Table of Contents

Contents

[1 Vocabulary 1](#_Toc412552227)

[2 Setup Scrum 3](#_Toc412552228)

[2.1 Scrum Software 3](#_Toc412552229)

[2.1.1 Scrum Tool Selection 3](#_Toc412552230)

[2.1.2 Install the Software 3](#_Toc412552231)

[2.1.3 Configure the software 3](#_Toc412552232)

[2.2 Create a new Scrum Project 3](#_Toc412552233)

[2.2.1 Name the Project 3](#_Toc412552234)

[2.2.2 Establish a specific mission statement for the team 3](#_Toc412552235)

[2.2.3 Set a social contract 3](#_Toc412552236)

[2.2.4 Start to build the definition of Ready and Done 4](#_Toc412552237)

[2.2.5 Select a style for story points. 4](#_Toc412552238)

[2.2.6 Add a Sprint Name 4](#_Toc412552239)

[2.2.7 Configure the Task Board 4](#_Toc412552240)

[2.2.8 Update Quick Filters 5](#_Toc412552241)

[2.2.9 Configure Card layout 5](#_Toc412552242)

[2.2.10 Add Users 5](#_Toc412552243)

[3 Prototyping 6](#_Toc412552244)

[3.1 Road Mapping/Brainstorming 6](#_Toc412552245)

[3.1.1 Brainstorming techniques 6](#_Toc412552246)

[3.2 Requirement Prototyping 6](#_Toc412552247)

[3.3 Prototyping Steps 6](#_Toc412552248)

[3.3.1 Kick Off 6](#_Toc412552249)

[3.3.2 Early requirements Gathering 6](#_Toc412552250)

[3.3.3 Requirements Prototyping Cycle 6](#_Toc412552251)

[3.3.4 Design Finalization 7](#_Toc412552252)

[3.3.5 Development 7](#_Toc412552253)

[4 Getting Started With Production 8](#_Toc412552254)

[4.1 Build a backlog 8](#_Toc412552255)

[4.2 Create the first product backlog stories 8](#_Toc412552256)

[4.2.1 Define the Stories Acceptance Criteria 8](#_Toc412552257)

[4.2.2 StoryTime / Backlog Grooming 8](#_Toc412552258)

[4.2.3 Story Point estimation: 8](#_Toc412552259)

[4.2.4 Epics 8](#_Toc412552260)

[5 Install Programs 9](#_Toc412552261)

# Vocabulary

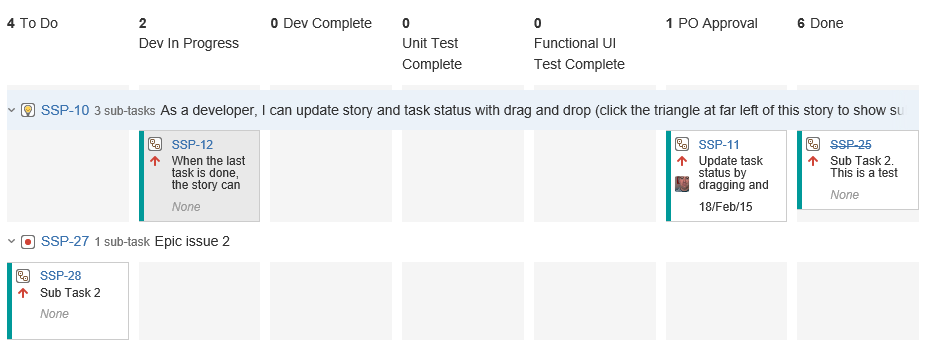
* Product Owner (PO).
  + Must have the vision for the product development. Better than a detailed roadmap.
  + Decides what is needed, not how.
  + Makes the business decisions
  + Responsible for Return On Investment (ROI)
  + Final arbiter of requirements questions.
  + Focused more on the WHAT than on the HOW.
* Development Team
  + Ideally, a learning team.
  + No hierarchy
  + Self-Organizing
    - No external job titles.
    - Leadership occurs naturally.
  + Flow from person to person
  + 4-9 People. For larger groups, still use 4-9 people in “Feature” teams.
  + Cross functional Group
  + Attempts to build a “potentially shippable product increment” every sprint
  + Collaborates
  + Includes the Scrum Master
* Scrum Master
  + No management power. Project management is spread throughout the team.
  + Acts as a facilitator
  + Protects the team from distraction
  + Removes impediments
  + Teaches people how to use Scrum
  + Promotes development best practices
  + Enforces time boxes.
* Time Box
  + Allocates a fixed time period to each planned activity.
  + Don’t treat it as a deadline.
  + A regular re-vist and re-assessment of what’s currently being done.
* Feedback Loop
  + A process has a feedback loop when the results of running the process are allowed to influence how the process itself works in the future.
    - Examples  
      Pair programming

Unit tests  
Continuous integration

Daily Scrum  
Sprints  
Code Review  
Have the customer and business experts work closely with you throughout the process.

Increase the frequency of release

* Project management
  + PO
  + Development Team
  + Scrum Master (only facilitator)
* Product Backlog
  + Stories are the items. Maybe use case scenarios.
  + Location where Product Backlog Items are stored.
  + Customer centric features provided by the PO
  + Anyone can add, but the PO has to prioritizes
  + Scrum Master makes the item visible
  + NO TASKS, NO PRODUCT FEATURES
  + Forced rank priority. Items are always ranked by priority.
  + Everything that could be done is listed.
  + Scrum master makes the PBI visible to the team.
  + Backlog Items consist of:
    - Stories
    - Use Cases
    - Essential Use Cases
* Sprint Backlog
  + What we are going to do for the current 2 weeks – 30 days.
  + Has a start and end date
  + 30 days but usually 2 weeks
  + WHAT = Product Backlog Items
  + HOW = Sprint Tasks / Kanban columns



* Epic
  + Large group of product backlog items.
* Outlier
  + Person who is really out of production.
* Sidebar
  + Ideas to discuss after meetings.
* Pair Programming
  + 2 people share one computer.
  + One person types while the other person pays attention.
  + Take turns typing and watching.
* Test Driven Development
  + Unit test faulty code
  + Write product code
  + Refactor code
* Refactor Code
  + Taking existing code and transforming it into a new form that behaves the same as before but that no longer “smells” Long routines are made smaller, duplicates are removed or replaced by one shared function.
  + Easier to maintain code because its easy to read and the intent is easy to grasp.
  + Extensibility: It’s easier to extend the capabilities of the application if it’s a recognized design pattern.
    - Techniques
      * Allow More Abstraction (polymorphism, code sharing).
      * Breaking code apart into logical pieces
      * Improve name and location of code.
* Technical Debt / Design Debt / Code Debt
  + The consequences of poor design, architecture, development with a project.
* Polymorphism
  + It allows you to avoid writing an explicit conditional when you have objects whose behavior varies depending on their types.
  + Biggest gain is when same set of conditions appear in many places in the program.
    - Example: Employee extends person. Employee can use the same functions as person and then extend it if necessary.
* Continuous Integration Server
  + Merges all code changes into a central location (repository)
  + Tests are run when a commit to the repository triggers the build.
  + Does not use a periodical schedule.
  + Need Code Repository
  + Automate the build
  + Make the build self testing
  + Everyone commits to the baseline
  + Every commit should be built.
* Self Deception
  + Team members deny or rationalize away the importance of opposing evidence or logical argument.
* Muda
  + Japanese word meaning “futility; uselessness;wasteful;etc”. Key concept in TPS.
* Spike
  + A story or task aimed at answering a question or gathering info.
  + A spike is like a spike in rock climbing.
  + No velocity
  + Enables future climbing
  + No code
  + No story in backlog
  + No story points
  + Research overhead
* Velocity
  + Completed PBI’s
  + Helps the product owner make forecasts.
* Applause
  + Do not applause during meetings
  + This could interfere with transparency.
* Exploratory Testing
  + Also known as XT or Agile testing.
  + Test design and test execution at the same time
  + are not defined in advance and carried out precisely according to plan
  + Not Ad hoc testing: normally refers to a process of improvised, impromptu bug searching
  + <https://university.utest.com/what-is-exploratory-testing/>
* Forecasted
  + The team has committed to do something.
* Invisible Gun Effect
  + The presence of someone with power and responsibility in the organization will prevent the team from stepping up to the same degree of self-management.
  + Colors subordinates interactions with each other.
* Pluses and Deltas
  + What went well and what could be improved.
* Extreme programming
  + I think modern Scrum uses these engineering practices
    - Unit test all code.
    - Avoid programming of features until needed
    - Programming in pairs
    - Clarity and simplicity in code
    - Flag management structure
  + Difference between scrum and Extereme programming
    - EP can change Sprint Backlog Items
    - PO does not prioritize the set of activities
* INVEST  
  Created by Bill Wake : <http://xp123.com/articles/invest-in-good-stories-and-smart-tasks/>   
  Helps with the characterstic’s of good stories. The acronym “INVEST” is used to remind us of what good stories are made of:
  + I – Independent
  + N – Negotiable
  + V – Valuable
  + E – Estimable
  + S – Small
  + T – Testable
* Measurable Viable Product (MVP)

A name for a product with the highest return on investment versus risk. A product that has just those core features that allow the product to be deployed and no more. It may be used by early adopters because they are more forgiving and will give feedback.

* Lean UX Design Process  
  Based on Agile and it’s a way to design applications with just enough features to see if it works. Outcome is a MVP. Best information is:  
  <http://www.smashingmagazine.com/2011/03/07/lean-ux-getting-out-of-the-deliverables-business/>   
  <http://fakecrow.com/shortening-ux-to-ui-time-a-different-approach-to-mvp-design/>
* Work In Progress (WIP) limit  
  “Stuff on your plate”  
  Limit the number of active tasks a person is working on. Avoids problems with Forgetting, leaving things unfinished, quality.

# Setup Scrum

Decide on a development framework to manage the product development. In this case, we’ll use Scrum. Others are:

* Agile
* Waterfall
* Extreme Programming (XP)

## Scrum Software

### Scrum Tool Selection

Select a tool that works for your organization. I like Hansoft and Jira. Jira’s cloud based solution is very popular. Hansoft is used in the game development industry.

Jira Agile comes with Scrum boards. <https://www.atlassian.com/software/jira>

### Install the Software

Make sure to create a solution that is accessible from anywhere. Cloud or local servers are great. All need to be accessible through the internet.

### Configure the software

After installing the software, make sure to tailor the following

## Create a new Scrum Project

### Name the Project

Meaningful names and code names are common. Project names that promote team identity centered on the product. Avoid names that would be used only in the industry and others wouldn’t know. Prefer self-explanatory names.

### Establish a specific mission statement for the team

What is the goal of the project. For example: Create a basic weather app using Visual Studio, JavaScript, …

### Set a social contract

Well use the following rules to govern how our team function and respects each other.

1. We will be united about our projects and always promote them with positivity.  
2. We will manage and update our charts in real-time to share information.  
3. Our meetings will have a clear agenda, time-box, and end with actionable goals.  
4. We will respect the time-box (start on time, end on time).  
5. We will not schedule regular meetings during a team member’s gym time.  
6. Our phones will be set to vibrate and only used for emergencies during meetings.  
7. If working from home, we will still attend confirmed meetings by Skype or phone.  
8. If we are late, we will owe the team a round of Starbucks at the next meeting (any drink, any size).  
9. We will stay focused on one topic at a time and listen to each other.  
10. We will respect each other, even if we have a difference of opinion.  
11. We will not accept a story unless it is in our sprint.  
12. We will work together to deliver our sprint (no individual failure, only team failure.

### Start to build the definition of Ready and Done

The following definitions are a start and can be updated during development. Update the definition of done as the project encounters problems.

**Definition of READY:**

• Is the story written as a narrative?  
• Does the story have acceptance criteria?  
• Does the story have assigned business value in relation to other stories?  
• Does the team have a shared understanding of what the story means?  
• Has the story been estimated by the team?  
• Is the story small enough to fit in a sprint?

**Definition of DONE:**

* + All written content edited and finalized in both languages
  + All acceptance criteria identified and passed
  + Product has been properly tested
  + Product code has been refactored
  + Potentially Shippable
  + Product accepted by Product Owner

#### Definition of Done for a feature (story or product backlog item)

* Coding is done.
* Coding comments.
* Released to a staging environment
* Testing and PO approval
* Documentation (just enough)

#### Definition of Done for a sprint (collection of features developed within a sprint)

#### Definition of Done for a release (potential shippable state)

### Select a style for story points.

There are many different types of story points used for time estimation. Story cards are best and here are a few:

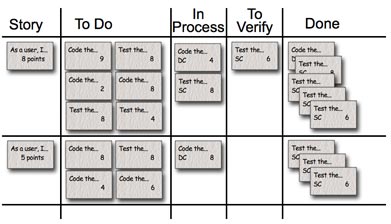
* T-Shirt Sizes
* Planning Poker
* Regular Playing Cards
* Fibonacci

### Add a Sprint Name

There are many approaches to naming. Give the sprint an incrementing number ie Sprint 1. Some sprints are names after months ie. Sprint 02 of July. Another format > Project, Release, Deliverable. Ie Ease of Use/3.1/Milestone 1.

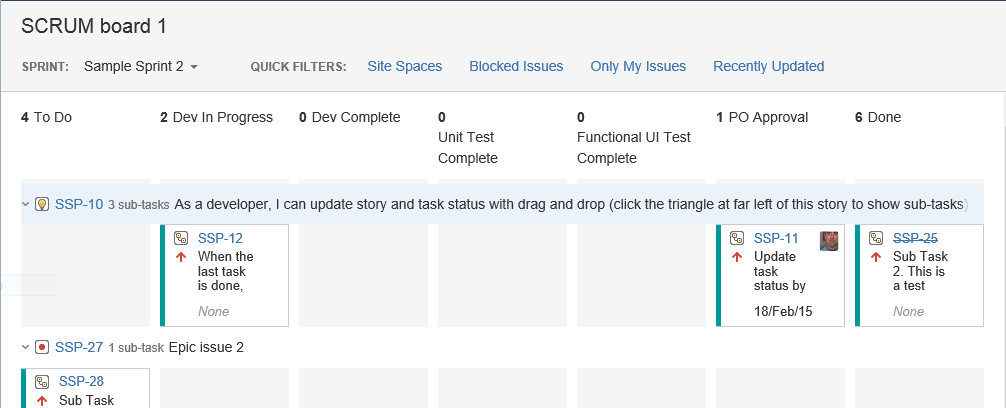
### Configure the Task Board

This is also called Scrum board. This is the location where sprint backlog items are visible. Team members update the task board continuously throughout the sprint. The task are moved around the board to show the working state. Each row represents a user story, use case, etc. Each task is represented with a card. Each cards starts in To Do.

[](http://www.mountaingoatsoftware.com/system/hidden_asset/file/29/MockedTaskBoard.jpg)

Add the following columns to the task board:

1. To Do
2. Dev In Progress
3. Dev Complete
4. Unit Test Complete
5. End-To-End Test / Functional UI Test Complete
6. PO Approval
7. Done



### Definitions of Done for a sprint

#### Development in Progress

Writing code, and coding comments

#### Unit Test Complete

#### Functional UI Test Complete

#### Approved

#### Committed

Release notes,

#### Done

### Update Quick Filters

Add the following quick filters for a starter.

* Site Spaces
* Blocked Issues

### Configure Card layout

Include the following to the card

* Due Date
* Estimate­

### Add Users

It will be necessary to add users as Administrators, Developers and Users. If necessary, add a new group.

# Prototyping

Done before the production starts.

## Road Mapping/Brainstorming

Delivers the following:

1. A vision and description of the completed system that is going to be delivered.
2. Shows the order you intend to implement the ideas. Sort deliverables by priority.
3. A minimum useful release you can show your customers. Basically, show them something that demonstrates how the software will work.

### Brainstorming techniques

* Write down everything you want to deliver.
* Brainstorming sessions are helpful.
* Incrementalize each idea you want to deliver.
* Sort delivery based on priority.

## Requirement Prototyping

Prototyping is necessary because:

* General objectives are in place.
* Detailed requirements are not in place.

## Prototyping Steps

### Kick Off

* + Discuss and agree on foundation objectives
  + Agree on requirements
  + Rate Objective and Requirements according to priority, completeness and expected complexity. Covers the entire project.

### Early requirements Gathering

* + Prototype lead will take ideas to stakeholders and extract information about the foundation objectives.
  + Begin work on high level flow prototype. Flow prototype glosses over specific details. Instead, acts as a placeholder for which PBI’s can be created.

### Requirements Prototyping Cycle

* Using the high level prototype, and talk with business stakeholders and end-users
  + Try typical user scenarios against the design.
  + Get their input and build that detail into to a new prototype.
* Multiple models/alternative approach may be necessary.
* Create sprints for prototypes
  + Timebox
  + Reviewed at the end of each sprint.
* Create/update an interactive requirements specification document.

### Design Finalization

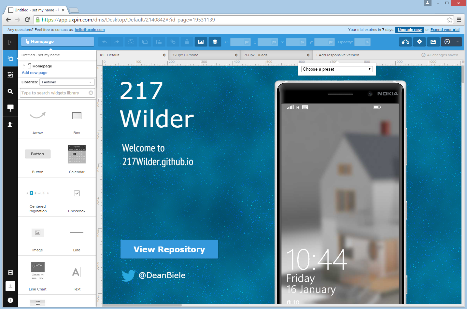
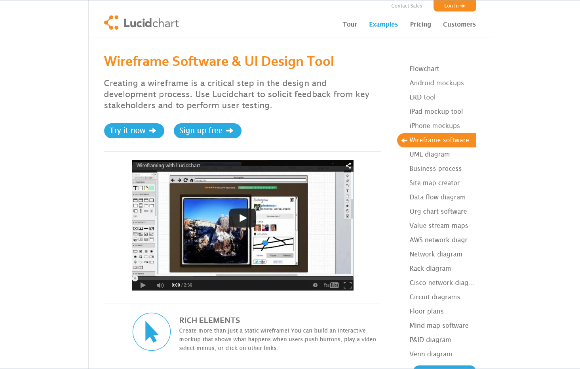
* Interactive requirements specification double as a simulation for the proposed system.
* Workshop is held
  + All key stakeholders
  + User representatives
* Signing off on the design.

### Development

* + Use the prototype as the foundation and guide for PBI’s.
  + Prototype lead may become the Project Owner.

### Software

There are many rapid prototyping application available. Here are two:

* UXPin <http://www.uxpin.com>   
  Cloud based Wireframing, Prototyping, Collaboration  
  
* LucidChart <https://www.lucidchart.com>  
  Cloud based diagram app. Includes wireframe software and UI Design Tool.  
  

# PO Making the Backlog/Backlog Grooming: Getting Started With Production

## Build a backlog

Now that you have a project and board, the PO needs to create stories.

## Who is Involved

PO

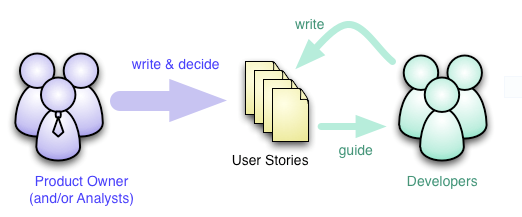
## Time

## Create the first product backlog items

Stories  
Short descriptions of the functionality that is needed in the product. Easy to read. Follow this format when writing stories.

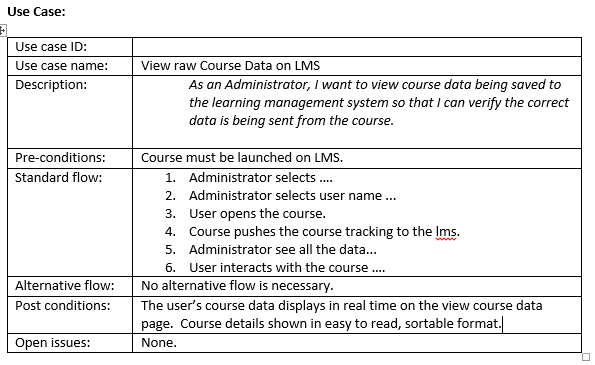
As a [role], I want [something] so that [benefit].  
As a [end user role], I want [the desire] so that [the rationale].  
As a [type of user], I want [some goal] so that [some reason].  
As a [**person in a role**] I want to [**perform some activity**] so that [**some goal is achieved**].

Stories can use INVEST principles to test the stories.  
User Stories are not decreed by the PO, but it is more of a conversation. A conversion between the customers and developers about how a development project should proceed. Conversations refine the story. The PO is the final decider and prioritizes.

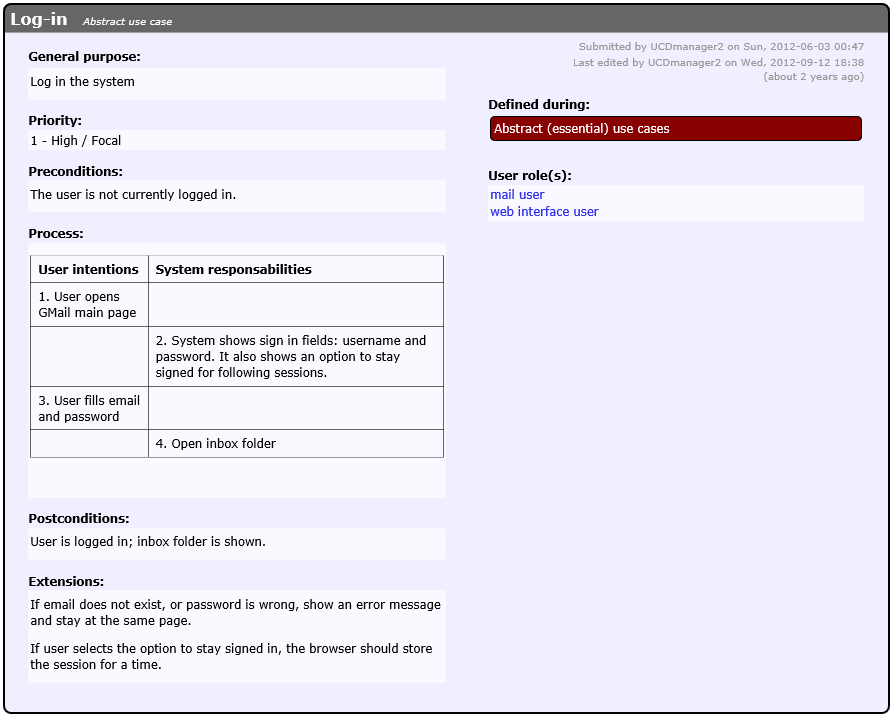


More detailed information here: <http://martinfowler.com/bliki/ConversationalStories.html>

Use Case Scenarios  
Leaves less interpretation to the development team. Explains the behavior the dev team needs to build into the software to meet the user needs. Describes the complete interaction between the software and users (possibly other systems).



Abstract (Essential) Use Case scenarios are another option. Describes the interaction between the user and the system at a high level of abstraction.  The goal of an essential use case is to convey the most important aspects of the user-system interaction by focusing on the user’s intent (avoiding any reference to an assumed UI design or technological implementation) and on the observable result of the system (without specifying the internal steps the system takes to achieve the result).



Example:

Stories that are going to be implemented are placed in the product backlog and are known as Product Backlog Items (PBI’s). The PO and scrum team write down all the functionality they can think of that will create an agile backlog.

Bill Wake’s invest criteria:  
*Good stories tend to be small. Stories typically represent at most a few person-weeks worth of work. (Some teams restrict them to a few person-days of work.) Above this size, and it seems to be too hard to know what’s in the story’s scope. Saying, “it would take me more than month” often implicitly adds, “as I don’t understand what-all it would entail.” Smaller stories tend to get more accurate estimates.*

Tips:  
Include enough detail about anything where you don’t want the programmer to make a decision.  
More detail is almost always better than too little detail.

### Define the Stories Acceptance Criteria

Take the product backlog “(something)” and add “, I expect the following:”. The following expectations should be clear, concise

### StoryTime / Backlog Grooming

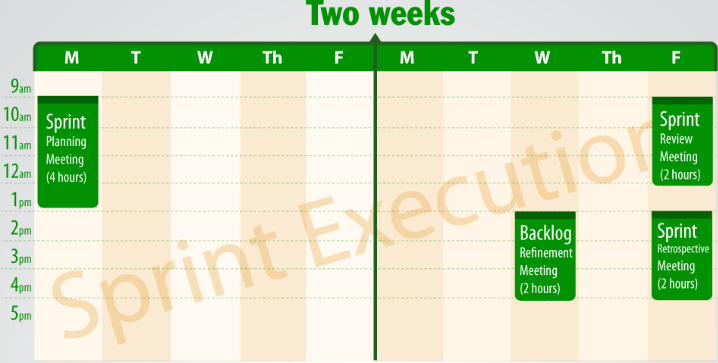
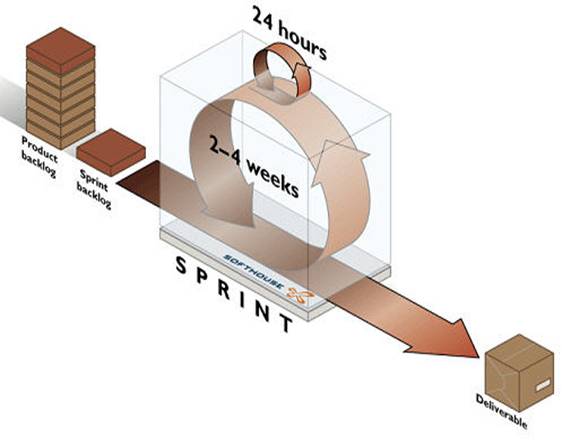
PO and stakeholders create and define stories. Also add acceptance criteria.

### Determine the Length of the Sprint

Take time to determine schedule. Many people like to end the sprint on Friday.

Sprint consists of:

Sprint Planning Meeting  
 Backlog Refinement Meeting  
 Sprint Review Meeting  
 Sprint Retrospective Meeting

### Story Point estimation:

Team members play product poker to define story points. Team members use this time to resolve ambiguity. Product poker allows team members to review the stories, create tasks and use product poker to assign ….

Teams estimate points and complexity using points from product poker. Stories are converted to Epics when they are too large.

Teams create acceptance criteria for individual stories.

### Epics

* Group of many user stories.
* An Epic needs to be broken down into smaller BPI’s
* Prioritize: More or less important
* PO prioritizes the product backlog items

# Start of Project Backlog Refinement Meeting

Please see Backlog Refinement Meeting for details. This meeting takes place at the start of production and during the sprints.

# Sprint Planning Meeting:

## Build a backlog

Team and PO negotiate which product backlog items will be committed to the sprint backlog. The team pulls the top priority items from the backlog and the Scrum Master commits them to the Sprint Backlog Items. The Sprint Backlog Items are broken into smaller tasks necessary to complete the committed PBI. Only 1 sprint is planned at a time.

Sprint planning meeting is made of two parts

**Sprint Planning Part 1:**

Choosing the PBI’s that will be added to the Sprint Backlog.

**Sprint Planning Part 2:**

Creating the tasks for each PBI added to the Sprint Backlog.

Many times the two parts are combined if it’s a small team.

## Who is Involved

PO  
Dev Team

## Time

4 hour time box for each 2 weeks of work.

## Meeting Agenda

* Scrum Master asks the PO if the PBI’s are prioritized the they want it?
* PO moves around PBI as necessary.
* PO Explains the goals of the Sprint.
* Dev Team Choose a PBI and creates Tasks that will get it into a shippable state.
* Team then looks at the tasks and decide whether this is doable during the sprint time period. If OK, then move onto the next Sprint Backlog Item.
* Team then agrees/disagrees if all the PBI’s can be done within the sprint period.
* Scrum Master then asks the team if the team is commited to the PBI’s
* Dev Team does a verbal commit.
* Scrum Master declares the Sprint Planning Meeting over.

# Daily Scrum Meeting

Team stands up during the meeting. Each member speaks during the meeting. The Scrum master facilitates. Team members report to each other. Other meetings can take place, this is the only official team meeting.

A ball can be used to indicate who is talking.

Each team member answers the following questions

1. What did I do yesterday?
2. What will I do today?
3. What impedes me?

## Who is Involved

* Dev Team
* Scrum Master
* PO is optional

## Time

15 minutes per day. Same time each day

Usually done during the morning. If worldwide, adjust as necessary.

## Meeting Agenda

* Scrum Master explains how they are help to help during the Scrum meeting.
* Scrum Master starts with their 3 answers.
* Moves their tasks on the Sprint Backlog Kanban.
* Scrum Master passes “ball” to the next team member.
* Each team member answers the 3 questions and moves their tasks on the Kanban board.
* Scrum Master makes sure the meeting is on topic. Side bar topics as necessary.
* Dev Team members add tasks to the Sprint Backlog as necessary.
* Scrum master concludes the meeting @ 15 minutes.

# Backlog Refinement Meeting

The PO and the Dev Team look at upcoming items in the PB. These are the items that are candidates for the upcoming sprint.

PBI’s need refinement because they are too large or poorly understood. Use this time to help prepare the Product Backlog for the next Sprint Planning Meeting.

The team estimates the amount of effort they would expend to complete items in the Product Backlog and provides other technical information to help the PO prioritize them.

The meeting helps clarify the development items.

Decomposes the higher priority product backlog items. Turn Epics into smaller backlog items.

This meeting is done at the start of the project and at some point a few days before the Sprint Review Meeting.

The first product backlog is based on a vision, business analysis and even marketing promises. The initial list gets the team started working on functionality quickly

## Who is Involved

* PO
  + Explains the current priorities
  + List top priority items
  + List backlog items on Software Program
  + When can it be done
* Scrum Master
  + Explains that the team works together
  + Focus the team members on top items
  + No Management
  + Asks the team
  + PBI’s use estimating cards to get opinion about effort
  + Handles the estimation cards
* Dev Team
  + Use estimation cards

## Time

2 hour timebox

## Meeting Agenda

The result of the meeting are to create Product Backlog Items are:

* (I)ndependant
* (N)egotiable
* (V)aluable
* (E)stimatable
* (S)mall
* (T)estable

1. Scrum master ask PO to tell the team about their current priorities.
2. PO goes thru the top few PBI list
3. Dev Team uses estimate card to determine PBI’s effort to complete it.
   1. Each team members explain their reason
   2. PO may add more information
   3. PBI’s may be found to be epics and made into smaller pbi’s.
4. Dev team determines if the PBI is INVEST.
5. All PBI’s will not need to be completed.
6. Scrum Master declares the Backlog refinement meeting is over. No other work is need since the other meetings will assign PBI’s to Sprint Tasks, etc.

# Sprint Review Meeting

The purpose of the sprint review meeting is to have the dev team show the work they have completed. That work should include a potentially shippable product increment. The team performs a live product demonstration to the product owner. Anyone interested can attend this meeting.

## Who is Involved

PO  
Dev Team  
Scrum Master  
Outside Stakeholders / Anyone

## Time

2 hours

## Meeting Agenda

* PO reminds us of the purpose of the sprint
* Live product demonstration
  + Team members show or have the PO attempt to do the PBI tasks.
* Scrum master explains to the PO that they should explain to everyone which PBI’s are considered done.
* PO reviews each PBI and sets the PBI’s as DONE. Done = meets the acceptance criteria.
* PO makes sure sprint goals are met.
* Items done are counted as velocity.
* Items that did not meet the acceptance criteria are not done. Task items not done or almost done are moved back to the backlog. Do not move to the To Do. Priority may also change.
* Scrum master adds PBI’s to the site.
* Completed items are “Torn up”.
* Anyone can submit PBI’s (who, what, where).
* Measure Velocity  
  Anyone can do this.

Combine the total PBI for the total Story Points.

Adjust Scope to hit release dates.

* Stakeholders feedback
  + Team listen to feedback from the stakeholders about the product.
  + May result in new product backlog items.
  + Can add to the BPI.

# Sprint Retrospective Meeting

Discuss what went well in the sprint. What could be improved on the next sprint, what we learned and what still puzzles us. Give feedback to each other.

## Who is Involved

PO  
Dev Team  
Scrum Master

## Time

2 hours

## Meeting Agenda

Scrum Master can perform the following to understand how the team is doing. This can be done for sprints, release and team history. Team should write out a few actions at the end of the meeting. It must be written by the team.

* Safety Check
  + Each team member is given a blank ballot.
  + Ask a Question about being awkward, and most comfortable.
  + Write a number of how safe the use feels in the team
  + Do not reveal to anyone.
  + Collect and tally votes
    - If votes are low have people break into small groups.
* Ask the following questions to the team to have a conversation.
  + What did we learn? What puzzles us?
    - Actions
  + What went well? What can be improved?
    - Actions
* Silent Writing
  + Stack of post it notes handed to team.
  + Take 10 minutes to write down ideas.
  + Write one observation per post it note about the previous sprint.
    - Good, bad, Neutral.
  + @ 10 minutes, stick notes on the board and discuss.
  + Write down the actions.
* Timeline Retrospective
  + Draw a timeline on board.
  + Write down tasks and put them on a timeline.
  + Read each other notes and discuss what it means.

# Problems to Avoid

What may cause development to get behind schedule:

* Failure to control the scope
* Unrealistic external commitments
  + Estimate Costs
  + Contingency Margin
  + Profit
* Excess multitasking and distraction
* Poor technical practice

# Example User Story

