Laura. I promised to love, honor, and cherish her. (We mutually agreed to drop "obey.") I didn't promise to love her all the time, honor her if there was more time and then cherish her as a stretch goal.

But what about Brian's statement that, "I assume that a 30-day sprint does not mean that new features that are completed are held until the end of the sprint." Well, yes and no. This depends upon what was agreed to during sprint planning. The default assumption should be that the team has the right to rip the system apart on day one of the sprint and does not have to have it put back together until the last day. (For argument's sake, I'm ignoring that this would be a bad idea for many reasons.)

If the product owner needs an interim deliverable during the sprint, that should be discussed and planned during the sprint planning meeting. There will almost certainly be some overhead in doing this and the team needs to account for that. Even better, though, I'd suggest trying a shorter sprint length if this is common.

Back to Brian:

And of course, the dangerous follow-up that almost seems inevitable - could the PO then decide or shift priorities (not adding or replacing tasks, though) throughout the sprint? I'm confident this is a bad idea and could lead to micro-management instead of team-driven development. But I can also see how the PO might have a good reason for something to be moved to the front of the line.

You're correct. This is a bad idea. One of the challenges with 30-day sprints is that it is a long time to make the business go without changing its collective mind. If these questions are legitimate concerns then a shorter sprint length is a likely solution.

Get 200 Real Life User Stories Examples Written by Mike Cohn

See user stories Mike Cohn wrote as part of several real product backlogs.

First Name

Email Address

[Privacy Policy](#)

Posted: March 20, 2008

Tagged: product backlog, user stories, product owner, sprint planning, prioritizing

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

Why I Don't Emphasize Sprint Goals

Sprint goals are considered a mandatory part of Scrum. Here's why I disagree.

Mar 21, 2023

[Read](#)

User Stories: How to Create Story Maps

Story maps help to create a shared understanding of the product, visualize user needs, and elicit ...

Mar 14, 2023

[Read](#)

How to Run a Successful User Story Writing Workshop

What you need to know to conduct a story-writing workshop with your team.

Feb 28, 2023 • 3 Comments [Read](#)

Be a Great Product Owner: Six Things Teams and Scrum Masters Need

Learn six ways effective product owners ensure their teams' success.

Jan 31, 2023 • 16 Comments [Read](#)

What Are Agile Story Points?

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

Epics, Features and User Stories

I've been getting more and more emails lately from people confused about the difference between ...

Nov 08, 2022 • 93 Comments [Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

FEATURED POSTS

[Scrum Product Backlog](#)

[What Is a Product?](#)

[Product Backlog Refinement \(Grooming\)](#)

[Writing the Product Backlog Just in Time and Just Enough](#)

[Prioritizing Your Product Backlog](#)

[Why There Should Not Be a “Release Backlog”](#)

Get Your Free Scrum Cheat Sheet

Sign up below to receive this FREE reference.

First Name

Email Address

[Privacy Policy](#)

[Download my PDF](#)

MORE PRODUCT BACKLOG POSTS:

[Don't Average During Planning Poker](#)

While I want teams to come to agreement, I don't care how heartfelt the agreement is.

[To Re-estimate or not; that is the question](#)

When we estimate it is important that we not mix knowledge-before-the-fact with knowledge-after-the-fact.

[Programmers and Testers Can Work on Things Smaller Than User Stories](#)

A common misperception is that testers cannot do any work during an iteration until the programmers ...

[Enterprise and Scrum](#)

The two best things about this book are that it: (1) provides a framework for adopting Scrum across ...

[Sprint and release planning should be in different units](#)

In sprint planning the team should always talk of tasks and hours.

[Sprint Planning](#)

Many teams try to divide and conquer when it comes to sprint planning, often with disjointed and ...

[The Need for Agile Project Management](#)

Ken Schwaber and I co-wrote this article to help counter the misperception that agile projects do ...

[The Upside of Downsizing](#)

This article describes how a project was successfully downsized from 100 to 12 developers. To make ...

NEWER POSTS

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools

[PDUs and SEUs](#)

[Agile Presentations](#)

[Mountain Goat on YouTube](#)

[Agile Mentors Podcast](#)

[Elements of Agile](#)

[Assessment](#)

Should Companies Measure Productivity in Story Points / Ideal Days?

by

Mike Cohn

Tagged: product backlog, teams, sprinting, story points, backlog,

55 customers, metrics, lean

Comments

Using story points or ideal days to measure productivity is a bad idea because it will lead the team to gradually inflate the meaning of a point--when trying to decide between calling something “two points” or “three points” it is clear they will round up if they are being evaluated on productivity as measured by the number of story points (or ideal days) finished per iteration.

My view is that points can be used as the best way to estimate and assess progress that we've ever had or they can be used as another weapon with which to hit the team. There are plenty of weapons with which you can hit your team. We

don't need to ruin points by using them that way as well. Some teams have measured productivity with things like the number of backlog items delivered or the % of backlog items completed vs. planned into a sprint. Teams will alter their behavior on those as well though so they can be gamed and misleading. These metrics can be useful but only as part of a suite of metrics collected at the end of each iteration. If we rethink the question of "how do we measure productivity" we might get a better answer.

Suppose you own a sandwich shop and want to measure the productivity of the sandwich maker in the back. He responds to our metric by making as many sandwiches as he can--regardless of whether anyone ordered them! At the end of the day there will be 200 extra sandwiches to throw away.

A better measure of him might be how quickly he makes any sandwich. So we'd measure the time from when the customer placed the order until the sandwich is put on a tray. Or for a more complete metric we may want to measure the time from when he receives an order until he is ready to receive the next order as this captures any cleanup or restart time.

So, one measure we may want to include in our suite of metrics could be the responsiveness of the development organization. This would be measured in the same way as in the sandwich shop. Datestamp each product backlog item and track the time from when something enters the product backlog until it either (a) comes out of an iteration or (b) is delivered into the hands of customers. Choosing between (a) and (b) will largely be a matter of how often you ship software. Option (b) is a better measure of rapid delivery of customer value but is impractical in some cases. It would be a bit of a useless measure for the Microsoft Vista team, for example.

[Download my PDF](#)

Posted: December 11, 2007

Tagged: product backlog , teams , sprinting , story points , backlog , customers , metrics , lean

[About the Author](#)

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the *Better User Stories* video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

What Is Cross-Functional Collaboration in Agile?

Perhaps the most prevalent and persistent myth in agile is that a cross-functional team is one on ...

Feb 14, 2023 • 49 Comments [Read](#)

What Happens When During a Sprint

Succeeding with Scrum is easier when you know when and why to conduct each of the Scrum events ...

Jan 03, 2023 • 34 Comments [Read](#)

What Are Agile Story Points?

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • 122 Comments [Read](#)

Don't Equate Story Points to Hours

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • 39 Comments [Read](#)

Epics, Features and User Stories

I've been getting more and more emails lately from people confused about the difference between ...

Nov 08, 2022 • 93 Comments [Read](#)

Relationship between Definition of Done and Conditions of Satisfaction

I'd like to clarify the relationship between two important concepts: a team's Definition of Done ...

Jun 28, 2022 • 38 Comments [Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile **Agile & Scrum Training** **More...**

New to Agile and Scrum? Online Certifications About Us

Scrum Video Courses Contact Us

User Stories In-Person Certifications Our Students

Planning Poker On-Site Courses Blog

Agile Software Development Training Schedule Books by Mike Cohn

Development Compare Courses Agile Tools

Agile Project Management PDUs and SEUs

Agile Presentations

Mountain Goat on YouTube

Agile Mentors Podcast

Elements of Agile

Assessment

Learn About Agile

An Introduction to Agile and Scrum

The Scrum Framework

[Scrum Roles: An Overview](#)

Scrum Master Role and Responsibilities

[Product Owner Role and Responsibilities](#)

Scrum Team Role & Responsibilities

The Chicken and the Pig

[Scrum Activities: An Overview](#)

Sprint Planning Meeting

Daily Scrum Meeting

Sprint Review Meeting

Sprint Retrospective

[Scrum Tools: An Overview](#)

Scrum Product Backlog

Sprint Backlog

Scrum Task Board

Release Burndown Chart

[Free Scrum Resources](#)

Reusable Scrum Presentation

User Stories

Planning Poker

Agile Software Development

Agile Project Management

In [Scrum](#), the product owner is the person accountable for what will be built.

What is the Role of Product Owners in Scrum?

What is a Product Owner? The role of a product owner in Scrum is to [work with stakeholders](#) to create a vision of the product they wish to create and communicate that product vision to the [Scrum team](#) and stakeholders. It is one of the key roles in Scrum, along with the [Scrum Master](#) and the cross-functional product development team.

Product owners have many responsibilities/accountabilities. Inside the Scrum framework, one accountability of the product owner is creating and maintaining the [product backlog](#), which is an evolving list of desired features for the product, typically ordered by priority and often written as [user stories](#). Others can write stories, split stories, and [suggest new product backlog items](#). At the end of the day, however, the product owner's job is to manage the product backlog, ensuring it reflects the current understanding of what the end product should be.

One activity the product owner engages in, therefore, is to keep the product backlog in order and always be looking a few sprints ahead to [ensure the product backlog items are ready for the team](#) to bring into a future sprint.

They also collaborate with the team and stakeholders to create a product roadmap, a developing and changing picture of what will be delivered, and when. This roadmap helps [remind everyone of the strategic destination](#), so that they don't get lost in the day-to-day details of a project.

They sometimes carry different titles in their organization depending on their other responsibilities, such as management roles (e.g., no need to debate product manager vs product owner). But no matter their job title, they must be empowered to make decisions (in collaboration with their stakeholders) so that the team always knows they are building the right thing, for the right people, at the right time.

On agile projects, the product owner is responsible for product strategy: what gets created, and in what order. They are not, however, responsible for the development process (how the features are created)—the team has that accountability—or exactly how much work the team brings into each sprint. ([They can make some architectural decisions, however,](#))

As part of release planning, and as an ongoing part of product backlog refinement, the team estimates the effort required to create each feature, and uses that estimate to help determine how long it will take to release some set of value to the customer.

During [sprint planning](#), the team selects the amount of work they believe they can do during each sprint. The product owner does not get to say, "We have four sprints left, therefore you must do one-fourth of the product backlog this sprint."

A key component of the role is to motivate the team with a clear, elevating goal. In fact, it's [essential for being effective](#). They are also responsible for describing the what and why of each feature to the team during formal and informal conversations throughout the sprint. That's why it's so important that they are readily available to their teams: their quick answers and clarifications help the team inspect and adapt in real time to ensure they are building the right thing.

In return for the Scrum team's commitment to completing a set of product backlog items each sprint, the product owner makes a reciprocal commitment to not throw new requirements at the team during the sprint. Requirements are allowed to change (and change is encouraged) but only outside the sprint.

Once the team starts on a sprint, it remains maniacally focused on the goal of that sprint.

How Do You Become a Product Owner?

Who is a product owner and how do you become one? Historically, the product owner is a project's key stakeholder. They might be a lead user of the system or someone with a marketing, product development, or user experience background. In essence, they can be anyone with a solid understanding of users, the marketplace, the competition, and future trends for the domain or type of system being developed.

This, of course, varies tremendously based on whether the team is developing commercial software, software for internal use, hardware or some other type of product. The key is that the person in the role needs to have a vision for what is to be built.

The product owner role requires an individual with certain skills and traits, including availability, business savvy, and communication skills. (Some also have technical skills, but it is not a requirement.)

They need to be available to their team. They show commitment by doing whatever is necessary to build the best product possible—and that means being actively engaged with their teams.

Business savvy is important because they are the decision maker regarding what features the product will have. That means, the agile PO should understand the market, the customer, and the business in order to make sound decisions.

Finally, communication. POs work closely with key stakeholders throughout the organization and beyond, and the news they have to share [isn't always what the stakeholders want to hear](#). As such, they must be able to communicate different messages to different people about the project at any given time.

The role is a big job, with competing inward-facing and outward-facing needs. For this reason, some larger organizations create product owner teams, or hierarchical layers, including a [chief product owner](#).

Most new POs (and many who have been doing the role for some time) find [role-specific training](#) to be helpful in learning how to be successful. At Mountain Goat Software, we offer the Scrum Alliance [Certified Scrum Product Owner® certification \(CSPO®\)](#) and the [Advanced CSPO](#). We also offer video training in specific PO skills, such as [writing better user stories](#), [estimating with story points](#) and [agile estimating and planning](#). You can even get [coaching on user story-writing workshops](#).

Product Owner vs Project Manager

Some POs and Scrum Masters come from a project management background. Project managers who want to make the transition to product ownership need to keep a few differences in mind.

A project manager sits outside the team. A product owner is a member of the Scrum team.

A project manager is responsible for the success of the project. The entire Scrum team is responsible for the success of a product ([see below](#))

Some project managers have direct management responsibility of team members. Product owners typically do not have direct reports on the Scrum team.

Project managers determine project plans and deadlines, and assign work accordingly. POs rely on their teams for accurate estimate ranges, and communicate the range of functionality that the team will deliver by a certain date, based on their current understanding of the product.

Project managers attempt to deliver a predefined project by minimizing changes. POs inspect the product at the end of each sprint and adapt product features and plans based on what they learn.

Project managers can make good product owners if they are willing to forgo some of their traditional responsibilities and empower the team more than in a traditionally managed project.

Product Owner vs Scrum Master

The primary difference between a product owner and a Scrum Master is that product owners are focused on the product; Scrum Masters are focused on the process the team uses to create that product. [Product owners should not be Scrum Masters](#); and Scrum Masters should not also be POs. Those roles should remain distinct from one another.

Product Owner vs Business Analyst

Product owners are often busy as the role can be time consuming. The most common way of augmenting their capacity is by adding one more or business analysts or system analysts to the team.

POs often delegates to the business analyst responsibility over the functionality in one or more areas of the product.

In tax preparation software, for example, the business analyst may be tasked with understanding new laws around depreciation. The analyst would research the topic and write the necessary product backlog items. In most cases, the product owner will retain authority over prioritizing work identified by the analyst.

Business analysts sometimes make the transition to product ownership; others prefer to be team members, where they can use their backgrounds to help split user stories and uncover hidden requirements.

Should Teams Designate a Technical Product Owner?

A common mistake on agile projects is having both a regular, business-oriented product owner and a so-called technical product owner. The theory is that a technical PO would be consulted on all technical issues, and a business PO would be consulted on all functionality issues.

But teams rarely encounter issues that are entirely technical or entirely functional. Most issues blend technical and functional considerations.

Most importantly, though, the product owner is the owner, not the co-owner. There cannot be two; only one.

Agile projects should not have two people in one role. To do so dilutes the authority of the one rightful PO and creates confusion in team members, who are left wondering which person to talk to about various issues.

What then is the correct role of the would-be technical product owner? They should be considered a stakeholder. A stakeholder is any person or group who has the right to influence a project or is affected by the project.

By that definition, an architect or senior technical person is really a stakeholder. And it's vital that a product's one and only product owner considers the opinions of all stakeholders, including those of architects or senior technical people.

The Whole Team is Responsible for a Product's Success

I occasionally hear two statements about product owners that I strongly disagree with. These statements are that they are "the single wring-able neck on the project" or that they are "the one throat to choke."

These each mean the same thing, that in an agile project, the product owner is the person ultimately responsible for the success of the project. This is wrong, however.

On an agile project—as well as in many other cases—there is no single, wring-able neck. To say there is a way of releasing the rest of the team from responsibility. And this is clearly wrong.

From a manager's perspective it can be nice to always be able to point to one person and say, "That's who I'll blame if things go wrong." However in agile project management the "one throat to choke" argument is false. Historically, there may be one person who takes the blame for things when they go wrong, but that doesn't mean that person was responsible for the failure.

Take the case of a sports team. At the start of a new season, who on a sports team do we say we'll hold responsible for winning the championship? Is it

the coach, the owner, the star player?

Teams that win championships find a way to win games, no matter the circumstances. If the game plan isn't working, the coach and players adapt. If the star player is having a bad day, someone else steps up. The whole team feels responsible for winning somehow, some way. If the team loses, it may be tempting to blame one person or another, but the team knows that each one of them is accountable for the loss. It's never just one person's fault. In reality, there is no single, wring-able neck.

Consider a non-sports analogy. If both parents were involved in raising a child (and assuming one of them isn't abusive or obviously negligent), which parent is the one throat to choke if a child grows up to be a convicted felon? There is a reason we call it a parental unit. Raising a child is a team effort.

The only way to ever create an environment of shared ownership and responsibility is to let go of the notion of having one throat to choke. That doesn't mean no one is responsible. That means that on a successful team, the team members must do their part, or even go beyond a perceived role, to ensure that the team reaches its goals.

Learn About Agile

New to Agile and Scrum?

Agile & Scrum Training

Online Certifications

More...

About Us

Scrum

Video Courses

Contact Us

User Stories

In-Person Certifications

Our Students

Planning Poker

On-Site Courses

Blog

Agile Software Development

Training Schedule

Books by Mike Cohn

Agile Project Management

Compare Courses

Agile Tools

Agile Presentations

Mountain Goat on YouTube

Agile Mentors Podcast

Elements of Agile

Assessment

Presentation by:
Mike Cohn

Tagged:
user stories, product owner, customers, agile processes,
presentations

The vast majority of what has been written about agile processes is intended for programmers and project managers. Unfortunately, very little information is available to help new product owners or customers identify the right thing to build or learn how to juggle competing priorities. As a product owner, agile product manager, or member of a customer team, you need help identifying user needs and communicating how best to meet those needs.

Leading agile speaker and author Mike Cohn has tailored a presentation specifically for you. Use this presentation to describe the role of user stories in capturing requirements and how tests can express a project's conditions of satisfaction. Deliver insight into how to balance features, time, and resources and how product owners affect the quality of the work. Provide a new understanding of how to prioritize features into a release based on something more than the nebulous "business value."

[View the Presentation](#)

[Download a PDF](#)

You may also be interested in:

User Stories: How to Create Story Maps	How to Run a Successful User Story Writing Workshop	Be a Great Product Owner: Six Things Teams and Scrum Masters Need	Epics, Features and User Stories	Advantages of User Stories over Requirements and Use Cases	Relationship between Definition of Done and Conditions of Satisfaction	User Story r... under user n...
Story maps help to create a shared understanding of the product, visualize user needs, and elicit ... Mar 14, 2023	What you need to know to conduct a story-writing workshop with your team. Feb 28, 2023	Learn six ways effective product owners ensure their teams' success. Jan 31, 2023	I've been getting more and more emails lately from people confused about the difference between ... Nov 08, 2022	At the surface, user stories appear to have much in common with use cases and traditional ... Oct 05, 2022	I'd like to clarify the relationship between two important concepts: a team's Definition of Done ... Jun 28, 2022	Mar 14, 2023 38 Comments

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile Assessment		

Presentation by: Mike Cohn **Tagged:** product owner, agile projects, prioritizing, presentations

Product owners are typically asked to prioritize based on the nebulous term "business value." But what, exactly, does that mean? As a product owner or agile product manager, you need some straightforward advice to help you balance all the competing needs of your business.

These slides from agile expert and author Mike Cohn introduce three guidelines you can use to help prioritize features. Learn how expected cost of change, useful knowledge, and learning gained from working software influence prioritization on agile projects.

[View the Presentation](#)

[Download a PDF](#)

You may also be interested in:

[Be a Great Product Owner: Six Things Teams and Scrum Masters Need](#)

Learn six ways effective product owners ensure their teams' success.

Jan 31, 2023

• 16 Comments

[Read](#)

[The Product Owner's Second Team](#)

Successful product owners will see their stakeholders as a team, not merely a set of individuals.

Nov 02, 2021

[Read](#)

[Top 5 changes in the 2020 version of the Scrum Guide](#)

Recently the 2020 version of the Scrum Guide was released. What changes were made that you need to ...

Mar 16, 2021

[Read](#)

[VIDEO] What does a virtual certified scrum course look like?

Our virtual Certified Scrum courses aren't tedious 16-hour Zoom snoozefests. This video shows what ...

May 19, 2020

• 4 Comments

[Read](#)

4 Steps to Persuade a Product Owner to Prioritize Refactoring

Teams often struggle to persuade their product owners to prioritize refactoring. This simple ...

Feb 04, 2020

• 8 Comments

[Read](#)

What I Want for an Agile Christmas

It's coming on Christmas. That time of year when I drop hints to my family about gifts I'd like.

Dec 17, 2019

20 Comments

[Read](#)

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

by
Mike Cohn **Tagged:** product owner, teams, requirements, architecture
21 Comments

In general, the product owner's job is to specify *what* to build, not *how* to build it. But, there may be times when it is appropriate for a product owner to specify some architectural decisions.

For example, years ago when Sun was seeking to promote the Java language, they offered money to companies who used Java to develop certain applications. A couple of product owners I worked with back then mandated that their teams use Java.

In some cases, these were applications with marginal business cases that would not have been justified without funding from Sun, so those mandates made sense. The developers didn't object because they wanted to experiment with the hot, new technology of the time.

A product owner mandating a technical decision should not happen often and product owners should exercise great care in doing it. And product owners who do so had better be right when they dictate a technical decision.

In most of the cases when they dictated Java, they really weren't right—Java in its early days really wasn't up to some of the challenges of some of those applications.

As another example, consider an embedded device. The product owner has decided the product will be economically viable if produced on a particular piece of hardware, but not on a more expensive piece of

hardware.

The product owner may tell the team that they are restricted to using the lesser hardware. Sure, the team would probably *prefer* the more expensive hardware, but the product would not be economically viable then.

So, product owners can dictate architectural decisions. But they should do so sparingly, wisely and ideally, in consultation with the team.

Get Your Free Scrum Cheat Sheet

Sign up below to receive this FREE reference.

First Name

Email Address

[Privacy Policy](#)

Posted: June 16, 2015

Tagged: product owner, teams, requirements, architecture

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied* for

Agile Software Development, Agile Estimating and Planning, and Succeeding with Agile as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

What Is Cross-Functional Collaboration in Agile?

Perhaps the most prevalent and persistent myth in agile is that a cross-functional team is one on ...

Feb 14, 2023 • 49 Comments [Read](#)

Be a Great Product Owner: Six Things Teams and Scrum Masters Need

Learn six ways effective product owners ensure their teams' success.

Jan 31, 2023 • 16 Comments [Read](#)

Don't Equate Story Points to Hours

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • 39 Comments [Read](#)

Relationship between Definition of Done and Conditions of Satisfaction

I'd like to clarify the relationship between two important concepts: a team's Definition of Done ...

Jun 28, 2022 • 38 Comments [Read](#)

Agile Spikes Deliver Knowledge So Teams Can Deliver Products

On agile projects, a spike is a time-boxed research activity that helps teams make better decisions ...

May 11, 2022 • 48 Comments [Read](#)

What Is a High-Performing Agile Team?

How to build and sustain a Scrum team that exceeds the sum of its parts.

Apr 19, 2022 [Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile **Agile & Scrum Training** **More...**

New to Agile and Scrum? Online Certifications About Us

Scrum Video Courses Contact Us

User Stories In-Person Certifications Our Students

Planning Poker On-Site Courses Blog

Agile Software Development Training Schedule Books by Mike Cohn

Development Compare Courses Agile Tools

Agile Project Management PDUs and SEUs

Agile Presentations

Mountain Goat on YouTube

Agile Mentors Podcast

Elements of Agile

Assessment

by

Brian **Tagged:**

Milner product owner, agile manifesto, stakeholders

Comments

For several years confusion plagued the Scrum community when using the word team. Since we had both a Scrum team and a development team, each conversation needed clarification about which team was meant. In the 2020 version of the Scrum Guide, the authors did away with the term “development team” in favor of simply “Developers,” partly in hopes of putting an end to this confusion. Finally, we have but one team—the Scrum team.

Now that this has been addressed and all confusion cast aside, I propose that there is still a second team and that it's vital to a successful product owner. Product owners need to approach their stakeholders as team members in order to maximize communication and collaboration across that group.

Of course product owners clearly are full members of the Scrum team. Think of a Sprint Review: the entire Scrum team presents their product increment to those outside of their Scrum team. The demarcation line is clear—those inside present to those outside.

However, I have seen many product owners treat their stakeholders as a set of individuals, each with their own needs and desires. These POs run from stakeholder to stakeholder not only to gather their feedback, but also to share information between their stakeholders. They get snarled up in trying to explain to one stakeholder why a certain feature is important to some other stakeholder. They waste energy juggling the priorities of individual stakeholders. As the Bard tells us, “That way madness lies.”

Successful product owners see their stakeholders as a team instead. Just as a Scrum team does, their stakeholder team should collaborate to make decisions. This is why we suggest such activities as [Luke Hohmann's Buy A Feature](#) to help a group of stakeholders come together to foster priority decisions. Stakeholders have to convince each other of the priority of their features: they leave the game with a common agreement on priority order.

If you have a set of individual stakeholders, here are a few ways to help them act as a team:

Meet as a team - If you regularly have individual meetings with your stakeholders, consider regular group meetings. When you make this a routine, stakeholders will come to count on that specific time as their chance to be heard. And if some stakeholders don't make it, they likely had higher priorities than that meeting. The stakeholders who most need to be heard will be the ones that show up.

Hone your facilitation skills - What are you going to do with your stakeholders once you have them all together? A little facilitation skill can go a long way here. Give them multiple ways of expressing their opinions and having their voices heard. Create activities in which stakeholders discuss the product with each other. Provide the space and a structure so they can offer their ideas, consider them together, and make group decisions about them.

Don't treat all stakeholders equally - We have to recognize as product owners that we have different types of stakeholders. Our goal is to identify our key stakeholders and spend our efforts forming them into a team. Fran Ackermann and Colin Eden in their book [Making Strategy](#) describe this using their *four-square power-interest grid*. Stakeholders are either high or low interest and have either high or low power. We're most keenly interested in the high-interest/high-power stakeholders. You'll need strategies for each—but if you don't have the high-interest/high-power stakeholders in the room, you may be wasting your time.

While there is only one team in Scrum, product owners will be most successful when they treat stakeholders as a second team. After all, the Agile Manifesto tells us that “the best architectures, requirements, and designs emerge from self-organizing teams.” Why wouldn’t we apply this wisdom to our stakeholders as well?

Get Your Free Scrum Cheat Sheet

Sign up below to receive this FREE reference.

First Name

Email Address

We hate spam and promise to keep your email address safe. Unsubscribe at any time. [Privacy Policy](#)

Posted: November 2, 2021

Tagged: product owner, agile manifesto, stakeholders

About the Author

Brian is the Senior Vice President of Training and Coaching with Mountain Goat Software and is a Certified Scrum Trainer.

Starting out as a developer, Brian worked up through management layers, then transitioned to Scrum Master and then Coach. His practical experience in both waterfall and agile organizations helps him clarify what works and what doesn't, plus he has many years' experience helping teams transition to agile.

Brian also brings more than 20 years of software development experience to his classes. People remark on his ability to answer even tough questions with ease, enthusiasm, and insight.

Scrum Certifications include: CSM, CST, CSPO, CAL-Educator, CAL-1, A-CSM, A-CSPO, CSP-CM, CSP-PO, Path to CSP Educator, Scrum Foundations Educator.

You may also be interested in:

Be a Great Product Owner: Six Things Teams and Scrum Masters Need Learn six ways effective product owners ensure their teams' success. Jan 31, 2023 • 16 Comments Read	7 Ways to Get and Improve Fast Feedback Here's how to get the rapid feedback you need to ensure you build the right product. Jul 26, 2022 Read	Top 7 Ways to Get Stakeholders to Attend Sprint Reviews Poorly attended sprint reviews cause real problems. Fortunately there are easy fixes. Mar 08, 2022 Read	Are Fixed-Price Projects Agile? Fixed-price projects may seem at odds with agile. But that doesn't have to be the case. Oct 26, 2021 Read	Out of Office What does working at a "sustainable pace" mean to you? Sep 14, 2021 Read	Top 5 changes in the 2020 version of the Scrum Guide Recently the 2020 version of the Scrum Guide was released. What changes were made that you need to ... Mar 16, 2021 Read
---	---	--	--	---	--

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile Assessment		

by
Brian Tagged:
Milner product owner, agile, scrum master, scrum
Comments

Every few years Ken Schwaber and Jeff Sutherland, the authors of Scrum, get together to update the Scrum Framework as it is defined in the Scrum Guide. These are the official rules of Scrum and they've gone through many iterations.

At the end of 2020, their latest version dropped, and it's simply titled the [2020 Scrum Guide™](#). The goal was to simplify the framework and reduce unneeded portions, so the framework can be more accessible to a larger audience. This can best be seen in the decrease in size of the Scrum Guide (from 19 pages to 13).

They also made several changes to the framework, and while some aren't that important, I believe some are worth considering.

Here I'll outline what I think are the top 5 changes, tell you why they matter (or don't), and also give you my opinion on the changes. This is by no means a comprehensive list of changes. It's just the top 5 that I think are worth discussion.

Change 1: Accountabilities replace roles

CHANGE: The term "role" has been dropped in favor of the term "accountabilities" to label the individuals on a Scrum team.

IMPACT TO YOUR TEAM: For experienced teams - You can take or leave this change, since names and labels are more likely to be merely academic for you and your team. For new teams - You might decide simply to switch the words—or not.

BRIAN'S OPINION: Apparently some people thought that the term “role” meant job title. The authors wanted to find a term more like a position on a sports team. For example, all players are baseball players but each one is responsible for a specific duty: a catcher catches, a first baseman keeps track of happenings at first base. But your first baseman is of course also up to bat now and then, and in the same way any job title then might fill a specific Scrum team role without needing to hire someone with that job title. Scrum Masters don’t have to have “Scrum Master” on the resume; a project manager can play the role of Scrum Master as long as they adhere to what the Scrum Guide spells out for that role. “Accountabilities,” then, is an attempt to separate the two ideas: a role and a title aren’t the same thing.

The term “accountability,” though, doesn’t solve the problem. When most people (myself included) think of accountabilities, they think of responsibilities: “I am accountable for putting my dishes in the dishwasher.” The problem comes with the fact that I can be accountable for multiple things but the new Scrum Guide states that the Scrum team “consists of one Scrum Master, one product owner, and developers.” These terms represent some undefined word that contains multiple other accountabilities. For example, each developer is accountable for quality, creating a plan for the sprint, and holding one another accountable. That missing word sounds an awful lot like a role to me. If that’s a problem for some, stick with the sports theme of Scrum and call them positions. Regardless of how you think of them, they are parts that people step into in order to play Scrum. “Accountability” just doesn’t describe that, in my opinion, and will only add to the confusion.

Change 2: No more development team

CHANGE: There is no longer a development team, just developers.

Scrum Guide authors Jeff and Ken stated that they had always disliked having a “team within a team” by having a development team inside the Scrum team. In this latest version, they fixed that by getting rid of the development team concept. The Scrum team is the only team in Scrum now and it consists of three accountabilities: Scrum Master, product owner, and developers.

IMPACT TO YOUR TEAM: For experienced teams - This change is likely only cosmetic. And it harkens back to the past; we did once call them developers, not dev team. But do understand that this shift in language is to try to emphasize the “one team” concept.

For new teams - Simply understand that there is only the Scrum team now that is made up of three accountabilities: product owner, Scrum Master, and developers.

BRIAN'S OPINION: While I get the purpose of this change and agree that there is only one team in Scrum, I worry a bit about the unintended consequences of this change. This is mostly due to the next aspect of the new Scrum Guide: the guidelines for developers. Previous versions spelled out explicitly the rights and responsibilities of this group. Now that’s missing in the name of brevity. One phrase I am particularly sad to see go runs, “No one (not even the Scrum Master) tells the Development Team how to turn Product Backlog into Increments of potentially releasable functionality.” While some might consider this implied and therefore not needed, I believe there are plenty of organizations that aren’t used to operating in this manner and need

this particular prompt telling them that the *how* of building the product lies squarely with the developers.

On the surface, then, it should be a positive change. As practitioners though, we need to help organizations understand the intent of independence is still there even if it's no longer spelled out explicitly.

As long as we are tinkering with names, I would have preferred to update this to something like producers or creators. Developer is a software-centric term and disinclines many outside of software from employing Scrum. I wish the new Scrum Guide had taken this chance to provide a word less connected to the software world.

Change 3: No more Servant Leaders

CHANGE: The Scrum Guide has dropped the term "Servant Leader" in describing Scrum Masters and replaced it with "true leader."

IMPACT TO YOUR TEAM: For experienced teams - Likely no impact here unless "true leader" has an alternate meaning that isn't clear. For new teams - Since you don't have any history with this phrase, this likely has no impact to you.

BRIAN'S OPINION: I think this change is the hardest to understand of the bunch. What is offensive about Servant Leadership as a term? And what in the world is a "true leader?"

I can only speculate as to the reason for this change. Perhaps it's connected to the movement in the tech world to drop the terms Master/Slave in reference to databases, since some people thought it could be considered a reference to slavery and therefore offensive. Whatever your opinion on that one, I did also hear some rumblings in agile circles that perhaps we should reconsider the term Scrum Master, and even Servant Leader, for the same reasons.

Servant Leadership has nothing to do with slavery. Servant Leadership was a movement begun by Robert Greenleaf in the 1970s that's meant to paint the picture of a humble leader serving those who wish to be led. It's an incredibly powerful paradigm shift that occurred in the business world—and the role of the Scrum Master has traditionally been linked to this term for good reasons. I said above that I think "true leader" is a meaningless phrase. There is no "true leader" movement or philosophy. It sounds as if someone wanted to keep the meaning of Servant Leadership but replace the words with something similar. It would be a shame if this is where the new term emerged; it would be based on misinformation and a lack of understanding of what Servant Leadership is.

I hope the authors will reconsider, that they'll re-examine the heritage and political correctness of the term "Servant Leader." I'd like the term to find its way back into a future version of the Scrum Guide.

Change 4: Adding in Commitments

CHANGE: Each Artifact now has an accompanying "Commitment": Product Goal for Product Backlog, Sprint Goal for Sprint Backlog, and Definition of Done for the Increment.

IMPACT TO YOUR TEAM: For experienced teams - This should not impact your team provided you've already been working with a Product Vision. For new teams - You will hear less and less about a Product

Vision and more about this Product Goal in Scrum. They seem to be generally synonymous.

BRIAN'S OPINION: This change is a bit more tricky. Of these commitments, Sprint Goal and Definition of Done both existed in previous versions of the Scrum Guide, but not as an attached commitment to their Artifacts. The Product Goal is the only new idea, at least when it comes to the Scrum Guide.

Most trainers and coaches have been teaching product owners for years about the benefits of having a Product Vision, which is the guiding force for the product. As Roman Pichler says, it is the product's "true north." Now let's see how the new version of the Scrum Guide refers to the Product Goal:

"The Product Goal describes the future state of the product which can serve as a target for the Scrum Team to plan against."

If you are having trouble seeing the difference between a Product Vision and this Product Goal, you are not alone. Why did the authors choose to ignore that the community was already using the term Product Vision and replace it with Product Goal? I can't find a reason other than that it pairs better with Sprint Goal. If a case can be made for how Product Vision and Product Goal are different, I've not heard it yet. So if you are using Product Visions today and you start calling them Product Goals, that might be all you would have to do to technically align to the new Scrum Guide.

There is one small detail of this addition as it's defined in the Scrum Guide that irks me. For both the Product Goal and the Sprint Goal, this new Scrum Guide says that these items are IN their respective backlogs. If it weren't spelled out so explicitly, I wouldn't give this a second thought. But how is it possible considering that the Product Backlog is made up of everything known that's needed to produce the Product? The Sprint Backlog similarly contains everything known that is needed to produce the sprint's Product Increment. In both cases, the backlog is a place of things to do: features, tasks, bugs, spikes, etc. How is a goal then IN the backlog? Unless we are redefining the backlog or creating special new sections of the backlogs for goals, it doesn't seem to make sense that they are IN the backlogs. I yield the soap box.

Change 5: Sprint Planning three questions

CHANGE: Sprint Planning has expanded from two questions (WHAT and HOW) to three (WHY, WHAT, and HOW).

IMPACT TO YOUR TEAM: Small, if any.

BRIAN'S OPINION: For years now, the Sprint Planning event has had a goal of answering two questions: WHAT are we going to do and HOW are we going to accomplish it? In this latest version of the Scrum Guide, the authors have added the question up top that asks, "Why is this Sprint valuable?" They then go on to explain that the Scrum team collaborates to create a Sprint Goal.

None of that is new.

As part of the WHAT discussion, teams have always had a Sprint Goal that the product owner proposes and the whole Scrum team then collaborates to define. If we are being picky with words here, I'm not sure the Sprint Goal is giving us the WHY for the sprint. It's giving us the goal of the sprint but not why that goal is important. Put another way, the Scrum team doesn't have to justify their goal, they just have to define it.

In my mind then, this is a regular part of defining WHAT the team will work on. If your team needs three delineated questions to determine the Sprint Goal, the Product Backlog items, and the tasks to create them, feel free.

Asking two questions or three likely makes no difference to how Scrum teams run their sprints. I've yet to hear a valid explanation of why this was explicitly called out in the new Scrum Guide.

A Common Theme to These Changes

You've likely noticed a theme here. Most of the changes in this new version will not have a substantial impact to how you are running Scrum in your organization today. Sure, you might change the terms you use or you might answer a question slightly differently on a test, but otherwise you will not need to change a thing to adhere to this newest version of Scrum.

The bigger impact comes in what was removed. In order to make the Scrum Guide smaller, elements that some will see as key have been removed. If organizations remember the 2017 Scrum Guide and aren't changing their methodology now, there may not be a problem. For new organizations adopting Scrum for the first time who don't have that history to call upon, there might be crucial holes in the framework that will not be filled in a way consistent with Scrum's historical strengths. I hope these organizations will seek out guidance to help with their lack of experience. If organizations are learning from the 2020 Scrum Guide alone, we may not see the impact of these changes for several years down the road.

For new teams, I suggest that they read the 2017 version in order to have a better understanding of the 2020 one.

My advice for established teams? Keep doing what you are doing—if it's working for you. If not, inspect and adapt.

Get Your Free Scrum Cheat Sheet

Sign up below to receive this FREE reference.

First Name

Email Address

We hate spam and promise to keep your email address safe. Unsubscribe at any time. [Privacy Policy](#)

Posted: March 16, 2021

Tagged: product owner, agile, scrum master, scrum

About the Author

Brian is the Senior Vice President of Training and Coaching with Mountain Goat Software and is a Certified Scrum Trainer.

Starting out as a developer, Brian worked up through management layers, then transitioned to Scrum Master and then Coach. His practical experience in both waterfall and agile organizations helps him clarify what works and what doesn't, plus he has many years' experience helping teams transition to agile.

Brian also brings more than 20 years of software development experience to his classes. People remark on his ability to answer even tough questions with ease, enthusiasm, and insight.

Scrum Certifications include: CSM, CST, CSPO, CAL-Educator, CAL-1, A-CSM, A-CSPO, CSP-CM, CSP-PO, Path to CSP Educator, Scrum Foundations Educator.

You may also be interested in:

Be a Great Product Owner: Six Things Teams and Scrum Masters Need Learn six ways effective product owners ensure their teams' success. Jan 31, 2023 • 16 Comments Read	Six Attributes of a Great ScrumMaster Scrum Masters exhibit a shared set of traits. Discover 6 in-demand qualities, plus a bonus ... Jan 17, 2023 • 32 Comments Read	Agile Mentors Podcast: Recap of Opening Series on Scrum Recapping our agile podcast's initial launch series of topics Sep 27, 2022 Read	From Project Manager to Scrum Master -3 Tips for Making the Transition What does a good project manager need to do to become a great Scrum Master? Aug 23, 2022 Read	Seven Tips for Running a Virtual Daily Scrum How do you effectively facilitate a daily scrum with a remote team? May 03, 2022 Read	What Does Scrum Mean by Cross Functional Teams? What exactly does cross functional mean and why does it matter? Apr 05, 2022 Read
--	--	---	--	--	---

Sorry, the browser you are using is not currently supported. Disqus actively supports the following browsers:

[Firefox](#)
[Chrome](#)
[Internet Explorer 11+](#)
[Safari](#)

[gkevans](#) • 2 years ago

Nicely framed observations, Brian. I have been puzzled over some of the changes. Abstraction can become a liability. My conclusion is that Sutherland and Schwaber are so close to Scrum that they may have embraced their intimate knowledge too tightly and set new teams without deep Scrum experience at a disadvantage. Scrum could always be described on a single page, but the nuances of execution...they take years.

[Brian Milner](#) • 2 years ago

Thanks, [gkevans](#). Yes, I hope as well that newer Scrum teams don't lose anything important in this shortened version. Time will tell!

[next bubble](#) • 2 years ago

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn

[Compare Courses](#)

[Agile Tools](#)

[Agile Project Management](#)

[PDUs and SEUs](#)

[Agile Presentations](#)

[Mountain Goat on YouTube](#)

[Agile Mentors Podcast](#)

[Elements of Agile](#)

[Assessment](#)

by
Mike Cohn
Tagged:
product owner, scrum master, courses
4 Comments

Before I dive into today's post I want to pause and thank everyone who got in touch after [last week's blog](#) to show their support for the new trainers joining the Mountain Goat Software family.

Many people knew of or had taken classes with Lance, Julie, Mitch, or Scott and were full of warm wishes and praise for their talents. The response we had was overwhelming, and I firmly believe we have the best blog readers for their enthusiasm at our new direction.

We also received a lot of questions about the virtual classes and I thought that a virtual walkthrough would be a neat way to show what it's like to take a virtual certified scrum class.

The team and I put together a short video to tell you all about it and I wanted to share that with you today.

Currently, we are running Virtual Certified ScrumMaster® and Virtual Certified Scrum Product Owner® classes.

If you'd like to read more or book a class, click the button below:

Download Scrum Master Guide

Get a free copy of Situational Scrum Mastering: Leading an Agile Team

[Get my free guide now!](#)

Posted: May 19, 2020

Tagged: product owner, scrum master, courses

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the *Better User Stories* video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

Be a Great Product Owner: Six Things Teams and Scrum Masters Need

Learn six ways effective product owners ensure their teams' success.

Jan 31, 2023 • 16 Comments [Read](#)

Six Attributes of a Great ScrumMaster

Scrum Masters exhibit a shared set of traits. Discover 6 in-demand qualities, plus a bonus ...

Jan 17, 2023 • 32 Comments [Read](#)

From Project Manager to Scrum Master -3 Tips for Making the Transition

What does a good project manager need to do to become a great Scrum Master?

Aug 23, 2022 [Read](#)

Seven Tips for Running a Virtual Daily Scrum

How do you effectively facilitate a daily scrum with a remote team?

May 03, 2022 [Read](#)

What Does Scrum Mean by Cross Functional Teams?

What exactly does cross functional mean and why does it matter?

Apr 05, 2022 [Read](#)

How to Create Self-Sufficient Scrum Teams

Mar 01, 2022 [Read](#)

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile Assessment		

by
Mike Cohn **Tagged:**
product owner, prioritizing, refactoring
8 Comments

Teams often struggle to convince their product owners to allow time for refactoring, especially significant refactoring efforts. In this post I want to share a four-step framework teams can use to justify refactoring and persuade their product owners to allow it.

What Is Refactoring?

First, let's be clear on what *refactoring* means. Refactoring is changing the *structure* but not the *behavior* of code. This means a team would not undertake refactoring to fix a bug. Fixing a bug involves changing the behavior of the code.

A team could, however, refactor to prevent further bugs from occurring. In fact, preventing future problems is one of the most common reasons to refactor.

For example, a team has experienced a disproportionate number of bugs in a portion of the product's code. The team would like to "clean up" (refactor) that code so that when they revisit that code in the future, the code is not as fragile.

You can see why, then, that in changing the structure but not the behavior of code, refactoring can be understandably hard for product owners to prioritize highly. The "if it's not broke, don't fix it" mentality

Assessing the Cost of Not Refactoring

When a product owner is reluctant to allow a given refactoring, I've found it best for the development team to present their argument in the language of the product owner. That is, to present an economic justification of the refactoring.

To do that, I suggest the following four-step process:

- Estimate the impact of the status quo
- Estimate the effort to perform the refactoring
- Estimate how much time will be saved after refactoring
- Estimate the payback period

Following these steps enables a team to put their case to the product owner economically. For example, they may be able to present a refactoring by saying, "We think this will take a third of an iteration to do, but it will pay back that investment in only five sprints." When the case for a refactoring is made this way, it is much more straightforward for a product owner to decide whether the refactoring is worth pursuing.

Step 1: Estimate the Impact of the Status Quo

The first step is to estimate the impact of leaving the code as is; that is, the cost of not refactoring.

Try to do this by gathering actual data on time spent investigating, fixing, and validating defects in the code that you want to refactor. When data is unavailable, it's OK to take an educated guess.

If you do need to guess, I recommend guessing conservatively so as to not overstate the case in favor of refactoring. In this way the team can be more confident they are truly recommending an advisable course rather than merely the course they'd prefer.

When a team gets a reputation for overpromising the benefits of technical work, such as refactoring, they make it harder for themselves to sell the benefits of similar work in the future.

An Example

Let's consider an example that uses a mix of real data combined with some educated guessing by a team, since I find that is the most common.

In this case, our team digs into its product backlog or defect tracking software to determine how many bugs have been reported against a particular functional area within their system. They decide to look at data over the preceding six two-week iterations, approximately three months. I find three months a good compromise between using more data and spending more time on the analysis.

By looking at the work performed in the prior six iterations, the team determines that 12 bugs were fixed during that period. Eight were categorized as high severity. Four were medium severity. There were additionally three low-severity defects that were not fixed.

This is all data. If the team has tracked actual effort to fix these bugs, use that. However, many teams do not track that data and may need to guess at how long the defects took to fix. That's OK. As you'll see, this doesn't need to be perfect.

Let's suppose our team estimates that each high-severity bug took 12 hours to fix. That includes investigating, coding and testing plus any time that was spent discussing or documenting the solution.

Next, the team guesses that the medium-severity bugs each took half as long, so six hours each.

Putting all this together, our team spent $8 * 12 + 4 * 6 = 120$ hours over the previous three months.

What About the Unfixed Bugs?

Remember, there were also three low-severity bugs that were reported but were not fixed. Handling those requires a judgment call.

If the product owner really, really, really would have liked those fixed but just couldn't justify it, include an estimate for fixing them. On the other hand, if the team's definition of low severity is such that those defects are so minor, they don't need to be fixed, leave them out of the calculation.

Why I Recommend Hours Rather than Story Points

I recommend using hours for these calculations because most teams do not put story points on their defects. However, if your team does estimate defects in story points, you can use points for these calculations. I'll proceed from here, though, sticking with hours. Nothing in the analysis changes except the units.

Step 2: Estimate the Effort of Refactoring

The second step in preparing to persuade a product owner to prioritize a refactoring is to estimate the effort of performing that refactoring.

To do this, team members should estimate the refactoring just like they would any other product backlog item. That means the estimate could be in story points or in hours.

The estimate will then need to be converted into the same unit used in step one when the team estimated the impact of not doing the refactoring. Since the team in that example used hours, I'll continue with hours for this step.

Let's suppose our team has a product backlog item to refactor the troublesome code. I suggest they have the equivalent of a miniature iteration planning meeting during which they discuss what work they anticipate as part of the refactoring.

They might, for example, determine that only coding and testing are needed. But to better understand the work they split that out into three coding and two testing tasks. Each task is estimated quickly and roughly--again, this doesn't need to be perfect. In our case, let's say that is a total of 40 hours.

Step 3: Estimate How Much Time Will Be Saved

In the third of our four steps, the team estimates how much time will be saved after the refactoring is complete. This again will usually be a combination of data and educated guessing.

To see how this is done, let's return again to our example. In step one, our team looked at data and determined they had fixed twelve defects (eight high-severity, four medium-severity) during the previous six iterations. They estimated they spent 120 hours correcting those defects.

This means the team was spending 20 hours per iteration ($120 \div 6 = 20$) addressing defects in the code they want to refactor.

In most cases, it would be unrealistic to think that refactoring will eliminate all defects and all need to revisit old code. So let's have our team make an assumption here that the refactoring they propose will reduce by half the time spent on defects in that area of the system.

In other words, instead of spending twenty hours per iteration on defects in that part of the system, refactoring will reduce that to only ten hours needed per iteration.

It's quite possible the team could improve things further than that. But, again I recommend being conservative when estimating improvements.

Step 4: Estimate the Payback Period

It's time to see if the refactoring will be worth doing. To do that, divide the effort to perform the refactoring (as determined in step two) by the time saved (as determined in step three).

In this example, our team estimated that the desired refactoring would take 40 hours. And they estimated that it would save them 10 hours per iteration.

The payback period is calculated by dividing the number of hours needed to refactor by the number of hours saved per iteration as shown here.

Hours to Refactor

Hours Saved per Iteration

Since the team here plans 40 hours of refactoring and will save 10 hours per iteration, that is a payback period of four iterations.

A payback period of four iterations should be looked upon quite favorably by most product owners. This means that after four iterations the team's time spent on refactoring will be repaid and each subsequent iteration will have additional hours available compared to what the team has before refactoring.

How Short Does a Payback Period Need to Be?

A team and product owner should look for refactoring opportunities that have the shortest possible payback periods. But there's no strict guidance along the lines of "all refactorings have to be repaid in n iterations."

The suitable payback period is influenced by many factors on the project, including the urgency of other features and the projected life of the product. It would be hard to justify any refactoring on a product set to be retired in six months.

Putting the Argument in the Product Owner's Terms

By assessing the refactoring using these four steps, a team is able to present an economic case for the refactoring. Their rough estimates of the investment (the time to perform the refactoring) divided by the per-iteration savings gives the product owner a general idea of how long until the investment is recouped.

When a product owner is presented with an argument for refactoring in these terms, the product owner can make a rational decision based on the economic merits of the refactoring. That benefits the team, the product owner, and the product's users and customers.

What's Your Experience?

Is your product owner receptive to requests to refactor? Or, if you are a product owner what helps persuade you to approve time spent refactoring? Please share your thoughts in the comments section below.

Get Your Free Scrum Cheat Sheet

Sign up below to receive this FREE reference.

First Name

Email Address

[Privacy Policy](#)

Posted: February 4, 2020

Tagged: product owner, prioritizing, refactoring

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile Assessment		

by
Mike Cohn **Tagged:**
20 product owner, transitioning to agile, scrum master, leadership
[Comments](#)

It's coming on Christmas. That time of year when I drop hints to my family about gifts I'd like.

And they give me socks instead.

My family should know me better than anyone. But since dropping hints doesn't work with them, I'm going to assume it won't work with others either. So on behalf of team members everywhere, I'm going to tell Scrum Masters, product owners, and managers a couple of things I'd like for Christmas. And I'll be more explicit than dropping subtle hints. The last thing I need is more socks!

From My Scrum Master

There are two things I'd really appreciate from my Scrum Master.

More Focus on the Principles and Values of Agile and Scrum

It would be great if you could focus more on the principles and values of agile and Scrum rather than focusing just on having us adhere to the rules of Scrum.

As an example, it's great to tell us a daily scrum has to be done in 15 minutes, but occasionally taking 20

minutes is really helpful for us. We understand that wouldn't be good for everyone. And we don't want to spend an hour.

But if everyone values the little extra depth we're covering in those meetings, why do we have to rush through them?

Help us be aware of the rules but use the principles and values to help us do what's best for us.

Rather than Telling Us What to Do, Try Selling Us on the Idea

Rather than telling us what to do, try selling us on the idea. You told us we were doing two-week sprints. That pissed us all off because we each wanted to do four-week sprints. And we doubted we could finish anything even in that amount of time.

But, I have to say it: You were right. And we all acknowledge that now.

But, wouldn't it have been better if you could have sold us on the idea rather than told us it was what we were doing? When you *sell* rather than *tell*, it may take longer, but we become stronger supporters of the idea in the end.

From My Product Owner

We've come a long way since the last holiday season, Product Owner. Last year we didn't invite you to retrospectives and didn't allow you to participate in daily scrums. Since then, we've learned that the more we function as one whole team, the more successful we are.

Despite the progress we've made together, there are a couple of things I'd like to ask of you this year.

Your Time

We know you're busy with work outside the team. But we really need more of your time. Or at least more access to you. Maybe we can set up a Slack channel where you promise to always respond to by the end of each day?

More Understanding of Why We're Doing what We're Doing

You seem to have a really solid grasp of where you want our product to evolve over the next few years. The more you can help get that in our heads, the better it will be for all of us.

A couple of good techniques I've seen for sharing your vision are:

Write the press release we'd like to go out a few months from now. It's not the real press release--it's what we'd want people saying about us.

Write a magazine or app store review we'd like to earn. Again, it's not the real review. But it can be our shared vision of what we'd like said in a review.

Your Trust

When we tell you we need time to clean up something in the code or to “refactor” the code, it’s because something in the code is causing real problems.

It’s not like we’re asking to do something fun. We’re asking to do something equivalent to cleaning the garage. Please take the request seriously. We understand you can’t always let us clean something up the minute we ask, but please don’t take too long or the problem will get bigger.

From my Managers

I need you to understand I really enjoy working here. If I didn’t, I’d leave. Life is too short to work where work is always work and never fun. And so I’m here because I enjoy it. But here are a couple of things you could do that would really help.

Stop Trying to Measure My Productivity

Managers have been trying to measure the productivity of knowledge workers, like me, ever since Peter Drucker coined that term in 1959.

If no one has solved this challenge yet, you and I aren’t going to be the two to do it. Velocity doesn’t measure my productivity. And if you try, I’ll just gradually inflate my estimates.

Velocity will go up but my productivity will stay the same.

Do you want to know if my team or I are productive? Observe us. Talk to us. Talk to our customers.

What is productivity, anyway? We could develop more features faster if we stopped trying to build the right features. No one wants that.

Understand I Can Be as Agile as You’ll Let Me

Being agile isn’t something I can do on my own. I need your help. You don’t need to be fully agile yourself. But you and others in the company need to stop getting in our way.

For example, stop those meetings we have all too often to talk about and often estimate work that is way in the future and may not be needed anyway.

Get the Facilities group to move us onto one floor. I understand why half my country is in another country halfway around the world. But there’s no reason for my teammates here to be spread across two floors.

When you ask us to lock down a date and exactly what features will be delivered, you really hamstring our ability to be agile. Our options then become limited to things like working overtime, implementing just enough of some features, adding people, and cutting quality. When we try cutting quality, it sometimes works in the very short-term but always costs us more in the end.

So, you don't need to be agile yourself. But you need to understand the ways in which your actions affect our ability to be agile.

What Do You Want?

What would you like for Christmas this winter? Or whatever holiday you celebrate this time of year, whether that's Hanukkah, Pancha Ganapati, Kwanzaa, Newtonmas, Rohatsu, Festivus, the winter solstice, or even just the start of a new year? Please use the comments below to share what you'd like to get from your Scrum Master, agile coach, product owner, boss, managers, organization, teammates, or even me.

Download Scrum Master Guide

Get a free copy of Situational Scrum Mastering: Leading an Agile Team

[Get my free guide now!](#)

Posted: December 17, 2019

Tagged: product owner, transitioning to agile, scrum master, leadership

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

Be a Great Product Owner: Six Things Teams and Scrum Masters Need

Learn six ways effective product owners ensure their teams' success.

Jan 31, 2023 • 16 Comments [Read](#)

Six Attributes of a Great ScrumMaster

Scrum Masters exhibit a shared set of traits. Discover 6 in-demand qualities, plus a bonus ...

Jan 17, 2023 • 32 Comments [Read](#)

8 Reasons Scrum Is Hard to Learn (but Worth It)

Transitioning to Scrum is worth it, but some aspects are challenging.

Oct 11, 2022 [Read](#)

Elements of Agile: An Agile Assessment Tool for Iterative Improvements

New to agile, or needing an agile re-boot? We've got a new no-cost assessment and actionable ...

Sep 13, 2022 [Read](#)

From Project Manager to Scrum Master -3 Tips for Making the Transition

What does a good project manager need to do to become a great Scrum Master?

Aug 23, 2022 [Read](#)

Seven Tips for Running a Virtual Daily Scrum

How do you effectively facilitate a daily scrum with a remote team?

May 03, 2022 [Read](#)

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

by
Mike Cohn **Tagged:**
107 product owner, leadership
[Comments](#)

Scrum has only three official roles: Scrum Master, product owner, and team member. No distinction is made for roles like programmer, tester, database engineer, analyst, designer, or so on.

A tester is, for example, a team member. The tester has the same responsibilities as every other team member: produce a potentially release product increment by the end of the sprint that achieves the sprint goal.

How each team member helps the team accomplish this goal will differ. A tester will, whenever possible, select testing work from the sprint backlog. A designer will select design work.

But everyone on a good Scrum team is expected to help out any way they can.

Specialist Roles on Scrum

In contrast to the role of team member are the roles of product owner and Scrum Master. These roles can be thought of as specialists.

Neither the product owner nor the Scrum Master has a responsibility to directly aid in creating a potentially releasable product increment during the sprint. On many teams, they *can* help but it isn't required as part of

Scrum.

Why Are These Roles Unique?

What is so unique about the product owner and Scrum Master roles that each is a distinct role, unlike tester, programmer and so on?

Many teams already experiment with rotating Scrum Master duties among multiple team members. Others treat it as an additional set of responsibilities taken on by a team member.

Yet the vast majority of Scrum teams still have a separate product owner role filled by a dedicated person representing the interests of the organization or its stakeholders.

An Intriguing and Appealing Future

I think an intriguing and appealing future is presented by the idea that the product owner role could go away. Current product owner responsibilities would become shared across the entire team, much as testing responsibilities are shared today.

I do not believe there is anything inherent in the product owner role that prevents it being shared.

Just as testing on a good agile team today is a responsibility shared by the entire team so too would product decisions be shared.

Teams are expected to self organize and determine for themselves the right answer when faced with architectural choices. The same could be expected of product decisions.

When faced with a decision that would have previously been made by their product owner, team members instead talk about it and decide collaboratively.

This really is no different than team members making architectural decisions.

Rather than having a dedicated product owner, a team might have a team member with more expertise making product-level decision. But this is no different than having a team member with more experience testing who prefers to test but who will do anything.

What's Necessary for This to Work?

A few things are necessary for this to work. First, developers need to move beyond thinking of themselves as [code monkeys](#). They cannot just show up at work and announce, "I'll code whatever you want, but tell me exactly what it is." For a Scrum development team to participate at this level, team members need to bring their whole heart and passion.

Second, product owners will need to let go of the idea that product-level decisions are solely theirs to make. A tester on a Scrum team may prefer testing over all other activities. But that doesn't mean testers get to make all testing decisions on their own.

These changes mean the entire team will be responsible for achieving whatever big goal it is assigned. Thinking that leads to referring to the product owner as “the single wringable neck” will go out the window.

So Who Is on the Team?

In what I’m describing and in what I’ve encountered already in a very small number of organizations, teams would exist as today with one exception. Rather than a dedicated product owner who is often thought of as separate from the team (and is defined as such by the Scrum Guide), product ownership duties would be performed collaboratively by the team.

Organizations would very likely assign someone with product management experience to the team. This would be no different than how someone with testing experience is assigned to the team today.

This person would be expected to lead or guide many product management decisions. But the person would rely on *selling* teammates on the benefit of a decision rather than *telling* them a decision.

A Natural Progression

I am opposed to anything that creates an us/them divide within a team. I’ve written previously about the importance of [including product owners in daily scrums and retrospectives](#) and I’ve written about [including team members in creating the product backlog](#).

As teams more fully embrace whole-team thinking, differences among roles will be seen as more artificial even as individuals in those roles do more highly specialized work.

I believe we are already seeing this in how organizations view Scrum Masters. In coming years it will be a natural progression for this whole team thinking to extend to product owners.

Teams will have some stakeholder or business representative on the team, but that person will not be *solely* responsible for product decisions as is commonly the case today. Rather, teams will make those decisions entirely collaboratively and with shared responsibility.

What Do You Think?

What do you think about doing away with a distinct product owner role? If your product owner were viewed more as a regular team member today and had to rely on selling the correctness of his or her ideas rather than decreeing them, would that lead to better decisions? Please share your thoughts in the comments below.

Way too many comments for me to keep replying to each individually. I’m continuing to read each, but I simply do not have the time to reply individually to each. Thanks for the discussion. I knew this would create a lot of dissenting views. So thanks for engaging in the debate.

Get Your Free Scrum Cheat Sheet

Sign up below to receive this FREE reference.

First Name

Email Address

[Privacy Policy](#)

Posted: September 24, 2019

Tagged: product owner, leadership

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

Be a Great Product Owner: Six Things Teams and Scrum Masters Need Learn six ways effective product owners ensure their teams' success. Jan 31, 2023 • 16 Comments Read	From Project Manager to Scrum Master - 3 Tips for Making the Transition What does a good project manager need to do to become a great Scrum Master? Aug 23, 2022 Read	How to Create Self-Sufficient Scrum Teams Mar 01, 2022 Read	My Favorite Hard Questions to Ask When Making a Decision Asking hard questions is a great way to confirm the right decision is being made. Feb 08, 2022 Read	The Product Owner's Second Team Successful product owners will see their stakeholders as a team, not merely a set of individuals. Nov 02, 2021 Read	7 Questions to Determine if Being a Scrum Master Is Right for You Thinking of a career as a Scrum Master? 7 questions to help you decide it's the right job for ... May 18, 2021 Read
---	--	---	---	--	--

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

by
Mike Cohn
29 product owner, collaboration, stakeholders
[Comments](#)

Saying no can be very difficult. Most of us like to please others. But when we say no, we disappoint the requester.

But saying no to stakeholders is an important part of the product owner's job. The product owner is tasked with optimizing the value delivered by a product, not with saying yes to every customer request.

For every time a product owner says yes to some stakeholder's feature request, the product owner will need to say no to some future customer request. The team's time is limited and a yes today will necessitate saying no to some later opportunity.

This means that learning to say no to stakeholders is a skill every product owner needs to master. I want to share six guidelines for how you can do so politely but firmly.

1. Be Clear with Stakeholders: Is this No? Or No, for Now?

When product owners need to tell stakeholders that they are not prioritizing a particular feature request, they should be clear about what their "no" means.

If you are you saying that you'll never have the team work on this feature, don't leave the door open to

encourage the stakeholder to ask again later. That's a waste of their time and frustrating for you to have to continually say no when you know you'll never do that feature.

If on the other hand you're telling the customer no for now--that later you might work on that feature--be clear about that as well. You might, for example, say,

I'm sorry we can't work on that right now. Believe me, I wish we could. But we've already made commitments to deliver [name whatever it is] by August. I need to keep the team focused on that or we risk missing that commitment. But if you remind me about this in August, I promise to consider doing it after that.

Notice in this example reply, I didn't suggest promising you'd do it later, only that you'd consider it.

Also, notice that you put the burden back on the stakeholder or customer. Make them re-initiate the request or remind you when the time is right. I do this to make sure the feature is still important to them. If the stakeholder isn't willing to do something as minor as remind you about a feature request, I'd question whether the feature was ever very important or urgent.

Most importantly, don't let the stakeholder walk away thinking they should ask again in a month if your answer was really that you never intend.

2. Express Appreciation and Empathy for Customers

When presented with a feature request, product owners should always thank the stakeholder and indicate their understanding of why the feature is important to that stakeholder.

Someone took time to ask something of your team. That means they're interested in your product. Tell the customer that you appreciate their taking the time to ask. Something like this is all you need to say:

Thank you. I appreciate you thinking of how our product could be made better.

Beyond being appreciative, product owners should also show empathy for the stakeholder's situation. You're about to say no to a request that is likely very important to them. The feature may, in the stakeholder's mind at least, be required to fulfill goals assigned by their boss, and might even affect them financially.

Before conveying your empathy, be sure you understand why the feature is important to the customer. And if you don't understand, make sure you do before saying no to the feature request.

You can express your empathy for their situation with a statement like,

I can see why this feature is important to you in achieving [whatever it is].

Be sure you're sincere about this. I don't think I'm unique in being frustrated by false empathy.

3. Offer Only One Reason for Saying No

When saying no, it's best for product owners to provide one compelling reason rather than a list of reasons.

When offered a list of reasons, people tend to pick the weakest reason and argue against it.

Imagine I am your customer and I ask you to have your team put aside their current work in favor of a feature I want. You tell me,

I'm sorry but I can't do that. We've already planned this sprint. I'd need the team to have another planning meeting, and they won't like that. And I'm confident that what we're currently working on is higher priority.

What part of your argument do you think I'll attack? The need to do another planning meeting or that the current work is higher priority?

I'll go after the argument that the team would need to do another planning meeting and won't like it. I might offer to make the meeting less unpleasant by doing it over lunch and I'll buy the pizza.

Even if you still don't like my plan, I've now changed the conversation: We're arguing about whether to conduct a meeting rather than over the merits of the feature. That's a more difficult argument to win.

And it's the wrong basis for making the decision.

Be firm and offer your one most compelling reason for saying no. If the stakeholder successfully convinces you to change your mind by arguing that point, it may well be worth considering whether your secondary reasons for saying no are sufficient. If not, you might need to say yes to the feature.

4. Convey that You Each Have the Same Goal

If the product owner and the requesting stakeholder share the same overarching goal, the product owner should mention it when delivering unwelcome news.

A product owner and stakeholders each often have many different goals. And, yes, sometimes, they are in conflict. But usually there is a higher, product-level goal that is shared and that you can reference.

This is particularly easy if you are both part of the same company. In that case, say something like,

As much as I'd love to have the team work on that for you, we need to stay focused on [larger, shared goal].

As I'm sure you remember, we've all been given the goal to [larger, shared goal].

Reminding the customer that you share a common goal helps them understand why you need to say no, even if they still don't agree fully with the answer.

5. Explain the Consequences of Saying Yes to the Stakeholder

When rejecting a stakeholder request, product owners should explain the consequences of saying yes.

This can help the stakeholder see why you feel compelled to say no. You could, for example, say,

*If we worked on your feature instead, we won't be able to meet the big deadline.
The team is already working long hours. I can't add this to their workload without removing something else that we've already committed to someone else.*

Explaining the consequences will help the stakeholder understand, and hopefully empathize with, why you are saying no.

6. Offer the Customer an Alternative

Instead of outright saying no to a customer, a product owner may be able to offer an alternative.

You might offer something like:

We can't possibly do everything you've asked, but what if we did this subset of it?

I can't have the team work on this right now, but what if we start on it in three weeks?

Be careful: Only offer an alternative if you really mean it.

Saying No Doesn't Need to be Difficult

Product owners often fear saying no and disappointing the stakeholder or customer. But saying no doesn't have to be so difficult.

I've found that being clear, providing one reason rather than many, being empathetic and appreciative, conveying that we share the same ultimate goal, explaining the consequences of saying yes, and offering an alternative make saying no much easier.

When done well, saying no can improve, rather than harm, a product owner's relationship with the stakeholder.

What Do You Think?

How do you tell stakeholders no? Have you found certain techniques helpful or harmful? Please share your thoughts in the comments below.

Get Your Free Scrum Cheat Sheet

Sign up below to receive this FREE reference.

First Name

Email Address

[Privacy Policy](#)

Posted: June 19, 2018

Tagged: product owner, collaboration, stakeholders

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

Be a Great Product Owner: Six Things Teams and Scrum Masters Need Learn six ways effective product owners ensure their teams' success. Jan 31, 2023 • 16 Comments Read	Don't Equate Story Points to Hours I've been quite adamant lately that story points are about time, specifically effort. But that does ... Nov 22, 2022 • 39 Comments Read	7 Ways to Get and Improve Fast Feedback Here's how to get the rapid feedback you need to ensure you build the right product. Jul 26, 2022 Read	For Better Agile Planning, Be Collaborative The best plans are created by developers and stakeholders working together. Jul 12, 2022 Read	Scrum, Remote Teams, & Success: Five Ways to Have All Three In a remote-first world, how does an agile team thrive working in a virtual, distributed way? Jun 07, 2022 Read	What Does Scrum Mean by Cross Functional Teams? What exactly does cross functional mean and why does it matter? Apr 05, 2022 Read
--	--	--	---	---	---

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

by
Mike Cohn
44 Tagged:
Comments

See if this problem sounds familiar:

Your agile team runs a series of sprints, always doing the highest priority work as chosen by your product owner. But after that series of sprints, you look back at what the team accomplished. And it's not very satisfying. All the team seemed to do was move from emergency to emergency, putting out one fire after another.

The team very likely got a lot of work done. But it doesn't add up to much.

What happened? You fell into the trap of prioritizing based on what seemed most urgent at the start of each sprint, a frequent side effect of the iterative and incremental nature of agile. There's a better way.

The Agile Prioritization Trap: Selecting Work Each Iteration

Selecting whatever work seems most important at the start of each iteration can be sub-optimizing in some cases. It can lead product owners to prioritize work on the crisis du jour, whether that's

A hot tech support issue

Something that cost a sale yesterday

The latest whim of an important stakeholder

While any of these may be the most important thing to work on, all too often they are not strategic. And by choosing to work on whatever someone is screaming about in the moment, the product owner forgoes the opportunity to make progress on something bigger, more important or more strategic.

Prioritizing Without Goals Is Like Scaling Mountains without Maps

My fellow residents of Colorado and I are proud of our 53 “Fourteeners,” which are mountain peaks that are at least 14,000 feet high (4,267 meters). There are two ways to summit a Fourteener.

In the first approach, you simply drive to the base of the mountain and start hiking toward the highest thing you see. That will almost certainly be a *false peak*, which only appears to be the summit because it obscures the real summit.

But, having climbed the false peak, you can now see a higher peak. Believing it to be the true summit, you climb it. And discover that it, too, is a false peak.

You proceed this way until you finally do see the true summit. But between you and the summit is a deep valley. And to reach the 14,000-foot summit, you have to first descend 10,000 feet before climbing back up.

Clearly, this is not a good way to summit one of Colorado’s Fourteeners.

The second approach involves buying a topographic map at a local hiking store. Equipped with the map, you can then plan a more efficient route up the mountain that avoids the need to descend and re-ascend 10,000 feet.

Multi-Iteration Goals: The Product Owner’s Map

For the agile product owner, the equivalent of the topographic map is having a larger, multi-iteration goal. Without a multi-iteration goal, the product owner is merely climbing from false peak to false peak. That product owner and team may eventually arrive at their ultimate destination, but often only after descending

and climbing out of the equivalent of the 10,000-foot valley.

It's so tempting, though, to address those short-term crises instead.

First, it gives the product owner a sense of accomplishment: Something has been completed and can be ticked off the to-do list. Second, it appeases whoever is asking for the immediate solution. Most of us like to please other people, and this does that, often with only an invisible or unacknowledged impact on the bigger, longer-term, and usually more important goal.

When Prioritizing, Remember What Is Urgent Is Seldom Important

United States President Eisenhower knew how critical it is to distinguish between important matters and urgent ones. Although history is unclear on whether Eisenhower actually phrased it this way, he is often quoted as saying,

"What is important is seldom urgent, and what is urgent is seldom important."

However he phrased it, Eisenhower was saying that those urgent crises we all get tempted to work on are not usually important in terms of achieving our overarching goal.

A good product owner will be careful to consider both the importance and the urgency of product backlog items when prioritizing work for the team.

Product Owners Should Identify a Significant Objective Quarterly

But if a product owner doesn't know what it's important, the urgent will always win. And this leads to the unsatisfying series of sprints I described at the start of this post. When an agile team goes from urgent issue to urgent issue, the immediate gratification fades as the team and the stakeholders realize no progress was made toward more important goals.

This means it's vital for a product owner to define an important significant objective. I find that the best significant objectives will take the team about three months to accomplish.

Taking more than a single iteration is important because it focuses the team on a longer time horizon. This means the significant objective should be an important business result, such as migrating a portion of a system to a new technology; adding a big, important and noticeable feature; or any number of things.

But it needs to be *significant*. For a new product, it could be creating a minimum viable product (MVP). For an existing product, it might be adding a *minimum marketable feature* (MMF). Some organizations will refer to it as a *wildly important goal* (WIG).

Equipped with a significant objective—whether in the form of an MVP, MMF, or WIG—the product owner will be better able to assess urgent needs. If addressing an urgent issue is more important than working toward the significant objective, by all means the team should be directed toward the urgent issue.

The intent of establishing a significant objective is not to create a slavish devotion to a multi-iteration goal. Rather, it is to establish a mechanism to support the product owner's prioritization decisions.

Without a significant objective against which the importance of urgent issues can be compared, the urgent will always win.

What's Your Experience?

Have you experienced the unsatisfying result of a series of sprints that focused only on urgent issues? How does your product owner or team guard against favoring the urgent over important, longer-term goals?

Please share your thoughts in the comments below.

Get Your Free Scrum Cheat Sheet

Sign up below to receive this FREE reference.

First Name

Email Address

[Privacy Policy](#)

Posted: May 8, 2018

Tagged: product owner, iterations, prioritizing, iterating

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and

techniques to build extremely high-performance teams. He is the author of User Stories Applied for Agile Software Development, Agile Estimating and Planning, and Succeeding with Agile as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

[Be a Great Product Owner: Six Things Teams and Scrum Masters Need](#)

Learn six ways effective product owners ensure their teams' success.

Jan 31, 2023 • 16 Comments

[Read](#)

[How Implementation Intentions Help My Sprints](#)

Planning an exact time to do something within a sprint helps make sure you get the important stuff ...

Mar 22, 2022

[Read](#)

[The Product Owner's Second Team](#)

Successful product owners will see their stakeholders as a team, not merely a set of individuals.

Nov 02, 2021

[Read](#)

[Top 5 changes in the 2020 version of the Scrum Guide](#)

Recently the 2020 version of the Scrum Guide was released. What changes were made that you need to ...

Mar 16, 2021

[Read](#)

[\[VIDEO\] What does a virtual certified scrum course look like?](#)

Our virtual Certified Scrum courses aren't tedious 16-hour Zoom snoozefests. This video shows what ...

May 19, 2020 • [4 Comments](#)

[Read](#)

4 Steps to Persuade a Product Owner to Prioritize Refactoring

Teams often struggle to persuade their product owners to prioritize refactoring. This simple ...

Feb 04, 2020 • [8 Comments](#)

[Read](#)

Learn About Agile

New to Agile and Scrum?

Agile & Scrum Training

Online Certifications

More...

About Us

Scrum

Video Courses

Contact Us

User Stories

In-Person Certifications

Our Students

Planning Poker

On-Site Courses

Blog

Agile Software Development

Training Schedule
Compare Courses

Books by Mike Cohn
Agile Tools

Agile Project Management

PDUs and SEUs

Agile Presentations

Mountain Goat on YouTube

Agile Mentors Podcast

Elements of Agile Assessment

by
Mike Cohn
Tagged:
product owner, scaling, scaled agile, chief product owner
37 Comments

The product owner role can be one of the most challenging on a Scrum project.

On all projects, the product owner is torn between competing inward-facing and outward-facing needs. Among the inward-facing tasks are participating in planning meetings, sprint reviews, sprint retrospectives and daily scrums; managing the product backlog; answering questions from the team; and simply being available to the team during the sprint.

A product owner's outward-facing tasks include talking to users about their needs; creating and interpreting user surveys; traveling to customer sites; attending industry trade shows; managing stakeholder expectations; prioritizing the product backlog; determining product pricing; developing a medium- and long-term product strategy; watching for industry and market trends; performing competitive analysis; and more.

Too Much for One Person to Do

On a project with one team of developers, this can be a vast but achievable amount of work. On a large project with multiple teams, however, the product owner role is too big for one person, so we must find ways of scaling it.

A Note on the Term "Chief Product Owner"

The labels "chief product owner" and "product line owner" are the ones I generally favor, but they are representative only; use others if you wish. In addition to these, I've seen both program owner, super product owner, area owner and feature owner used successfully.

For consistency with the bulk of existing Scrum literature, I prefer to use "product owner" for the individual who works directly with one or two teams, prioritizing their work and doing all of the other things associated with the product owner role. In these multilayer hierarchies, the product owner is often someone whose business card reads: "Business Analyst."

As a project grows to include multiple teams, ideally a new product owner is found for each. If you cannot achieve a one-to-one correspondence between teams and product owners, try to have each product owner responsible for no more than two teams. This is usually the most that one product owner can effectively work with.

At some point, as the overall size of the project grows, it makes sense to introduce a hierarchy of collaborating product owners. The figure below shows such a hierarchy, with a product owner working with each team, two product line owners each working with a cluster of teams and a chief product owner.

Naturally, layers can be added or removed as needed for the scale of the project.

Sharing Responsibility, Dividing Functionality

A chief product owner is responsible for having an overall vision of the entire product or product suite. The chief product owner conveys this to the entire team in whole-team meetings, e-mails, team get-togethers and through whatever other means are available.

But the chief product owner is almost certainly too busy to assume hands-on responsibility for working as the actual product owner for one of the five- to nine-person teams building the product. At this level, the external-facing requirements of the role are too great.

A good chief product owner will be very involved with teams—attending daily scrums occasionally, walking through team areas whenever in the office, and offering support and feedback. But the chief product owner will need to rely on the product line owners and product owners to handle the intricate details of their product segments within the overall project vision.

An Example of a Chief Product Owner and Product Line Owner

Suppose, for example, we decide to develop an office productivity suite that will include a word processor, spreadsheet, presentation software and personal database. Competing with Microsoft Office, Google Apps and other products will be daunting, but our chief product owner is fearless.

Because the chief product owner will be focused on strategic issues, competitive positioning and such, product line owners are selected to own the individual products in the suite—the word processor, spreadsheet, presentation program and database.

Each product line owner in turn identifies product owners who will be responsible for feature areas within the product. The product line owner for the word processor, for example, may work with one product owner who is responsible for tables, another responsible for stylesheets and printing, another who is responsible for the spell checker and so on.

Although as previously mentioned, the chief product owner is too busy to be the product owner for one team, it is possible for a chief product owner to act as the product line owner for part of the product.

Continuing with the preceding example, our chief product owner may choose to also be the product line owner responsible for the word processor, perhaps because of being in that role previously.

Similarly, a product line owner will often want to stay involved in a more hands-on manner and will work also as one of the product owners. Perhaps our product line owner for the spreadsheet product also acts as the product owner for the team that will add charts to the spreadsheet product.

Although functionality can be divided along these lines, it is important for all product owners to feel a shared responsibility for the full product. They must also instill this feeling of shared responsibility in the teams they work with.

How Do You Scale Up the Product Owner Role?

How do you handle the product owner role on projects with three or more teams? Please share your thoughts in the comments below.

[Unlock module one now](#)

Posted: November 8, 2016

Tagged: product owner, scaling, scaled agile, chief product owner

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied* for Agile Software Development, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

by Roman Pichler



Published May 2016

Publisher: Pichler Consulting

Most agile processes are empty of any advice on forming a company or product strategy. Product backlogs or featurelists are just assumed to exist or to spring spontaneously from the mind of a product owner or key stakeholder. In his new book, "Strategize," Roman Pichler fills this void in agile thinking.

Roman is a long-time Scrum trainer based in the UK. He has previously written books about the overall Scrum framework and about [succeeding as a product owner](#).

In "Strategize," Pichler covers how to form and then validate a strategy, including identifying the right audience for the product and delivering just the features they need. The book covers roadmapping, including addressing the unfortunate misconception that because a team is agile, they don't need to know where they're headed.

Pichler presents a very helpful roadmap selection matrix that helps identify what type of roadmap is appropriate for different types of projects. I've already put this to use in discussions with clients. And I'm becoming convinced that if a company had a bad experience with roadmapping in the past, it was likely because of doing the wrong type of roadmapping. This book's roadmap selection matrix will fix that.

This book should be read by anyone involved in determining the future direction for a product or entire organization.

Tagged: product owner, continuous improvement, agile processes, book reviews

You may also be interested in:

[Be a Great Product Owner: Six Things Teams and Scrum Masters Need](#)

Learn six ways effective product owners ensure their teams' success.

Jan 31, 2023 • 16 Comments

[Read](#)

[Elements of Agile: An Agile Assessment Tool for Iterative Improvements](#)

New to agile, or needing an agile re-boot? We've got a new no-cost assessment and actionable ...

Sep 13, 2022

[Read](#)

The Product Owner's Second Team

Successful product owners will see their stakeholders as a team, not merely a set of individuals.

Nov 02, 2021

[Read](#)

Top 5 changes in the 2020 version of the Scrum Guide

Recently the 2020 version of the Scrum Guide was released. What changes were made that you need to ...

Mar 16, 2021

[Read](#)

[VIDEO] What does a virtual certified scrum course look like?

Our virtual Certified Scrum courses aren't tedious 16-hour Zoom snoozefests. This video shows what ...

May 19, 2020 • [4 Comments](#)

[Read](#)

4 Steps to Persuade a Product Owner to Prioritize Refactoring

Teams often struggle to persuade their product owners to prioritize refactoring. This simple ...

Feb 04, 2020 • [8 Comments](#)

[Read](#)

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile Assessment		

by Craig Larman and Bas Vodde



Published Aug 2016

Publisher: Addison-Wesley Professional

"Large-Scale Scrum" by Craig Larman and Bas Vodde is great for anyone looking to scale Scrum up to medium and large projects. It provides a contrast to the very heavyweight Scaled Agile Framework (SAFe), and "Large-Scale Scrum" comes with its own cutesy acronym, LeSS. In fact the subtitle of the book is "More with LeSS."

The book defines two scaling models. First is standard LeSS, which Larman and Vodde say is typically used for projects with around five teams. It can certainly scale beyond there. But for much larger projects, the book also defines "LeSS Huge," which the authors report having used on projects with over 1,000 people.

The book is organized as you might expect with chapters devoted to key Scrum topics such as the product owner, the product backlog, sprint planning, reviews and retrospectives, and so on.

I found the book to strike a perfect balance between being overly prescriptive and too general. You'll leave the book with plenty of advice on how to scale a Scrum project. But you won't leave feeling hamstrung by having too many rules placed on your teams.

In fact, the authors include a nice summary of LeSS and LeSS Huge rules at the end of the book, and it

covers only three pages.

Tagged: product owner, sprint planning, continuous improvement, scrum, book reviews

You may also be interested in:

[Why I Don't Emphasize Sprint Goals](#)

Sprint goals are considered a mandatory part of Scrum. Here's why I disagree.

Mar 21, 2023

[Read](#)

[Be a Great Product Owner: Six Things Teams and Scrum Masters Need](#)

Learn six ways effective product owners ensure their teams' success.

Jan 31, 2023 • [16 Comments](#)

[Read](#)

[Agile Mentors Podcast: Recap of Opening Series on Scrum](#)

Recapping our agile podcast's initial launch series of topics

Sep 27, 2022

[Read](#)

[Elements of Agile: An Agile Assessment Tool for Iterative Improvements](#)

New to agile, or needing an agile re-boot? We've got a new no-cost assessment and actionable ...

Sep 13, 2022

[Read](#)

3 Ways to Help Agile Teams Plan Despite Uncertainty

We might not like ambiguity, but it's a fact of life. Find out how to plan with uncertainty in mind.

Jun 21, 2022

[Read](#)

How Implementation Intentions Help My Sprints

Planning an exact time to do something within a sprint helps make sure you get the important stuff ...

Mar 22, 2022

[Read](#)

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile Assessment		

by
Mike Cohn **Tagged:** product owner, meetings, stakeholders, sprint review
36 Comments

Some teams use the [sprint review](#) as a time for product owners or key stakeholders to formally approve the product backlog items completed during the sprint. Is this a good idea?

In general, a sprint review should not be used by a team to get formal sign-off on their work from their product owner. The team and product owner should be working so closely during a sprint that the team knows what the product owner thinks of what they've built.

No surprises is my No. 1 rule for the sprint review.

It is absolutely acceptable for a product owner to reject the work of a team on a product backlog item. But the team should know that's coming.

Team members should not walk into a sprint review expecting glowing praise from the product owner but then be blindsided by a litany of complaints about a feature.

But what about acceptance by a client? Can a sprint review be used for formal sign-off or acceptance in those cases?

Ideally, in cases in which a client hires a vendor to develop a product, someone at the client company would act as the product owner. And in those cases, it can be OK for formal sign-off on features to occur

during the sprint review. But I'd still stick with the advice that there should be no surprises during the review.

Even though the client product owner is providing feedback to the team during the sprint, it's possible that the product owner needs to wait to fully accept something until other stakeholders have a chance to comment on the work.

As a simple example, my daughter recently asked me if she could go on a school trip. I said it was fine with me, but--guess what--we needed to check that it was OK with her mother. That is, my wife might have had plans for our family during that time that I didn't yet know about.

This will be a common situation for client product owners in contract development situations. The product owner interacting with the team daily may like how a feature has been built, but may need to confirm that the stakeholders he or she represents agree. Sure, we can say that the product owner should simply go ask. But that can be impractical and might best be done in a sprint review.

But in outsourced, contract development, the client doesn't always provide the product owner. Many times, the client hires the vendor to take care of everything.

The client is, of course, the true product owner. The client will ultimately accept or reject what is developed. But, on a day-to-day basis, the client doesn't want to be "bothered." And so the typical solution in this case is for the vendor to appoint a product owner from someone within its own organization.

And in this case, true acceptance (or "sign off") on product backlog items cannot happen before the sprint review. The true product owner (from the client) is not sufficiently available and engaged to accept things any more frequently.

Sure, the team may have a preliminary sign-off from their own product owner representative during the sprint. But the true, client product owner may completely reverse that decision in the actual sprint review.

So the ultimate answer depends, like so many things, upon the context in which you're operating. And so I'll say that I'm not too concerned by actual, formal sign-off occurring during a sprint review. But I always want to stick with a policy of no surprises during the review.

Sign off or not, as needed. But the team should always have a good idea of what's coming before they get to the review.

What Do You Do?

What does your team do in sprint reviews? Has the product owner largely seen everything before then? Are product backlog items formally accepted during the review? Please share your thoughts in the comments below.

Get Your Free Scrum Cheat Sheet

Sign up below to receive this FREE reference.

First Name

Email Address

We hate spam and promise to keep your email address safe. Unsubscribe at any time. [Privacy Policy](#)

Posted: September 27, 2016

Tagged: product owner, meetings, stakeholders, sprint review

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

Be a Great Product Owner: Six Things Teams and Scrum Masters Need

Learn six ways effective product owners ensure their teams' success.

Jan 31, 2023 • 16 Comments [Read](#)

What Happens When During a Sprint

Succeeding with Scrum is easier when you know when and why to conduct each of the Scrum events ...

Jan 03, 2023 • 34 Comments [Read](#)

Agile Mentors Podcast: Recap of Opening Series on Scrum

Recapping our agile podcast's initial launch series of topics

Sep 27, 2022 [Read](#)

7 Ways to Get and Improve Fast Feedback

Here's how to get the rapid feedback you need to ensure you build the right product.

Jul 26, 2022 [Read](#)

Seven Tips for Running a Virtual Daily Scrum

How do you effectively facilitate a daily scrum with a remote team?

May 03, 2022 [Read](#)

Top 7 Ways to Get Stakeholders to Attend Sprint Reviews

Poorly attended sprint reviews cause problems. Fortunately there are easy fixes.

Mar 08, 2022 [Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile **Agile & Scrum Training** **More...**

New to Agile and Scrum? Online Certifications About Us

Scrum Video Courses Contact Us

User Stories In-Person Certifications Our Students

Planning Poker On-Site Courses Blog

Agile Software Development Training Schedule Books by Mike Cohn

Development Compare Courses Agile Tools

Agile Project Management PDUs and SEUs

Agile Presentations

Mountain Goat on YouTube

Agile Mentors Podcast

Elements of Agile

Assessment

by

Mike Tagged:

Cohn user stories, product owner, teams, sprinting, agile, iterative,
118 waterfall, overlap work

Comments

I've noticed something disturbing over the past two years. And it's occurred uniformly with teams I've worked with all across the world. It's the tendency to create an iterative waterfall process and then to call it agile.

An iterative waterfall looks something like this: In one sprint, someone (perhaps a business analyst working with a product owner) figures out what is to be built.

Because they're trying to be agile, they do this with [user stories](#). But rather than treating the user stories as short placeholders for future conversations, each user story becomes a mini-specification document, perhaps three to five pages long. And I've seen them longer than that.

These mini-specs/user stories document nearly everything conceivable about a given user story.

Because this takes a full sprint to figure out and document, a second sprint is devoted to designing the user interface for the user story. Sometimes, the team tries to be a little more agile (in their minds) by starting the design work just a little before the mini-spec for a user story is fully written.

Many on the team will consider this dangerous because the spec isn't fully figured out yet. But, what the heck, they'll reason, this is where the agility comes in.

Programmers are then handed a pair of documents. One shows exactly what the user story should look like when implemented, and the other provides all details about the story's behavior.

No programming can start until these two artifacts are ready. In some companies, it's the programmers who force this way of working. They take an attitude of saying they will build whatever is asked for, but you better tell them exactly what is needed at the start of the sprint.

Some organizations then stretch things out even further by having the testers work an iteration behind the programmers. This seems to happen because a team's user stories get larger when each user story needs to include a mini-spec and a full UI design before it can be coded.

Fortunately, most teams realize that programmers and testers need to work together in the same iteration, but not extend that to being a whole team working together. This leads to the process shown in this figure.



This figure shows a first iteration devoted to analysis. A second iteration (possibly slightly overlapping with the first) is devoted to user experience design. And then a third iteration is devoted to coding and testing. This is not agile. It might be your organization's first step toward becoming agile. But it's not agile.

What we see in this figure is an iterative waterfall.

In traditional, full waterfall development, a team does all of the analysis for the entire project first. Then they do all the design for the entire project. Then they do all the coding for the entire project. Then they do all the testing for the entire project.

In the iterative waterfall of the figure above, the team is doing the same thing but they are treating each story as a miniature project. They do all the analysis for one story, then all the design for one story, then all the coding and testing for one story. This is an iterative waterfall process, not an agile process.

Ideally, in an agile process, all types of work would finish at exactly the same time. The team would finish analyzing the problem at exactly the same time they finished designing the solution to the problem, which would also be the same time they finished coding and testing that solution. All four of those disciplines (and any others I'm not using in this example) would all finish at exactly the same time.

It's a little naive to assume a team can always perfectly achieve that. (It can be achieved some times.) But it

can remain the goal a team can work towards.

A team should always work to overlap work as much as possible. And upfront thinking (analysis, design and other types of work) should be done as late as possible and in as little detail as possible while still allowing the work to be completed within the iteration.

If you are treating your user stories as miniature specification documents, stop. Start instead thinking about each as a promise to have a conversation.

Feel free to add notes to some stories about things you want to make sure you bring up during that conversation. But adding these notes should be an optional step, not a mandatory step in a sequential process.

Leaving them optional avoids turning the process into an iterative waterfall process and keeps your process agile.

Download Scrum Master Guide

Get a free copy of Situational Scrum Mastering: Leading an Agile Team

[Get my free guide now!](#)

Posted: August 25, 2015

Tagged: user stories, product owner, teams, sprinting, agile, iterative, waterfall, overlap work

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com.

If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

User Stories: How to Create Story Maps

Story maps help to create a shared understanding of the product, visualize user needs, and elicit ...

Mar 14, 2023

[Read](#)

How to Run a Successful User Story Writing Workshop

What you need to know to conduct a story-writing workshop with your team.

Feb 28, 2023 • [3 Comments](#)

[Read](#)

What Is Cross-Functional Collaboration in Agile?

Perhaps the most prevalent and persistent myth in agile is that a cross-functional team is one on ...

Feb 14, 2023 • [49 Comments](#)

[Read](#)

[Be a Great Product Owner: Six Things Teams and Scrum Masters Need](#)

Learn six ways effective product owners ensure their teams' success.

Jan 31, 2023 • [16 Comments](#)

[Read](#)

[What Happens When During a Sprint](#)

Succeeding with Scrum is easier when you know when and why to conduct each of the Scrum events ...

Jan 03, 2023 • [34 Comments](#)

[Read](#)

Don't Equate Story Points to Hours

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • [39 Comments](#)

[Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

Prioritize and Optimize Over a Slightly Longer Horizon

by
Mike Cohn **Tagged:**
20 product owner, sprint planning, planning, prioritizing, optimize
[Comments](#)

A lot of agile literature stresses that product owners must prioritize the delivery of value. I'm not going to argue with that. But I am going to argue that product owners need to optimize over a slightly longer horizon than a single sprint.

A product owner's goal is to maximize the amount of value delivered over the life a product. If the product owner shows up at each sprint planning meeting focused on maximizing the

value delivered in only that one sprint, the product owner will never choose to invest in the system's future.

The product owner will instead always choose to deliver the feature that delivers the most value in the immediate future regardless of the long-term benefit. Such a product owner is like the pleasure-seeking student who parties every night during the semester but fails, and has to repeat the course during the summer.

A product owner with a short time horizon may have the team work on features A1, B1, C1 and D1 in four different parts of the application *this* sprint because those are the highest valued features.

A product owner with a more appropriate, longer view of the product will instead have the team work on A1, A2, A3 and A4 in the first sprint because there are synergies to working on all the features in area A at the same time.

Agile is still about focusing on value. And it's still about making sure we deliver value in the short term. We just don't want to become shortsighted about it.

Get Your Free Scrum Cheat Sheet

Sign up below to receive this FREE reference.

First Name

Email Address

[Privacy Policy](#)

Posted: June 30, 2015

Tagged: product owner, sprint planning, planning, prioritizing, optimize

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the *Better User Stories* video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile Assessment		

by
Mike Tagged:
Cohn product owner, teams, collaboration, prioritizing, skillset,
27 dependencies, order
Comments

A product owner hands 10 story cards to the team. The team reads them and hands the fifth and sixth cards back to the product owner. By the end of the sprint, the team delivers the functionality described on cards 1, 2, 3, 4, and 7. But the team has not touched the work of cards 5 and 6.

And I say this is OK.

Standard agile advice is that a team should work on product backlog items in the order prescribed by the product owner. And although this is somewhat reasonable advice, I want to argue that good agile teams violate that guideline all the time.

There are many good reasons for a team to work out of order. Let's consider just a few, and consider it sufficient evidence that teams should be allowed to work out of order.

1. Synergies

There are often synergies between items near the top of a product backlog—while a team is working on item No. 3, they should be allowed to work on No. 6. If two items are in the same part of the system and can be done faster together than separately, this is usually a good tradeoff for a product owner to allow.

2. Dependencies

A team may agree that No. 4 is more important than No. 5 and No. 6 on the product backlog. Unfortunately, No. 4 can't be done until No. 7 has been implemented. Finding a dependency like this is usually enough to justify a team working a bit out of the product owner's prioritized sequence.

3. Skillset Availability

A team might love to work on the product owner's fourth top priority, but the right person for the job is unavailable. Sure, this can be a sign that additional cross-training is needed on that team to address this problem – but, that is often more of a long-term solution. And, in the short term, the right thing to do might simply be to work a bit out of order on something that doesn't require the in-demand skillset.

4. It's More Exciting

OK, this one may stir up some controversy. I'm not saying the team can do 1, 2, 3, 4, and then number 600. But, in my example of 1, 2, 3, 4 and 7, choosing to work on something because it's more exciting to the team is OK.

On some projects, teams occasionally hit a streak of product backlog items that are, shall we say, less than exciting. Letting the team slide slightly ahead sometimes just to have some variety in what they're doing can be good for the morale of the team. And that will be good for product owners.

Bonus Reason 4a: It's More Exciting to Stakeholders

While on the subject of things being more exciting, I'm going to say it is also acceptable for a team to work out of order if the item will be more exciting to stakeholders.

It can sometimes be a challenge to get the right people to attend sprint reviews. It gets especially tough after a team runs a few boring ones in a row. Sometimes this happens because of the nature of the high-priority work—it's stuff that isn't really visible or is esoteric perhaps to stakeholders.

In those cases, it can be wise to add a bit of sex appeal to the next sprint review by making sure the team works on something stakeholders will find interesting and worth attending the meeting for.

5. Size

In case the first four (plus) items haven't convinced you, I've saved for last the item that proves every team occasionally works out of product owner order: A team may skip item 5 on the product backlog because it's too big. So they grab the next one that fits.

If a team were not to do this, they would grab items 1, 2, 3, and 4 and stop, perhaps leaving a significant portion of the sprint unfilled. Of course, the team could look for a way to perhaps do a portion of item 5 before jumping down to grab number 7. But sometimes that won't be practical, which means the team will at least occasionally work out of order.

Product Owners Aren't Perfect

A perfect product owner would know everything above. The perfect product owner would know that the team's DBA will be full from tasks from the first four product backlog items, and so wouldn't put a database-intensive task fifth. The perfect product owner would know that items 2 and 5 both affect the same Java class, and would naturally prioritize them together.

But most product owners have a hard time being perfect. A better solution is for them to put the product backlog in a pretty good prioritized order, and then leave room for fine tuning from the team.

Download Scrum Master Guide

Get a free copy of Situational Scrum Mastering: Leading an Agile Team

[Get my free guide now!](#)

Posted: December 16, 2014

Tagged: product owner, teams, collaboration, prioritizing, skillset, dependencies, order

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

What Is Cross-Functional Collaboration in Agile?

Perhaps the most prevalent and persistent myth in agile is that a cross-functional team is one on ...

Feb 14, 2023 • 49 Comments [Read](#)

Be a Great Product Owner: Six Things Teams and Scrum Masters Need

Learn six ways effective product owners ensure their teams' success.

Jan 31, 2023 • 16 Comments [Read](#)

Don't Equate Story Points to Hours

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • 39 Comments [Read](#)

From Project Manager to Scrum Master -3 Tips for Making the Transition

What does a good project manager need to do to become a great Scrum Master?

Aug 23, 2022 [Read](#)

For Better Agile Planning, Be Collaborative

The best plans are created by developers and stakeholders working together.

Jul 12, 2022 [Read](#)

Relationship between Definition of Done and Conditions of Satisfaction

I'd like to clarify the relationship between two important concepts: a team's Definition of Done ...

Jun 28, 2022 • 38 Comments [Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile

New to Agile and Scrum?

Scrum

User Stories

Planning Poker

Agile Software Development

Agile Project Management

Agile Presentations

Mountain Goat on YouTube

Agile Mentors Podcast

Elements of Agile

Assessment

Agile & Scrum Training

Online Certifications

Video Courses

In-Person Certifications

On-Site Courses

Training Schedule

Compare Courses

PDUs and SEUs

More...

About Us

Contact Us

Our Students

Blog

Books by Mike Cohn

Agile Tools

Who Picks the Sprint Length on a Scrum Team?

by

Mike Tagged:

Cohn product owner, sprinting, scrum master, consensus, sprint length,
35 process
Comments

An important consideration for every Scrum team is how long its sprints should be. Choose a length that's too long, and it will be hard to keep change out of the sprint. Choose a length that's too short, and a team may struggle with completing significant work within the sprint or weaken their definition of done to do so.

But who is it that gets to select a team's sprint length?

Of course, the answer is the *whole team* – that collective of ScrumMaster plus product owner plus team members such as programmers, testers, designers, DBAs, analysts and so on.

But what if that broad set of individuals cannot agree? Do they argue endlessly, perhaps sticking with their waterfall or ad hoc process until consensus finally emerges?

No. The ScrumMaster is ultimately the one who gets to choose a team's sprint length.

A good ScrumMaster will do everything possible to arrive at a consensus. But, when the ScrumMaster exhausts his or her collaborative, facilitative skills without arriving at a consensus, the good ScrumMaster makes the decision.

This should not happen often. I hope most ScrumMasters never need to say, "I've listened to everyone, but here's what we're doing." But since the ScrumMaster can be considered a team's *process owner*, the ultimate decision does belong to the ScrumMaster.

Let's consider one example from my past.

In this case, I was consulting to a team doing four-week sprints. They were struggling to pull the right amount of work into their sprints. For the six months before I'd met them, team members were consistently dropping about a third of the work each sprint.

They were a good team doing high-quality work. They simply didn't know how much of it they could do in four weeks. Their optimism was getting the better of them, and they were consistently overcommitting.

I asked the team to think about how they'd like to solve the problem and tell me their suggestions the next day. I was thrilled the next day when they announced that they should clearly change the length of their sprints. "Yes," I told them, "Definitely."

They were relieved that I agreed with them and said so: "Wow! We didn't think you'd let us go to six-week sprints!"

I had to inform them that while I agreed with changing sprint length, the better solution would be to go to shorter rather than longer sprints.

We ended with me—as a consulting ScrumMaster—setting a two-week length.

Why did I do this?

The team was already pulling too much work into a four-week sprint. They were, in fact, probably pulling six weeks of work into each four-week sprint. But, if they had gone to a six-week sprint, they probably would have pulled eight or nine weeks of work into those!

This team needed more chances to learn how much work fit into a sprint (of any length). As their ScrumMaster—especially coming to the team as a consultant—I could see that more easily than they could.

I want to end by repeating my caution that this is not something that should happen very often. I can count on one hand the number of times when I've flat-out chosen a length for the team after failing to gain consensus.

I'll stand by the value of having done so in each case. But I only did so after significant effort to gain consensus. Overriding team members on something as important as sprint length should be done with great caution. But, it's a process issue and, therefore, in the domain of the ScrumMaster.

What about you? Were there times when your team couldn't agree on a sprint length? How did you resolve it?

Get Your Free Scrum Cheat Sheet

Sign up below to receive this FREE reference.

First Name

Email Address

We hate spam and promise to keep your email address safe. Unsubscribe at any time. [Privacy Policy](#)

Posted: December 9, 2014

Tagged: product owner, sprinting, scrum master, consensus, sprint length, process

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the *Better User Stories* video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

Be a Great Product Owner: Six Things Teams and Scrum Masters Need

Learn six ways effective product owners ensure their teams' success.

Jan 31, 2023 • 16 Comments [Read](#)

Six Attributes of a Great ScrumMaster

Scrum Masters exhibit a shared set of traits. Discover 6 in-demand qualities, plus a bonus ...

Jan 17, 2023 • 32 Comments [Read](#)

What Happens When During a Sprint

Succeeding with Scrum is easier when you know when and why to conduct each of the Scrum events ...

Jan 03, 2023 • 34 Comments [Read](#)

From Project Manager to Scrum Master -3 Tips for Making the Transition

What does a good project manager need to do to become a great Scrum Master?

Aug 23, 2022 [Read](#)

Seven Tips for Running a Virtual Daily Scrum

How do you effectively facilitate a daily scrum with a remote team?

May 03, 2022 [Read](#)

What Does Scrum Mean by Cross Functional Teams?

What exactly does cross functional mean and why does it matter?

Apr 05, 2022 [Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile

New to Agile and Scrum?

Scrum

User Stories

Planning Poker

Agile Software Development

Agile Project Management

Agile Presentations

Mountain Goat on YouTube

Agile Mentors Podcast

Elements of Agile

Assessment

Agile & Scrum Training

Online Certifications

Video Courses

In-Person Certifications

On-Site Courses

Training Schedule

Compare Courses

PDUs and SEUs

More...

About Us

Contact Us

Our Students

Blog

Books by Mike Cohn

Agile Tools

ScrumMasters Should Not Also Be Product Owners

by
Mike Cohn **Tagged:**
product owner, teams, scrum master, scrum roles, skills, focus,
28 personalities
[Comments](#)

Hey, ScrumMaster: Step away from the index cards! You should not be a team's product owner if you are also the team's ScrumMaster.

Different individuals should fill these two roles, and there are many reasons for this. Let's consider a few of them in this post.

Each Has a Different Focus

First, a product owner and ScrumMaster are focused on different aspects of a Scrum project. The product owner spends his or her time thinking about what to build. That should largely be determined independent of the capabilities of the team, which are the concern of the ScrumMaster.

That is, while a product owner is determining what to build, the ScrumMaster is helping the team work together so they can.

In many ways, this can be thought of as similar to the idea that a team's programmer and tester should be separate. Sure, a good programmer can test and a good tester can program. But, separating those roles is usually a good idea.

Each Is Probably Pretty Busy

Second, it is quite likely that being either ScrumMaster or product owner requires full- or near full-time attention. Putting one person in both roles at the same time will almost certainly shortchange one of the two.

The Two Roles Require Different Personalities

There is some overlap in the skills and personality traits that make good product owners and ScrumMasters. However, the roles are different, and it is extremely unlikely that someone will be able to excel at both, especially at the same time.

I've written [elsewhere](#) about how the different types of tasks each performs makes this so.

A Natural Tension Exists Between the Roles

Additionally, a natural tension should exist between the two roles. Although each is undeniably committed to the success of the product or system being developed, product owners naturally want more, more, more.

ScrumMasters, on the other hand, are more attuned to the issues that can arise from a team under undue pressure to deliver more, more, more. When a balance exists between the roles, a product owner is free to follow his or her natural tendency to ask for more, safe with the assurance that the ScrumMaster is there to prevent pushing too hard.

Are There Ever Exceptions?

Certainly. I've encountered many situations in which the ScrumMaster and product owner were the same person, and where I felt that was appropriate. Some of these have included small organizations that could not afford the luxury of dedicated or separate individuals.

Other situations were small teams who had started in the pursuit of a technical product owner's vision. On such small teams, any one personality can have an outsized effect on the team, regardless of any formal role played by the person.

Other exceptions have been ScrumMasters involved in contract development. It is common on such a project for the "true product owner" to exist within the client asking for the software to be built.

Unfortunately, it is also common for such true product owners not to want to be deeply involved in the project at the level a Scrum team needs. It is in these cases that a good ScrumMaster often steps up and into the role as a proxy for that true product owner.

So, sure there are exceptions—just like there are to any rule. However, none of those exceptions should exist for the long term. And anyone in both roles simultaneously should be aware of the challenges the dual role presents.

[Take the Assessment](#)

Posted: December 2, 2014

Tagged: product owner, teams, scrum master, scrum roles, skills, focus, personalities

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and*

Planning, and Succeeding with Agile as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

What Is Cross-Functional Collaboration in Agile?

Perhaps the most prevalent and persistent myth in agile is that a cross-functional team is one on ...

Feb 14, 2023 • 49 Comments [Read](#)

Be a Great Product Owner: Six Things Teams and Scrum Masters Need

Learn six ways effective product owners ensure their teams' success.

Jan 31, 2023 • 16 Comments [Read](#)

Six Attributes of a Great ScrumMaster

Scrum Masters exhibit a shared set of traits. Discover 6 in-demand qualities, plus a bonus ...

Jan 17, 2023 • 32 Comments [Read](#)

Don't Equate Story Points to Hours

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • 39 Comments [Read](#)

From Project Manager to Scrum Master -3 Tips for Making the Transition

What does a good project manager need to do to become a great Scrum Master?

Aug 23, 2022 [Read](#)

Relationship between Definition of Done and Conditions of Satisfaction

I'd like to clarify the relationship between two important concepts: a team's Definition of Done ...

Jun 28, 2022 • 38 Comments [Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile **Agile & Scrum Training** **More...**

New to Agile and Scrum? Online Certifications About Us

Scrum Video Courses Contact Us

User Stories In-Person Certifications Our Students

Planning Poker On-Site Courses Blog

Agile Software Development Training Schedule Books by Mike Cohn

Development Compare Courses Agile Tools

Agile Project Management PDUs and SEUs

Agile Presentations

Mountain Goat on YouTube

Agile Mentors Podcast

Elements of Agile

Assessment

The Agile Household: How Scrum Made Us a Better Family

by
Martin Lapointe Tagged:
18 product owner, sprinting, backlog, lean
[Comments](#)

I'm always fascinated by stories about Scrum (or any agile process) being used outside of software development. When Martin Lapointe told me how he and his family used Scrum -- and especially a task board -- to manage their recent relocation from Paris to Montreal, I immediately asked him to share that story. I'm sure you'll find it as interesting, amusing, and informative as I did. - Mike Cohn

Ever since discovering the “Agile Manifesto,” I have been trying to integrate its core set of values into my day-to-day routines in hopes of improving processes outside of the office environment. With this in mind, my family and I

embarked on an agile adventure that produced amazing results we never expected!

Since my childhood, I have longed to live and experience life in a different country. I have always been especially interested in exploring Europe in hopes of better understanding the many different cultures that make it such an amazing place. This dream had always been on the back burner, and then in 2011, with the help of my wife, Pascale, and my two girls, Elisabeth and Sarah (8 and 5 years old), we decided to make it a reality.

Carpe Diem (Seize the Day)!

With our family mantra in mind, we picked up, left Montreal and migrated to Paris, France.

As expected, when displacing a family of four from a three-story house to a small Parisian apartment, the move was exhausting and quite chaotic. Almost right off the plane, I started my new job as a ScrumMaster at a telecom company, and meanwhile, Pascale embarked on her new role as “Product Owner” of our household. Our two girls were rapidly immersed into the Parisian lifestyle and school system.

*"With our family in mantra in mind,
we picked up, left Montreal and
migrated to Paris, France."*

The next two years flew by at light speed! Before we knew it, it was time to start planning our return to Canada.

Looking back on our initial move to Paris, we wish we had better prepared ourselves and been more organized as a family while facing this major life change. In the face of another move, we began thinking of ways to approach yet another life changing experience.

Our hearts filled with emotions as we started compiling our to-do lists of Post-its that had to be completed before D-day. As the tasks piled up, we started to wonder how we could possibly get all of this done while trying to maintain a balance of work and living a happy life until the end of our European adventure.

What if Scrum Was the Answer to Our Challenge?

Then, one night, we said to ourselves let's try something different. What if Scrum was the answer to our challenge? In the agile world, we try to leverage experience and failures to improve, so why not use the same approach to our big move?

Having some positive experiences experimenting with Scrum on a non-IT related team, I said to myself, *why not take the same approach with my family?* So, we gathered in our bedroom with our Post-its and sharpies in hand, and said, let's create a backlog!

Elisabeth, being the curious one, immediately asked me, “Daddy, what's a backlog?” I responded by explaining that a backlog is everything we need to do before leaving Paris. Elisabeth quickly asked, “Can I add the Eiffel tower carrousel to the backlog?” Of course, we said yes!

Mom then asked, "Can we also add taking out the trash?" I said yes, of course! Within seconds, everybody was writing valid stories down on Post-its. Sarah drew her story, because she hadn't learned to write yet: a picture of her favourite Parisian sweets, Pierre Hermé macarons!

In one evening we compiled more than 50 family stories. These stories weren't perfect by any means. There were no acceptance criteria, no estimations, but the girls were on board, and dare I say it, excited, and that was more than enough!

Next, we asked ourselves: what can we realistically accomplish in a week? Mom wanted to add the entire house cleaning tasks to the backlog. The girls wanted to add all of the fun stuff, and I wanted to add the must-see tourist attractions we hadn't yet visited.

So we took a small board and made three columns: "To do," "Ongoing" and "Done." We negotiated ruthlessly, but ultimately Elisabeth and Sarah got the most stories in (don't worry! The parents got their revenge in the following iterations ☺). The first iteration was about to begin.

"We had never seen this much work get done so fast, with so much happiness, ease, understanding and visibility!"

The Scrum momentum was on. We agreed upon one-week iterations (sprints) and then took time on the weekends to plan the next iteration. Each morning, we would have a quick gathering (daily stand-up) and the girls were very anxious to move the Post-its around the board. We had never seen so much work get done so fast, with so much happiness, ease, understanding and visibility!

With the Scrum approach, we were able to get the entire apartment-cleaning task list done, administrative tasks complete and some great museum visits in without any conflicts. Scrum was turning out to be a great tool for our family to use when trying to improve clarity and set priorities for big challenges!

We had the joy of experiencing our last days in Paris in a way that we will never forget.

Scrum Was Turning Out to be a Great Tool for Our Family

Back in Montreal, it seemed as though something was missing ... It was Scrum! Realising this, we started a new backlog in the basement, and added a cork Scrum board in the kitchen. Our activities are now visible and updated each morning at the breakfast table.

If there's one overlying factor that we've taken away from this experience it's that we are so proud to be an agile family!

Get 200 Real Life User Stories Examples Written by Mike Cohn

See user stories Mike Cohn wrote as part of several real product backlogs.

First Name

Email Address

We hate spam and promise to keep your email address safe. Unsubscribe at any time. [Privacy Policy](#)

Posted: August 20, 2014

Tagged: product owner , sprinting , backlog , lean

About the Author

Martin Lapointe is an agile leader at Yellow Pages Group in Montreal, Canada. With certifications as a CSM, CSPO and PMP, Martin has a solid mastery of agile and Scrum project techniques.

In his role as an agile servant leader, Martin exercises his fervor for communication, making initiatives work, creating interactions, generating collaboration and organizing change positively. Martin applies agile values to both his work environment, and personal and family life. You can connect with Martin on Twitter [@mdelapointe](#).

You may also be interested in:

[Be a Great Product Owner: Six Things Teams and Scrum Masters Need](#)

Learn six ways effective product owners ensure their teams' success.

Jan 31, 2023 • 16 Comments

[Read](#)

[What Happens When During a Sprint](#)

Succeeding with Scrum is easier when you know when and why to conduct each of the Scrum events ...

Jan 03, 2023 • [34 Comments](#)

[Read](#)

[What Are Agile Story Points?](#)

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022 • [122 Comments](#)

[Read](#)

[Four Reasons Agile Teams Estimate Product Backlog Items](#)

Estimating product backlog items provides benefits beyond predicting when a project will be finished.

May 17, 2022

[Read](#)

[How Implementation Intentions Help My Sprints](#)

Planning an exact time to do something within a sprint helps make sure you get the important stuff ...

Mar 22, 2022

[Read](#)

The Product Owner's Second Team

Successful product owners will see their stakeholders as a team, not merely a set of individuals.

Nov 02, 2021

[Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

Ray Bradbury on the Benefits of Short Releases

by
Mike Cohn
Tagged:
user stories, product owner, customers
8 Comments

In 2001, author [Ray Bradbury gave a talk](#) at the annual Writer's Symposium by the Sea in San Diego. Fortunately for those of us who were not there, his speech was video recorded and is available.

Bradbury—the author of books such as *Fahrenheit 451*, *The Martian Chronicles*, *Something Wicked This Way Comes*—was offering advice to an audience mostly of aspiring writers. But I watched his speech recently and was struck by the appropriateness of Bradbury's

advice to those of us on agile project.

Early in his talk, Bradbury recommends that aspiring writers write short stories rather than novels. He says that writing a novel at the start of one's career is a mistake because "you could spend a whole year writing one, and it might not turn out well because you haven't learned to write yet."

Bradbury advises instead writing "a hell of a lot of short stories," suggesting writing one per week. He says "at the end of a year, you have 52 short stories, and I defy you to write 52 bad ones."

It "can't be done," Bradbury jokes, saying that with that much practice, the aspiring writer will eventually come up with something good.

So I'll go along with Bradbury, but instead of writing short stories, let's put out new features. Let's give our customers one release per week for the next year. And, like Bradbury, I defy you to put out 52 bad releases.

Sure, some of your releases will have things your users don't want. You thought your users would want them, some minimal amount of research may have shown they would, but when you put that release out, you found out otherwise. What you can't do, though, is put out 52 releases and fail to learn from them.

Like Bradbury's aspiring writers, you'll learn more about your customers, your product, and your team. And, like Ray Bradbury, that will help you find your bestseller.

Even if you don't have time to listen to Bradbury's full speech (it's 54 minutes), listen to the first 6 minutes. He makes at least one other point very relevant to the short iterations of agile. If you listen, let me know in the comments below, what else you heard that's relevant to agile.

Get 200 Real Life User Stories Examples Written by Mike Cohn

See user stories Mike Cohn wrote as part of several real product backlogs.

First Name

Email Address

[Privacy Policy](#)

Posted: July 29, 2014

Tagged: user stories, product owner, customers

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the *Better User Stories* video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

User Stories: How to Create

Story Maps

Story maps help to create a shared understanding of the product, visualize user needs, and elicit ...

Mar 14, 2023

[Read](#)

How to Run a Successful User Story Writing Workshop

What you need to know to conduct a story-writing workshop with your team.

Feb 28, 2023 • 3 Comments [Read](#)

Be a Great Product Owner: Six Things Teams and Scrum

Masters Need

Learn six ways effective product owners ensure their teams' success.

Jan 31, 2023 • 16 Comments [Read](#)

Epics, Features and User Stories

I've been getting more and more emails lately from people confused about the difference between ...

Nov 08, 2022 • 93 Comments [Read](#)

Advantages of User Stories over Requirements and Use Cases

At the surface, user stories appear to have much in common with use cases and traditional ...

Oct 05, 2022 • 41 Comments [Read](#)

Relationship between Definition of Done and Conditions of Satisfaction

I'd like to clarify the relationship between two important concepts: a team's Definition of Done ...

Jun 28, 2022 • 38 Comments [Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Sorry, the browser you are using is not currently supported. Disqus actively supports the following browsers:

[Firefox](#)
[Chrome](#)
[Internet Explorer 11+](#)
[Safari](#)

[Bijay Jayaswal](#) • 8 years ago

This talk is simply awesome. I wish this was available some 15 years ago when I had not written a single book or a user story. Now every Agile practitioner has to learn the art of writing good user stories iteration after iteration. How do you write plenty of good user stories in your Agile team? Here is what I would suggest:

1. Encourage everyone in your team from Product Owner, Developers to ScrumMaster to write.
2. Post every development/fix idea as a user story in your product backlog the earliest possible.
3. Make sure that all the user stories are supported by Acceptance Criteria and DOD, and
4. Get early reviews of a user story from the stakeholders to determine its value.

Mike Cohn • 8 years ago

Hi Bijay—

··· ··· ··· ··· ··· ···

Learn About Agile **Agile & Scrum Training** **More...**

- New to Agile and Scrum?
 - Scrum
 - User Stories
 - Planning Poker
 - Agile Software Development
 - Agile Project Management
 - Agile Presentations
 - Mountain Goat on YouTube
 - Agile Mentors Podcast
 - Elements of Agile
 - Assessment
- Online Certifications
 - Video Courses
 - In-Person Certifications
 - On-Site Courses
 - Training Schedule
 - Compare Courses
 - PDUs and SEUs
- About Us
 - Contact Us
 - Our Students
 - Blog
 - Books by Mike Cohn
 - Agile Tools

Teams Should Go So Fast They Almost Spin Out of Control

by
Mike Cohn Tagged:
19 product owner, teams, teamwork, testing
[Comments](#)

Yes, I really did refer to guitarist Alvin Lee in a Certified Scrum Product Owner class last week. Here's why.

I was making a point that Scrum teams should strive to go as fast as they can without going so fast they spin out of control. Alvin Lee of the band Ten Years After was a talented guitarist known for his very fast solos.

Lee's ultimate performance was of the song "[I'm Going Home](#)" at Woodstock. During the performance, Lee was frequently on the edge of

flying out of control, yet he kept it all together for some of the best 11 minutes in rock history.

I want the same of a Scrum team—I want them going so fast they are just on the verge of spinning out of control yet are able to keep it together and deliver something classic and powerful.

Re-watching Ten Years After's Woodstock performance I'm struck by a couple of other lessons, which I didn't mention in class last week:

One: Scrum teams should be characterized by frequent, small hand-offs. A programmer gets eight lines of code working and yells, "Hey, Tester, check it out." The tester has been writing automated tests while waiting for those eight lines and runs the tests. Thirty minutes later the programmer has the next micro-feature coded and ready for testing. Although a good portion of the song is made up of guitar solos, they aren't typically long solos. Lee plays a solo and soon hands the song back to his bandmates, repeating for four separate solos through the song.

Two: Scrum teams should minimize work in progress. While "I'm Going Home" is a long song (clocking in at over eleven minutes), there are frequent "deliveries" of interpolated songs throughout the performance. Listen for "Blue Suede Shoes," "Whole Lotta Shaking" and others, some played for just a few seconds.

OK, I'm probably nuts, and I certainly didn't make all these points in class. But Alvin Lee would have made one great Scrum teammate. Let me know what you think in the comments below.

[Take the Assessment](#)

Posted: June 24, 2014

Tagged: product owner, teams, teamwork, testing

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the *Better User Stories* video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

What Is Cross-Functional Collaboration in Agile?

Be a Great Product Owner: Six Things Teams and Scrum

Don't Equate Story Points to Hours

Relationship between Definition of Done and Conditions of

Scrum, Remote Teams, & Success: Five Ways to Have All

Agile Spikes Deliver Knowledge So Teams Can Deliver

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile **Agile & Scrum Training** **More...**

New to Agile and Scrum? Online Certifications About Us

Scrum Video Courses Contact Us

User Stories In-Person Certifications Our Students

Planning Poker On-Site Courses Blog

Agile Software Development Training Schedule Books by Mike Cohn

Development Compare Courses Agile Tools

Agile Project Management PDUs and SEUs

Agile Presentations

Mountain Goat on YouTube

Agile Mentors Podcast

Elements of Agile

Assessment

Presentation by:

Mike Cohn

Tagged:

product owner, teams, sprinting, backlog, sprint planning, meetings, management, project management, scrum master, scrum, agile software development, presentations, distributed teams, video recorded, keynote

You may have heard Scrum is one of the leading agile software development processes. With more than 650,000 [Certified ScrumMasters](#) worldwide, it's a proven, scalable process for managing software projects. Since its origin in Japanese new product development in the '80s, Scrum has become recognized as one of the best project management frameworks for handling rapidly changing or evolving projects, especially those with technology or requirements uncertainty.

But what is the Scrum methodology, and how does it work? This [introduction to Scrum PPT](#) will explore just that. Whether you're a manager, programmer, tester, product owner, or just want to improve product delivery, check out these Scrum presentations by Certified Scrum Trainer and author Mike Cohn of Mountain Goat Software.

In this Scrum presentation, you'll learn about product and sprint backlogs, sprint planning and sprint review meetings, and how to conduct a sprint retrospective. You'll take away key insight into measuring and monitoring progress, and scaling Scrum to work with large and distributed teams. Plus, you'll learn the roles and responsibilities of the ScrumMaster, the product owner and the Scrum team.

If you are interested in introducing Scrum methodology to your organization, [download the freely redistributable PowerPoint or Keynote version](#) of this presentation.

[View the Presentation](#)

[Download a PDF](#)

[Watch the Video](#)

Watch the video to get 1 PDU credit for:

[Learn more about PDUs](#)

The PMI Registered Education Provider logo is a registered mark of the Project Management Institute, Inc.

You may also be interested in:

[Why I Don't Emphasize Sprint Goals](#)

Sprint goals are considered a mandatory part of Scrum. Here's why I disagree.

Mar 21, 2023

[Read](#)

[What Is Cross-Functional Collaboration in Agile?](#)

Perhaps the most prevalent and persistent myth in agile is that a cross-functional team is one on ...

Feb 14, 2023

• 49 Comments

[Read](#)

Be a Great Product Owner: Six Things Teams and Scrum Masters Need

Learn six ways effective product owners ensure their teams' success.

Jan 31, 2023

• 16 Comments

[Read](#)

Six Attributes of a Great ScrumMaster

Scrum Masters exhibit a shared set of traits. Discover 6 in-demand qualities, plus a bonus ...

Jan 17, 2023

• 32 Comments

[Read](#)

What Happens When During a Sprint

Succeeding with Scrum is easier when you know when and why to conduct each of the Scrum events ...

Jan 03, 2023

34 Comments

[Read](#)

What Are Agile Story Points?

Story points are perhaps the most misunderstood topic in agile. Story points are not based on just ...

Dec 06, 2022

122 Comments

[Read](#)

New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile Assessment		

Schedule vs. Cost: The Tradeoff in Agile

by
Mike **Tagged:**
Cohn product owner, sprinting, agile projects, prioritizing, user
71 experience
[Comments](#)

To a large extent, agile is about making tradeoffs. Product owners learn they can trade scope for schedule: get more later or less sooner. Agile projects need to strike a balance between [no upfront thinking and too much upfront thinking](#), a subject I've written about before.

I want to write now about a tradeoff that isn't talked about a lot in agile circles. And this is the tradeoff between schedule and cost. We've all heard the old story that nine women can't make a baby in one month. But on software projects, it is possible (to some extent) to deliver a project faster by adding more people.

For example, suppose a project would take one person

one year to do. Two people might be able to do it in six-and-a-half months. That's one more total person-month to account for the overhead of communicating, for misunderstandings between the two, and so on. Adding a third, fourth or fifth person to the project will likely bring the calendar date in, but probably at the expense of more total person-months on the project.

Adding person-months to a project will presumably make the project more expensive to deliver. There is, then, a tradeoff to be made between schedule and cost. And most of the time, schedule wins. Companies always want things at the lowest cost they can—*but not at the expense of schedule*.

In an influential 1988 article in Harvard Business Review, George Stalk, Jr., declared that [time was the next source of competitive advantage](#). Shortening development cycles resulting in releasing new products faster is a competitive advantage—often a more important one than developing a new product at the lowest price.

This distinction is glossed over in many agile discussions. Just because a shorter schedule is more important most of the time, does not mean it is more important all of the time. This difference can lead to important but subtle differences in how an agile process may be applied.

For example, standard agile advice is that designers should work as closely as possible with programmers, testers and others on the team. User interface designs are ideally done in the same sprint in which the rest of the development work will occur. Occasionally, exceptions are made for particularly complex interfaces for which a designer may be allowed to work perhaps a sprint ahead of the others.

This is optimizing for schedule. Having the user interface designer work with the team will shorten the schedule. But this could come with the expense of some rework due to design inconsistencies discovered over a number of sprints. It could also come with additional costs due to the programmers sitting idle while waiting for the newest design.

In an agile process optimized for cost, however, the designer would not work as little ahead as possible. The designer would instead work as far ahead as possible while avoiding the opposite problems of designing hard-to-implement ideas and designing things that are no longer needed.

The designer in the latter case of our example should not strive to create a pixel-perfect design of every screen that will be part of the system. However, if cost is a more important consideration than schedule, it may very well be best for the designer to work a couple of sprints ahead.

I'd love to know what you think. What's more important on your projects? Schedule or cost? (No fair saying both. You cannot optimize for two things.) And, to lower cost, do your designers sometimes work further ahead than an agile process might otherwise say they should?

Get Your Free Scrum Cheat Sheet

Sign up below to receive this FREE reference.

First Name

Email Address

[Privacy Policy](#)

Posted: June 3, 2014

Tagged: product owner, sprinting, agile projects, prioritizing, user experience

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

Be a Great Product Owner: Six Things Teams and Scrum Masters Need

Learn six ways effective product owners ensure their teams' success.

Jan 31, 2023 • 16 Comments [Read](#)

What Happens When During a Sprint

Succeeding with Scrum is easier when you know when and why to conduct each of the Scrum events ...

Jan 03, 2023 • 34 Comments [Read](#)

How Implementation Intentions Help My Sprints

Planning an exact time to do something within a sprint helps make sure you get the important stuff ...

Mar 22, 2022 [Read](#)

The Product Owner's Second Team

Successful product owners will see their stakeholders as a team, not merely a set of individuals.

Nov 02, 2021 [Read](#)

Should You Re-Estimate Unfinished Stories?

We avoid having unfinished work, but it sometimes happens. Here's what to do.

Jun 15, 2021 [Read](#)

Top 5 changes in the 2020 version of the Scrum Guide

Recently the 2020 version of the Scrum Guide was released. What changes were made that you need to ...

Mar 16, 2021 [Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile

New to Agile and Scrum?

Scrum

User Stories

Planning Poker

Agile Software Development

Agile Project Management

Agile Presentations

Mountain Goat on YouTube

Agile Mentors Podcast

Elements of Agile

Assessment

Agile & Scrum Training

Online Certifications

Video Courses

In-Person Certifications

On-Site Courses

Training Schedule

Compare Courses

PDUs and SEUs

More...

About Us

Contact Us

Our Students

Blog

Books by Mike Cohn

Agile Tools

by

Mike Tagged:

Cohn user stories, product owner, sprinting, programming, customers,

41 scrum master, prioritizing

Comments

Participants in my Certified ScrumMaster courses are often surprised when I recommend that programmers participate in story-writing workshops. After all, a story-writing workshop is a meeting targeted at determining the functionality to be built next. I like to do story-writing workshops about quarterly to focus on bigger initiatives. But some projects will do story writing every sprint.

Regardless of how often these meetings occur, I think it's vital that programmers, testers, database engineers, and so on participate. Collectively I refer to individuals in any of these roles as *developers*, and I think there are four critical reasons to include developers in story-writing meetings.

Reason 1: Increased Engagement

The first reason I like to include developers is that doing so increases the engagement of developers. You want your developers excited about, bought into, and engaged with the product being built. A team with developers who come to work with no passion or interest in what they're developing will not perform at its highest level.

Involving developers in story writing allows them an opportunity to contribute to discussions about what is being built. It's unlikely their ideas will be as good as those of the product owner or most others with a deeper understanding of the

business and its customers, but still important enough to be heard.

Reason 2: Increased Learning

How did the business people get that deeper understanding of the business and its customers? By *spending time* thinking about the product, the business or its customers.

Bringing developers into story writing workshops gives them that opportunity. And this is our second reason for including developers: The more they are involved, the more they'll learn. Over time, you may just find that some of the best ideas *are* coming from the developers.

Reason 3: It Doesn't Cost as Much Time as You Think

A common objection to involving developers is that they don't have the time to spend in yet another meeting. Fortunately, some of the time spent in the actual meeting is saved in coming sprints because the developers will have fewer questions.

When the developers are there when ideas are generated, they will have fewer questions about why customers want something, if there are other ways to satisfy that need, and so on.

There is a time cost to having developers in these meetings, but it's not as great as you might think. And the other advantages outlined here more than make up for the investment.

Reason 4: It Leads to Increased Creativity

A final reason for including developers in story writing is the increased creativity it will lead to. With more people thinking about ways to make your product the best it can be, the product owner will have more ideas to choose from when prioritizing.

I find this especially true because developers often have such different ways of thinking about a problem than the business people do. Neither one is superior. It's the diversity of backgrounds, skills, and ways of thinking about a problem that creates the best solution for a product's users.

Have I Missed Some?

I'm sure I've missed some. What other reasons do you have that support having programmers (and other technical people) participate in story writing meetings?

[Get my guide now!](#)

Posted: May 6, 2014

Tagged: user stories, product owner, sprinting, programming, customers, scrum master, prioritizing

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

User Stories: How to Create Story Maps

Story maps help to create a shared understanding of the product, visualize user needs, and elicit ...



Mar 14, 2023

[Read](#)

How to Run a Successful User Story Writing Workshop

What you need to know to conduct a story-writing workshop with your team.

Feb 28, 2023 • 3 Comments [Read](#)

Be a Great Product Owner: Six Things Teams and Scrum Masters Need

Learn six ways effective product owners ensure their teams' success.

Jan 31, 2023 • 16 Comments [Read](#)

Six Attributes of a Great ScrumMaster

Scrum Masters exhibit a shared set of traits. Discover 6 in-demand qualities, plus a bonus ...

Jan 17, 2023 • 32 Comments [Read](#)

What Happens When During a Sprint

Succeeding with Scrum is easier when you know when and why to conduct each of the Scrum events ...

Jan 03, 2023 • 34 Comments [Read](#)

Epics, Features and User Stories

I've been getting more and more emails lately from people confused about the difference between ...

Nov 08, 2022 • 93 Comments [Re](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile **Agile & Scrum Training** **More...**

New to Agile and Scrum? Online Certifications About Us

Scrum Video Courses Contact Us

User Stories In-Person Certifications Our Students

Planning Poker On-Site Courses Blog

Agile Software Development Training Schedule Books by Mike Cohn

Development Compare Courses Agile Tools

Agile Project Management PDUs and SEUs

Agile Presentations

Mountain Goat on YouTube

Agile Mentors Podcast

Elements of Agile

Assessment

Making the Decision to Abnormally Terminate a Sprint

by
Mike Cohn
Tagged:
product owner, teams, sprint planning, scrum master
27 Comments

It's always good to have a Plan B. And all Scrum teams do--it's called an abnormal termination.

An abnormal termination is essentially blowing up the sprint and starting a new sprint instead. An abnormal termination is most frequently the result of a dramatic shift in business priorities--something previously considered important is no longer important, or something even more important is discovered.

But an abnormal termination can often be called if the team gets partway into a sprint and discovers that the work is going to take much longer than they'd anticipated in sprint planning.

Sometimes a few days of experience with a new technology or in an old part of the code reveals that implementing something will take a lot more effort than previously thought--so much more, in fact, that the product owner may not want the functionality at its new cost.

A question I've skirted around in writing the preceding is: Who decides if an abnormal termination should be invoked?

To me the answer is very clear, but there is a lot of debate about this among Scrum trainers. My opinion is that the product owner makes the call. However, many people say it should be the ScrumMaster. Let's look very practically at why it cannot be the ScrumMaster who makes the decision.

Suppose a ScrumMaster and product owner are at odds over abnormally terminating, with the ScrumMaster in favor of it. If we decide that ScrumMasters are the ones empowered to make the decision, this sprint will be abnormally terminated. The first thing a team does after any abnormal termination is plan a new sprint.

The planning meeting for the new sprint will begin--as all sprint planning meetings do--by asking the product owner what should be worked on. And the product owner will reply, "Exactly what you were working on 10 minutes ago, before the ScrumMaster called for an abnormal termination."

And there's the problem: A Scrum team works on what the product owner wants. If a Scrum rule gives the ScrumMaster the right to abnormally terminate, a product owner who disagrees with that decision can nullify the abnormal termination by having the team work in the "new" sprint on exactly what they were working on in the abnormally terminated sprint.

So giving authority over the decision to abnormally terminate to ScrumMasters doesn't work. I completely understand the appeal. It makes absolute sense until thinking through the implications, and then we realize it can't work. The decision to abnormally terminate must be given to product owners.

In wrapping up, though, I want to point out that authority over this decision is a bit theoretical. In practice, I've never seen a situation where a product owner, ScrumMaster, and team disagreed over abnormally terminating. Sure, it might take some argument and discussion to create consensus, but I've never seen a situation where the product owner, ScrumMaster, and team cannot come to agreement and need a dispute resolved by an authoritarian decision-maker.

Download Scrum Master Guide

Get a free copy of Situational Scrum Mastering: Leading an Agile Team

[Get my free guide now!](#)

Posted: February 25, 2014

Tagged: product owner, teams, sprint planning, scrum master

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

1 Why I Don't Emphasize Sprint Goals

Sprint goals are considered a mandatory part of Scrum. Here's why I disagree.

Mar 21, 2023

[Read](#)

What Is Cross-Functional Collaboration in Agile?

Perhaps the most prevalent and persistent myth in agile is that a cross-functional team is one on ...

Feb 14, 2023 • 49 Comments [Read](#)

Be a Great Product Owner: Six Things Teams and Scrum Masters Need

Learn six ways effective product owners ensure their teams' success.

Jan 31, 2023 • 16 Comments [Read](#)

Six Attributes of a Great ScrumMaster

Scrum Masters exhibit a shared set of traits. Discover 6 in-demand qualities, plus a bonus ...

Jan 17, 2023 • 32 Comments [Read](#)

Don't Equate Story Points to Hours

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • 39 Comments [Read](#)

From Project Manager to Scrum Master - 3 Tips for Making the Transition

What does a good project manager need to do to become a great Scrum Master?

Aug 23, 2022 [Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile **Agile & Scrum Training** **More...**

New to Agile and Scrum? Online Certifications About Us

Scrum Video Courses Contact Us

User Stories In-Person Certifications Our Students

Planning Poker On-Site Courses Blog

Agile Software Development Training Schedule Books by Mike Cohn

Development Compare Courses Agile Tools

Agile Project Management PDUs and SEUs

Agile Presentations

Mountain Goat on YouTube

Agile Mentors Podcast

Elements of Agile

Assessment

[Blog](#) [Six Times Two Plus One Equals A Good Project Cadence](#)

Six Times Two Plus One Equals a Good Project Cadence

by
Mike Cohn **Tagged:**
product owner, teams, sprinting, meetings, daily scrum
13 Comments

In last month's newsletter I wrote about the idea that everything happens within a sprint. There is no "outside a sprint" during which team members might do things like design, bug fixing, or anything else. In this newsletter I want to share one possible exception to that.

Something I've been doing for years is called 6x2+1. This stands for six two-week sprints followed by a one-week sprint. The idea for this came about many years ago when I was a VP of Software Development and in a meeting with the company's CEO and VP of Marketing.

We were discussing the coming quarter and what functionality my teams could deliver. The VP of Marketing asked me if we would have six or seven sprints in the coming quarter. He knew with two-week sprints we always finished six or seven sprints depending on when the quarter started.

To answer his question I started counting the sprints by thinking about the ending Friday of each: October 8, October 22, November 5, November 19, I counted. Then I had to figure out if November had 30 or 31 days. As I named each month on my knuckles, I decided from that point on each quarter would have exactly six sprints.

But six two-week sprints left me an extra week in the quarter. What to do with it? I decided I'd let the teams have that week to do whatever they want. Teams had been clamoring for more time to work on refactoring or any technically oriented thing they wanted but couldn't quite convince their product owners of. This would allow them one week each quarter in which to do whatever they wanted.

Selling the idea to product owners wasn't hard. I told them the team would still be adding value to their products during those weeks, it was just the teams would decide what was most valuable. To really sell the idea to product owners, I let them know one week per quarter the teams wouldn't need them as much, freeing them up for the bigger picture thinking and research the product owners had said they needed.

This was brilliant. I told the teams I was doing this for them. I told the product owners I was doing it for them. I was really doing it at first to get myself out of having to do date math on the fly.

There was one further benefit: This also gave each project one week per quarter that wasn't typically part of the plan when considering release dates. Whenever a project got into trouble and couldn't meet a planned date, we could use the thirteenth week as part of any normal sprint. We'd then give the team the seventeenth week, or any other later week.

Teams didn't care if they got the thirteenth or seventeenth week. They did care that they got a week sometime about once a quarter or so and very much looked forward to "their week."

Tying this back to last month's newsletter: Some of the teams with which I've done this 6x2+1 approach will treat the thirteenth week as outside of any sprint. Most teams will continue running Scrum for that one week. They'll do very simple planning and review meetings. Most will even conduct daily scrums just to hear what each other is working on.

But, some teams will do without Scrum altogether during that week. I'm OK with that although I'll admit my

preference for doing a one-week sprint with extremely lightweight (short) meetings. So, this is the one time I'd consider some work to be "outside" a sprint.

[Watch Now](#)

Posted: February 20, 2014

Tagged: product owner, teams, sprinting, meetings, daily scrum

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

What Is Cross-Functional Collaboration in Agile?

Perhaps the most prevalent and persistent myth in agile is that a cross-functional team is one on ...

Feb 14, 2023 • 49 Comments [Read](#)

Be a Great Product Owner: Six Things Teams and Scrum Masters Need

Learn six ways effective product owners ensure their teams' success.

Jan 31, 2023 • 16 Comments [Read](#)

What Happens When During a Sprint

Succeeding with Scrum is easier when you know when and why to conduct each of the Scrum events ...

Jan 03, 2023 • 34 Comments [Read](#)

Don't Equate Story Points to Hours

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • 39 Comments [Read](#)

Agile Mentors Podcast: Recap of Opening Series on Scrum

Recapping our agile podcast's initial launch series of topics

Sep 27, 2022 [Read](#)

Relationship between Definition of Done and Conditions of Satisfaction

I'd like to clarify the relationship between two important concepts: a team's Definition of Done ...

Jun 28, 2022 • 38 Comments [Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile **Agile & Scrum Training** **More...**

New to Agile and Scrum? Online Certifications About Us

Scrum Video Courses Contact Us

User Stories In-Person Certifications Our Students

Planning Poker On-Site Courses Blog

Agile Software Development Training Schedule Books by Mike Cohn

Development Compare Courses Agile Tools

Agile Project Management PDUs and SEUs

Agile Presentations

Mountain Goat on YouTube

Agile Mentors Podcast

Elements of Agile

Assessment

One of a Kind Beats Three of a Kind

by
Mike Cohn 9
Tagged:
product owner, teams, audio, prioritizing
[Comments](#)

This article has an audio version: [Download Audio Version](#)

I worked with a team last week that had three business people who each established the team's priorities. This caused a number of problems for the team because they had no clear guidance on who to listen to. The three business people each reported to the same vice president who had given them the vague guidance to not overload the team with their own priorities and to work out any conflicts among themselves.

This worked well enough--except when it didn't.

When there were conflicts the team was left in the unenviable position of having to resolve prioritization conflicts among the three business people. This was impossible. Resolving business conflicts like this required a deep business knowledge the developers didn't possess. Apparently the three business people didn't possess an adequate shared knowledge or they could have gotten beyond their differences of opinion and presented a shared set of priorities to the team.

In situations like this, I find a common approach is to push the problem down to the team. Sometimes that's right--Scrum teams are meant to be empowered and self-organizing, after all. But in cases of *conflict* I find it more appropriate to push the problems *up* the organization. In this case, rather than pushing the problems down to the team, the problem should have been pushed up the organization to the vice president to whom the three business people reported.

Because this was a Scrum project on which I was consulting, the business people were each called "product owner." And this made things very hard for the team. Team members knew they were to work on things prioritized by the product owner. But they didn't know what to do when they had three product owners who occasionally differed on priorities.

Conceptually, there is a very simple solution: Each Scrum team should have one person they recognize as their product owner. Practically, though, this can be hard to put in practice because companies often want a team to listen to multiple "product owners." To solve the problem, the organization needs to make the hard decision of classifying one of the individuals as the product owner. This makes the others (the former product owners) stakeholders.

The newly singled out product owner does not get enforce his or her will across the product. There are, of course, other stakeholders who must be satisfied. However, satisfying those stakeholders by prioritizing the proper mix of features will no longer be the responsibility of the team. The problem is pushed *up* the organization to the one rightful product owner.

[Take the Assessment](#)

Posted: August 20, 2013

Tagged: product owner, teams, audio, prioritizing

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of User Stories Applied for Agile Software Development, Agile Estimating and Planning, and Succeeding with Agile as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

What Is Cross-Functional Collaboration in Agile?

Perhaps the most prevalent and persistent myth in agile is that a cross-functional team is one on ...

Feb 14, 2023 • 49 Comments [Read](#)

Be a Great Product Owner: Six Things Teams and Scrum Masters Need

Learn six ways effective product owners ensure their teams' success.

Jan 31, 2023 • 16 Comments [Read](#)

Don't Equate Story Points to Hours

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • 39 Comments [Read](#)

Relationship between Definition of Done and Conditions of Satisfaction

I'd like to clarify the relationship between two important concepts: a team's Definition of Done ...

Jun 28, 2022 • 38 Comments [Read](#)

Agile Spikes Deliver Knowledge So Teams Can Deliver Products

On agile projects, a spike is a time-boxed research activity that helps teams make better decisions ...

May 11, 2022 • 48 Comments [Read](#)

What Is a High-Performing Agile Team?

How to build and sustain a Scrum team that exceeds the sum of its parts.

Apr 19, 2022 [Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile **Agile & Scrum Training** **More...**

New to Agile and Scrum? Online Certifications About Us

Scrum Video Courses Contact Us

User Stories In-Person Certifications Our Students

Planning Poker On-Site Courses Blog

Agile Software Development Training Schedule Books by Mike Cohn

Development Compare Courses Agile Tools

Agile Project Management PDUs and SEUs

Agile Presentations

Mountain Goat on YouTube

Agile Mentors Podcast

Elements of Agile

Assessment

Providing Feedback to Team Members

by

Mike Tagged:

Cohn product owner, teams, sprinting, meetings, continuous

15 improvement, audio, scrum master, communication

Comments

This article has an audio version: [Download Audio Version](#)

Even if you've taken the often stated agile stance of getting rid of periodic performance reviews in your organization, it can still be helpful to provide people with some commentary on how their work is perceived by others in the organization.

I like to do this by focusing on what an individual can start doing, stop doing and continue doing.

I email various people who can comment on an individual's work. This might include other team members, their ScrumMaster and product owner, stakeholders, or even people outside the company the person may have worked with on a regular basis.

I ask each to tell me what they think the person should start doing, stop doing and continue doing.

I don't weigh any one reply too heavily and instead I look for common threads or patterns in their responses. For example, one reply I received said the person should stop being so defensive. Another reply said the same person was occasionally rude in meetings and gave an example. The example corroborated the person's defensiveness and I was able to counsel the person to become more aware of it.

There are a couple of advantages to this approach:

It's very action-oriented. The person can be given a couple of specific behaviors to change (either to start or to stop).

It includes an element of praise for what others value in the person's behavior (the continues).

It can be done quickly enough that it can be done frequently and for a larger number of people as may be necessary with flatter organizational hierarchies.

It doesn't rely solely on your own observations of a person.

I've successfully used this as a manager providing feedback to my employees and as a coach helping ScrumMasters and product owners interact with their teams. If you're in either role, you might want to consider giving it a try.

(Note: I also use these same three questions--What should we start doing? Stop doing? Continue doing?--as a simple but effective way to conduct a [sprint retrospective](#).)

[Take the Assessment](#)

Posted: July 16, 2013

Tagged: product owner, teams, sprinting, meetings, continuous improvement, audio, scrum master, communication

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of User Stories Applied for Agile Software Development, Agile Estimating and Planning, and Succeeding with Agile as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

What Is Cross-Functional Collaboration in Agile?

Perhaps the most prevalent and persistent myth in agile is that a cross-functional team is one on ...

Feb 14, 2023 • 49 Comments [Read](#)

Be a Great Product Owner: Six Things Teams and Scrum Masters Need

Learn six ways effective product owners ensure their teams' success.

Jan 31, 2023 • 16 Comments [Read](#)

Six Attributes of a Great ScrumMaster

Scrum Masters exhibit a shared set of traits. Discover 6 in-demand qualities, plus a bonus ...

Jan 17, 2023 • 32 Comments [Read](#)

What Happens When During a Sprint

Succeeding with Scrum is easier when you know when and why to conduct each of the Scrum events ...

Jan 03, 2023 • 34 Comments [Read](#)

Don't Equate Story Points to Hours

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • 39 Comments [Read](#)

Agile Mentors Podcast: Recap of Opening Series on Scrum

Recapping our agile podcast's initial launch series of topics

Sep 27, 2022 [Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile **Agile & Scrum Training** **More...**

New to Agile and Scrum? Online Certifications About Us

Scrum Video Courses Contact Us

User Stories In-Person Certifications Our Students

Planning Poker On-Site Courses Blog

Agile Software Development Training Schedule Books by Mike Cohn

Development Compare Courses Agile Tools

Agile Project Management PDUs and SEUs

Agile Presentations

Mountain Goat on YouTube

Agile Mentors Podcast

Elements of Agile

Assessment

Selecting the Right User Role

by
Mike Cohn 19
Tagged: user stories, product owner
[Comments](#)

When writing user stories, one of the first considerations is who to write the user story for. Despite the name-user stories—it is sometimes beneficial to write stories for someone other than the user. No, I don't necessarily mean we want a bunch of “programmer stories.” Those are generally best avoided. Rather, I'm referring to writing what we may want to call “customer stories.”

As an example, I am currently working as the product owner on a website that will include display ads on some of the pages. I could write stories such as:

As a site visitor, I want to see ads so that when I click on them the site owner makes money and is able

to continue funding the site.

This is a valid story but it's pretty far from something a user really wants. It sounds false to me. Even though it's not a true *user* story, I would prefer to write:

As the company owner, I want ads on the site so that I make money when users click on them.

That certainly has a greater ring of truth to it! If you don't like *company owner*, I think it could just as well be the *company, product owner* or anything like that.

A similar question I was recently posed had to do with a company selling prepaid gift cards. Should user stories here be written from the perspective of the card holder redeeming the gift or from the perspective of the company that accepts the card.

As a merchant, we can accept gift cards for all transactions

As a cardholder, I can redeem a gift card at the store through which it was offered. (There could be additional complexities here as cards could be to a specific mall, rather than store. But those complexities aren't important to this discussion.)

I consider these stories to be semantically equivalent. Neither one tells me any more about the cardholder's goals. So I think either works. There will, however, be some related stories that seem better one way. For example, consider these:

As a cardholder, I can check the balance on a gift card by asking a clerk at the merchant.

As a merchant, I can inform a cardholder of the remaining balance on a gift card.

Although either of these could work, I have a slight preference for the first one because it *cardholder* is the initiator of the action of this story. I'd like the first story even better if I could make it more general:

As a cardholder, I can check the balance on a gift card so that I know how much I have left to spend.

And then add conditions of satisfaction to the story:

Must be able to check balance online by entering the card number

Must be able to check balance in a store by having a store employee swipe the card through a POS terminal.

Balance must be updated real-time.

So, while it's important to think about the best user role to embed in the story, I find that most of the time the user is quite obvious. When it's not, spend a little time thinking about the options and choose the one that feels most natural and best conveys the desired action.

Get Free User Stories Book Chapters

Please provide your name and email and we'll send you the sample chapters and we'll send a short weekly tip from Mike on how to succeed with agile.

[Get my chapters now!](#)

Posted: January 14, 2013

Tagged: user stories, product owner

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

User Stories: How to Create Story Maps

Story maps help to create a shared understanding of the product, visualize user needs, and elicit ...

Mar 14, 2023

[Read](#)

How to Run a Successful User Story Writing Workshop

What you need to know to conduct a story-writing workshop with your team.

Feb 28, 2023 • 3 Comments [Read](#)

Be a Great Product Owner: Six Things Teams and Scrum Masters Need

Learn six ways effective product owners ensure their teams' success.

Jan 31, 2023 • 16 Comments [Read](#)

Epics, Features and User Stories

I've been getting more and more emails lately from people confused about the difference between ...

Nov 08, 2022 • 93 Comments [Read](#)

Advantages of User Stories over Requirements and Use Cases

At the surface, user stories appear to have much in common with use cases and traditional ...

Oct 05, 2022 • 41 Comments [Read](#)

Relationship between Definition of Done and Conditions of Satisfaction

I'd like to clarify the relationship between two important concepts: a team's Definition of Done ...

Jun 28, 2022 • 38 Comments [Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile Assessment		

by Leonard A. Schlesinger, Charles F. Kiefer with contributions by Paul B. Brown



Published Mar 2012

Publisher: Harvard Business Review Press

I've never been a big fan of the Shewhart or Deming cycle of Plan-Do-Check-Act. Sure, it works fine for some things. I cooked an elaborate dinner for my family last night. Plan, Do, Check and Act (or Eat) worked well for that. But for what makes up much of the rest of my life, a plan-first approach has never seemed entirely appropriate.

That Plan-Do-Check-Act is often cited by agile practitioners has always surprised me. The cycle seems incongruous with agile and its antipathy toward upfront planning, upfront design, and the like.

The book [Just Start: Take Action, Embrace Uncertainty, Create the Future](#) by Leonard Schlesinger and Charles Kiefer with Paul Brown is about how entrepreneurs (and people in general) would benefit from less upfront thinking and more upfront acting--thus, the title of the book: Just Start.

In presenting this idea, the authors introduce an alternative to the Shewhart/Deming cycle. They propose a cycle of Act-Learn-Build. The first step, Act, does not refer to randomly selected, indiscriminate activity. Rather, it refers to deliberately choosing a step and acting upon it without endless upfront deliberation. One of my favorite quotes in the book is "If you have insufficient data, make your own."

Having acted, our next step is to Learn from the action. In the agile community we're fond of believing as long we have learned something from a sprint or iteration, we are moving forward, even if we discard whatever we build during that time box. We find support in this thinking from authors Schlesinger, Kiefer and Brown.

After Acting and Learning, this alternative cycle calls for us to Build upon what we've learned and do it all over again. The book does not dwell on this cycle--it's rather straightforward, after all. The main point of the book is to extoll us to avoid excessive deliberation and "just start."

I find this applicable to a variety of aspects of software and product development. We can "just start" in building the product without a disk full of spreadsheets. We can "just start" in developing without a whiteboard full of UML. The authors spend a good amount of time on risk management. They focus on advising us to determine and never exceed our "acceptable loss." This can apply to starting a business ("I can't afford to lose more than \$100,000"). It could just as readily apply to technical decisions ("I can't afford to lose more than a week on this approach but if it works out, we'll have a month of development time so I'm going to try it.")

A point I found interesting was a comment that of the serial entrepreneurs studied, none had tried to gather specific information about potential returns or the ideal size of an investment in a new venture. That is, they just started but did so with an awareness that they would not spend more than their acceptable loss. I was recently contemplating two new businesses. I chose to start one but not the other. I had created a rather extensive spreadsheet modeling the business I chose not to start. I didn't create such a model for the one I chose to start. Rather, and in keeping with the advice in this book, I started the business but did so with a maximum acceptable loss investment in mind.

One thing I found a bit too cutesy with the book was that the authors made up the word "Creaction" by combining creation with action and they use this throughout the book. I can't picture myself using this word but I have found myself thinking it more, so maybe my opposition to the new word is softening over time.

A nice bonus to the book is that the framework is presented with examples beyond the business world. Suggestions of how to apply the Just Start framework to one's personal life are provided.

I can highly recommend [Just Start: Take Action, Embrace Uncertainty, Create the Future](#) to anyone involved in agile projects, especially those on the product management or general management side of things.

Tagged: product owner, sprinting, management, agile projects, stakeholders, general business, scrum
product owner, book reviews

You may also be interested in:

[Be a Great Product Owner: Six Things Teams and Scrum Masters Need](#)

Learn six ways effective product owners ensure their teams' success.

Jan 31, 2023 • [16 Comments](#)

[Read](#)

[What Happens When During a Sprint](#)

Succeeding with Scrum is easier when you know when and why to conduct each of the Scrum events ...

Jan 03, 2023 • [34 Comments](#)

[Read](#)

[7 Ways to Get and Improve Fast Feedback](#)

Here's how to get the rapid feedback you need to ensure you build the right product.

Jul 26, 2022

[Read](#)

What Is a High-Performing Agile Team?

How to build and sustain a Scrum team that exceeds the sum of its parts.

Apr 19, 2022

[Read](#)

How Implementation Intentions Help My Sprints

Planning an exact time to do something within a sprint helps make sure you get the important stuff ...

Mar 22, 2022

[Read](#)

Top 7 Ways to Get Stakeholders to Attend Sprint Reviews

Poorly attended sprint reviews cause real problems. Fortunately there are easy fixes.

Mar 08, 2022

[Read](#)

Learn About Agile

New to Agile and Scrum?

Agile & Scrum Training

Online Certifications

More...

About Us

Scrum

Video Courses

Contact Us

User Stories

In-Person Certifications

Our Students

Planning Poker

On-Site Courses

Blog

Agile Software Development

Training Schedule

Books by Mike Cohn

Agile Project Management

Compare Courses

Agile Tools

Agile Presentations

PDUs and SEUs

Mountain Goat on YouTube

Agile Mentors Podcast

Elements of Agile

Assessment

by Kenneth S. Rubin



Published Aug 2012

Publisher: Addison-Wesley Professional

The book is a comprehensive overview of Scrum. It goes from the principles of agile through the mechanics of sprints to the roles on a Scrum team and all the way up to topics like technical debt and portfolio management with Scrum. A very helpful aspect of the book is the detailed "visual language," Kenny created while writing the book. He created icons for every possible aspect of Scrum and these are used to make up dozens and dozens of figures to illustrate all the work and knowledge flows of a Scrum project. His diagrams definitely go well beyond the typical double-loop depiction of Scrum.

The book is well-written and draws on Kenny's experience introducing Scrum to over 200 organizations. I've worked with Kenny in various capacities over the past twelve years and I can highly recommend his book. In fact, I'd considered writing a book just like this until he'd told me he was already writing it. So I did the next best thing and asked him to include the book in the series I edit for Addison-Wesley. Because it's in that series I wrote a foreword to the book, which I'll reproduce here to give you a bit more insight into what I like about this book:

From the Foreword

I had lunch today at a Burger King. A sign on the wall proclaimed the restaurant the “Home of the Whopper” and then proceeded to tell me there were over a million different ways to order a Whopper. If various combinations of extra or no pickles, tomatoes, lettuce, cheese, and so on can lead to over a million ways to make a hamburger, there must be billions of possible ways to implement Scrum. And while there is no single right way, there are better and worse ways to implement Scrum.

In *Essential Scrum*, Kenny Rubin helps readers find the better ways. His isn’t a prescriptive book—he doesn’t say, “You must do this.” Instead, he teaches the essential principles underlying success with Scrum and then gives us choices in how we live up to those principles. For example, there is no one right way for all teams to plan a sprint. What works in one company or project will fail in another. And so Kenny gives us choices. He describes an overall structure for why Scrum teams plan sprints and what must result from sprint planning, and he gives us a couple of alternative approaches that will work. But ultimately the decision belongs to each team. Fortunately for those teams, they now have this book to help them.

An unexpected benefit of *Essential Scrum* is the visual language Kenny introduces for communicating about Scrum. I found these images very helpful in following along with the text, and I suspect they will become commonplace in future discussions of Scrum.

The world has needed this book for a long time. Scrum started as a small concept. The first book to talk about it—*Wicked Problems, Righteous Solutions* in 1990 by DeGrace and Stahl—did so in six pages. But in the more than 20 years since that book appeared, Scrum has expanded. New roles, meetings, and artifacts have been introduced and refined. With each new piece that was added, we were at risk of losing the heart of Scrum, the part of it that is about a team planning how to do something, doing some small part of it, and then reflecting on what the team members did and how well they did it together.

With *Essential Scrum*, Kenny brings us back to the heart of Scrum. And from there teams can begin to make the decisions necessary to implement Scrum, making it their own. This book serves as an indispensable guide, helping teams choose among the billions of possible ways of implementing Scrum and finding one that leads to success.

Tagged: product owner, teams, sprinting, sprint planning, meetings, management, scrum master, book reviews

You may also be interested in:

Why I Don't Emphasize Sprint Goals	What Is Cross-Functional Collaboration in Agile?	Be a Great Product Owner: Six Things Teams and Scrum Masters Need	Six Attributes of a Great ScrumMaster	What Happens When During a Sprint	Don't Equate Story Points to Hours
Sprint goals are considered a mandatory part of Scrum. Here's why I disagree.	Perhaps the most prevalent and persistent myth in agile is that a cross-functional team is one on ...	Learn six ways effective product owners ensure their teams' success.	Scrum Masters exhibit a shared set of traits. Discover 6 in-demand qualities, plus a bonus ...	Succeeding with Scrum is easier when you know when and why to conduct each of the Scrum events ...	I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Mar 21, 2023

[Read](#)Feb 14, 2023 • 49 Comments [Read](#)Jan 31, 2023 • 16 Comments [Read](#)Jan 17, 2023 • 32 Comments [Read](#)Jan 03, 2023 • 34 Comments [Read](#)Nov 22, 2022 • 39 Comments [Read](#)**Learn About Agile**

New to Agile and Scrum?

Scrum

User Stories

Planning Poker

Agile Software Development

Agile Project Management

Agile Presentations

Mountain Goat on YouTube

Agile Mentors Podcast

Elements of Agile Assessment

Agile & Scrum Training

Online Certifications

Video Courses

In-Person Certifications

On-Site Courses

Training Schedule

Compare Courses

PDUs and SEUs

More...

About Us

Contact Us

Our Students

Blog

Books by Mike Cohn

Agile Tools

Presentation by:

Mike Cohn

Tagged:user stories, product owner, management, agile processes,
presentations

The vast majority of what has been written about agile processes is intended for programmers and project managers. As an agile product manager, product owner, or member of the customer team, you need help identifying user needs and determining how best to meet those needs. Yet very little information is available to help you identify the right thing to build or juggle competing priorities.

Leading agile speaker and author Mike Cohn has tailored a presentation specifically for you. Find out how agile product managers and Scrum product owners write user stories as requirements and express a project's conditions of satisfaction as tests. Gain insight into how to make tradeoff decisions between features, time, and resources. Take away a new understanding of how to prioritize features into a release based on something more than the nebulous "business value" and influence the quality of work by the developers.

[View the Presentation](#)[Download a PDF](#)**You may also be interested in:**

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

Presentation by: [Mike Cohn](#) Tagged: product owner, teams, agile processes, prioritizing, presentations

Building the right product at the right time is just as important as building a quality product. Project economics can make or break a project. Knowing the benefits (and the costs) of a project is essential to deciding whether it's a sound investment. What you need to know is how to quantify benefits in the same way technical teams estimate effort and cost.

These presentations by agile author and expert Mike Cohn explore return on investment (ROI) as well as traditional discounted cash flow methods such as net present value (NPV) and internal rate of return (IRR). Learn newer approaches, too, such as economic value added (EVA) and real options valuation, which are especially applicable for teams using agile processes or an incremental development process. The math is easy; the concepts are powerful. Leave with practical knowledge about how to apply these straightforward techniques to prioritizing and selecting projects.

[View the Presentation](#)[Download a PDF](#)

You may also be interested in:

Learn About Agile	Agile & Scrum Training	More...
New to Agile and Scrum?	Online Certifications	About Us
Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		

[Blog](#)[Please Help Me List The Problems With Using Agile Or Scrum](#)

Please Help Me List the Problems with Using Agile or Scrum

by [Mike Cohn](#) Tagged:
product owner, sprinting, meetings, agile problems

I'm trying to create a list of the biggest, most common, or hardest to overcome agile problems that a team might face when adopting Scrum. I could really use your help by contributing to the list by adding a comment to this post. I'm thinking of things like:

We have five product owners. What do we do?

We drop work from every sprint. How do we get out of that habit?

We can't ever get things "potentially shippable" at the end of a sprint. How can we?

We spend forever in planning meetings. How can we spend less time doing that?
etc...

So, what agile problems have you encountered or heard of teams encountering? Thanks for your help.

Get Your Free Scrum Cheat Sheet

Sign up below to receive this FREE reference.

First Name

Email Address

[Privacy Policy](#)

Posted: January 3, 2012

Tagged: product owner, sprinting, meetings, agile problems

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding

member of the Agile Alliance and Scrum Alliance and
can be reached at hello@mountaingoatsoftware.com.
If you want to succeed with agile, you can also have
Mike email you a short tip each week.

You may also be interested in:

[Be a Great Product Owner: Six Things Teams and Scrum Masters Need](#)

Learn six ways effective product owners ensure their teams' success.

Jan 31, 2023 • 16 Comments

[Read](#)

What Happens When During a Sprint

Succeeding with Scrum is easier when you know when and why to conduct each of the Scrum events ...

Jan 03, 2023 • [34 Comments](#)

[Read](#)

8 Reasons Scrum Is Hard to Learn (but Worth It)

Transitioning to Scrum is worth it, but some aspects are challenging.

Oct 11, 2022

[Read](#)

[Agile Mentors Podcast: Recap of Opening Series on Scrum](#)

Recapping our agile podcast's initial launch series of topics

Sep 27, 2022

[Read](#)

[Seven Tips for Running a Virtual Daily Scrum](#)

How do you effectively facilitate a daily scrum with a remote team?

May 03, 2022

[Read](#)

How Implementation Intentions Help My Sprints

Planning an exact time to do something within a sprint helps make sure you get the important stuff ...

Mar 22, 2022

[Read](#)

Comments on this post are closed.

Learn About Agile

New to Agile and Scrum?

Scrum

User Stories

Planning Poker

Agile Software Development

Agile Project Management

Agile Presentations

Mountain Goat on YouTube

Agile Mentors Podcast

Elements of Agile

Assessment

Agile & Scrum Training

Online Certifications

Video Courses

In-Person Certifications

On-Site Courses

Training Schedule

Compare Courses

PDUs and SEUs

More...

About Us

Contact Us

Our Students

Blog

Books by Mike Cohn

Agile Tools

Protecting the Team Cuts Both Ways

by
Mike Cohn 32 Comments
Tagged: product owner, teams, continuous improvement, scrum master

It is a generally accepted [Scrum](#) dictum that one of the Scrum Master's duties is to protect the team. The usual example is that the [Scrum Master](#) must protect the team from an overly aggressive product owner. There is nothing wrong with this example and many teams do need to be protected from a [product owner](#) whose desire for more functionality can push the team into cutting quality corners.

However, a good Scrum Master also protects the [Scrum team](#) from a problem that can be even more harmful: complacency. After achieving some easy, early improvement from Scrum, some agile teams become self-satisfied. They stop seeking further improvements. It is up to the Scrum Masters of those teams to protect them from this complacency.

So, a good Scrum Master will occasionally have to stand up to an aggressive product owner and say, "Now is not the time to push this team any harder. They're working as hard as they can and if pushed more, they're likely to get sloppy."

But my advice to good ScrumMasters is that for each time they say this, they should later tell the product owner something like, "Now's the time. The team is rested. They're ready. Let's see what they can do. It's time to push them for more."

If you're a Scrum Master, be sure to protect your team from both types of problem.

[Take the Assessment](#)

Posted: July 26, 2011

Tagged: product owner, teams, continuous improvement, scrum master

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

1 What Is Cross-Functional Collaboration in Agile?

Perhaps the most prevalent and persistent myth in agile is that a cross-functional team is one on ...

Feb 14, 2023 • 49 Comments [Read](#)

Be a Great Product Owner: Six Things Teams and Scrum Masters Need

Learn six ways effective product owners ensure their teams' success.

Jan 31, 2023 • 16 Comments [Read](#)

Six Attributes of a Great ScrumMaster

Scrum Masters exhibit a shared set of traits. Discover 6 in-demand qualities, plus a bonus ...

Jan 17, 2023 • 32 Comments [Read](#)

Don't Equate Story Points to Hours

I've been quite adamant lately that story points are about time, specifically effort. But that does ...

Nov 22, 2022 • 39 Comments [Read](#)

Elements of Agile: An Agile Assessment Tool for Iterative Improvements

New to agile, or needing an agile re-boot? We've got a new no-cost assessment and actionable ...

Sep 13, 2022 [Read](#)

From Project Manager to Scrum Master -3 Tips for Making the Transition

What does a good project manager need to do to become a great Scrum Master?

Aug 23, 2022 [Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Learn About Agile **Agile & Scrum Training** **More...**

New to Agile and Scrum? Online Certifications About Us

Scrum Video Courses Contact Us

User Stories In-Person Certifications Our Students

Planning Poker On-Site Courses Blog

Agile Software Development Training Schedule Books by Mike Cohn

Development Compare Courses Agile Tools

Agile Project Management PDUs and SEUs

Agile Presentations

Mountain Goat on YouTube

Agile Mentors Podcast

Elements of Agile

Assessment

Avast Combining the ScrumMaster and Product Owner, Matey!

by
Mike Cohn 21
Tagged:
product owner, agile project management, scrum master
[Comments](#)

A common question is whether it's acceptable to combine the role of product and ScrumMaster and give both sets of responsibilities to a single person. In general, combining these roles is a very bad idea. To see why, let's look back in history and the job of the 17th-century pirate ship captain. In a recent issue of the Harvard Business Review (October 2010), Professor Hayagreeva Rao wrote about the results of asking his MBA students to design the job of a 17th century pirate ship captain. His MBA students designed a job that lumped together two areas of responsibility:

1. *star tasks*—the strategic work of deciding which ships to attack, commanding the crew during battle, negotiating with other captains, and so on
2. *guardian tasks*—the operational work of distributing their pirate booty, settling conflict, punishing crew members, and organizing care for the wounded

The problem with this job description is that it mixes star and guardian tasks. As Professor Rao points out, there are very few individuals who excel at both types of task. Star tasks require risk-taking and entrepreneurship whereas guardian tasks require conscientiousness and consistency. A pirate captain good at identifying ships to attack and at leading his crew into battle would likely be bored by the administrative minutiae of the guardian tasks. Professor Rao claims that people tend to spend most of their effort on the tasks they are good at (and presumably enjoy). My experience certainly bears this out. Pirates avoided this problem by having a captain responsible for the star tasks and a quartermaster general responsible for the guardian tasks. So what does the decidedly non-collaborative, non-agile environment of a pirate ship have to do with [agile project management](#)? Well, it turns out that the product owner is largely performing star tasks and the ScrumMaster is largely performing guardian tasks. And so, for the same reason that pirate ships had separate individuals as captain and quartermaster general, our agile software development projects should have separate ScrumMasters and product owners.

Get Your Free Scrum Cheat Sheet

Sign up below to receive this FREE reference.

First Name

Email Address

[Privacy Policy](#)

Posted: October 10, 2010

Tagged: product owner, agile project management, scrum master

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile* as well as the [Better User Stories](#) video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.

You may also be interested in:

[Be a Great Product Owner: Six Things Teams and Scrum Masters Need](#)

Learn six ways effective product owners ensure their teams' success.

Jan 31, 2023 • [16 Comments](#)

[Read](#)

[Six Attributes of a Great ScrumMaster](#)

Scrum Masters exhibit a shared set of traits. Discover 6 in-demand qualities, plus a bonus ...

Jan 17, 2023 • [32 Comments](#)

[Read](#)

[From Project Manager to Scrum Master -3 Tips for Making the Transition](#)

What does a good project manager need to do to become a great Scrum Master?

Aug 23, 2022

[Read](#)

[Seven Tips for Running a Virtual Daily Scrum](#)

How do you effectively facilitate a daily scrum with a remote team?

May 03, 2022

[Read](#)

[What Does Scrum Mean by Cross Functional Teams?](#)

What exactly does cross functional mean and why does it matter?

Apr 05, 2022

[Read](#)

How to Create Self-Sufficient Scrum Teams

Mar 01, 2022

[Read](#)

The discussion here is closed but join us in the Agile Mentors Community to further discuss this topic.

Scrum	Video Courses	Contact Us
User Stories	In-Person Certifications	Our Students
Planning Poker	On-Site Courses	Blog
Agile Software Development	Training Schedule	Books by Mike Cohn
Agile Project Management	Compare Courses	Agile Tools
Agile Presentations	PDUs and SEUs	
Mountain Goat on YouTube		
Agile Mentors Podcast		
Elements of Agile		
Assessment		