**SCRUM THE BASICS**

**PROBLEM TO SOLVE**

Agile project mgt has taken the business world by storm for the better part of the last two decades. And most recently scrum is dominating the landscape. It may sound mysterious but at its root scrum is a set of simple processes and practices that can help you harness the speed of change. You will be learning from Kelley O’connell and she has been project management for more than 20years. She spend the first half of her career executing waterfall style projects.

But more recently she has been practicing and coaching agile, specifically scrum. In this course we’ll explore the basics of what scrum really is and how you can use it in your work environment. First we’ll talk about why so many organizations are moving to scrum. Then we’ll look at how scrum asks you to organize your work and your team. Finally we’ll discuss how to manage and measure your work on a continual basis. Our hope is that by the end of this course you’ll be well on your way to understanding how scrum has helped many companies and how you can take advantage of it, too.

**THE SCRUM APPROACH TO PROJECT SUCCESS**

Languages changes around us all the time. It’s like a living breathing thing. As we use and adapt words new defitnions are created to match what they mean in current times. Take Goggle for example when the words was originally coined in the 1930s it meant a numeric value of 10 to the 100th power. Fast forward about 70 years and with a spelling tweak it became the name and sole product of an internet company. Just a few short years after that it was transformed again from a noun to a verb about how to find an answer to really any question you can come up with.

Similarly the Scrum we’re talking about is an adaptation of the word scrummage taken from the game rugby. Now, Google going from a company name to the name of their application to the action of the app, kind of make sense. But scrum from rugby? How does that work? Well in rugby scrummage, scrum for short was the method used to restart play in a match after a foul. Visually its eight players form each team packed together with heads down all trying to take possession of the ball.

So, not exactly the poster child for project management but with a little imagination it makes sense. On a project team the goal is to get the project done. Historically using traditional methods this meant planning designing the whole project at the beginning and sticking to that plan with no variation. In real life project work is completely unpredictable. It’s impossible to know at the beginning exactly how a project will unfold and how to best meet its unique challenges.

The Agile project management movement comes out of a desire to adapt in real time to the changing circumstances that teams face. And this is where the rugby team comes in. the founders of the Agile movement recognized that in rugby the object is to move the ball down the field one possession at a time. So why couldn’t project do the same thing? Why not change the focus from just winning the whole game to winning each and every milestone and deliverable.

These innovators co-opted the word scrum to reflect this new approach an approach that breaks the deliverables and milestones into smaller pieces and gets the whole team together to focus on just that one goal until it’s done. Like in rugby if the small individual scores happened on a regular cycle, winning the game or delivering the project would take care of itself. So the sporting word scrum has been transformed in the last several years to have a new meaning.

It means to run your projects more like a rugby match, pursuing the small goals and deliverables that will get your project done.

**THE AGILE REVOLUTION**

Being a traditional Waterfall project manager made it really unlikely that your project would be successfully completed. Let alone according to the plan you laid out at the beginning. It was more like being a weatherman doing a forecast for a specific day next year. Except for random luck you’d be wrong. Waterfall as a methodology is not inherently a bad thing. In some cases it makes good sense like in a building construction where a predefined set of steps when executed in order will result in a building.

You can absolutely plan and schedule that whole project up front. It happens every time a home or an office building is built the problem comes when applying the technique to highly empirical work, like software development. Empirical work is more like a science experiment. You try something check out the results and if it didn’t work you try something else. You certainly can’t do that when building a house but with software or some other projects you do it every day. That’s the crux of why Waterfall didn’t work well for software development.

You literally cannot upfront plant the process of discovery. The frustration of highly skilled software developers working on Waterfall projects was the tipping point that led to the agile revolution. Tired of having a failure rate equivalent to a weatherman these individuals decided they had to come up with a better option. The result is known as the Agile Manifesto. Grounding themselves in the mindset of learn manufacturing where you do just enough just in time to meet the goal.

They started figuring out how that applies to software development. The result is the agile Manifesto and its underlying principles. Take a look at the Agile Manifesto. You can check it out at this online address.



<https://agilemanifesto.org/principles.html>

As if these words weren’t revolutionary enough this group of innovators supported this manifesto with 12 key principles. This manifesto and the principles became the foundation for a new set of project management and software development methodologies.

There are a couple of overriding themes that make Agile different. One of the key changes is that we’re asking our business partners to work with us throughout the whole project. Not just how up at the beginning to describe what they want then show up at the end and tell us how we missed the mark. We need direct, ongoing interaction to deliver what they really need. Another key is that we no longer want to measure success using milestones and project phases.

We want working software to tell everyone how we’re doing and we want to hear feedback the whole time. Perhaps the most revolutionary change is to allow teams to self-organize. They’ll do a much better job doing the design and tests from the ground level, then any upfront plant could do. Upfront planning is theoretical. Evolving design is both practical and tactical. It’ll get you to your goals faster with higher quality/

Between the manifesto and its principles this group of developers was done being the high-tech software version of the weatherman. They were ready to succeed. There is a better way to do software development and it’s Agile.

**SCRUM: THE LEADING PROJECT METHODOLOGY**

Scrum wants you to fail. In fact, it’s known for the slogan “fail fast” no I’m not joking. I realize it sounds weird but there’s a very good reason for this. Traditionally, project managers and developers would work for months for years before seeing the results. Most of the time around 80% in fact, the software and projects failed. So why you might ask are they signing up for more failure? Well really they’re not.

The trick is in focusing on the second word: fast. Failure is okay as long as you’re learning from it but if you have to wait too long you’re not going to learn nearly as much from it. Scrum takes the agile manifesto and its key principles and boils them down to a very simple framework that encourages small-scale focus and rapid learning cycles. That’s what fails fast really means, learn fast.

With that in mind, the basics of the framework are designed to encourage that fast feedback loop. The Scrum framework is not prescriptive. We generally refer to it as guardrails, like the ones you see on the highway. Those guardrails don’t tell you exactly where to drive in the lane, but they keep you within the boundaries that will result in a successful road trip. Scrum is much the same. With the Agile principles as the guideposts and a loose framework of activities executed on a regular short schedule your project will be set up for success.

In a nutshell, here’s the Scrum framework in its simples form. To start, the product owner has prioritized backlog of work for the team to do. Every two weeks or so, the team looks at the backlog and decides what they can accomplish in the next two weeks. The team develops and tests their solution to the backlog items until they’re done and ready for use. At the end of the two weeks the team demonstrates their accomplishments to the product owner and stakeholders.

Finally, they reflect on how things went during the two weeks and they decide what they can do to improve their work practices. That’s it. The short time frame and the focus on complete product at the end forces the team to fail fast. Or more appropriately learn fast.

**THE HEART OF THE FRAMEWORK**

**SOLVE PROJECT PROBLEMS WITH SCRUM**

Traditionally waterfall project teams face three constraints: time, cost and scope. They’re unable to change any of these things once their project starts. The problem is, curing the course of a project, the business environment changes around you. This means, by the time you’re done, what you build is no longer valuable. This isn’t anyone’s fault. Business needs are changing more rapidly than ever. Project requirements are shifting just a quickly to keep up.

So teams were doomed to failure on every project until the agile manifesto came along. Then we shifted our focus away from constraining all three project elements and decided to make one of them flexible. Scope that was huge. You, the business person can have this team, cost, for this long, time and you can build whatever you believe is most valuable. It’s that simple. All Agile methodologies follow that first, key paradigm shift.

Lock everything but scope and you’ve got yourself a framework that will deliver exactly what’s needed as quickly as possible. Agile is a broad umbrella of many methodologies that follow the same principles. Scrum is one of them. Scrum took it a step further by creating a framework to help teams stay focus and protect them from distractions. Essential to Scrum are two key roles the product owner and scrum master. The Agile manifesto writers recognize that historically they didn’t have access to the right business experts as often as they needed to help guide their daily decisions.

Scrum solved this by creating the role of the product owner. This business representative is committed 100% of the team. It’s their full-time job. They also realize there needs to be someone to help the team resolve day-to-day issues and counterbalance the ongoing requirements changes. So, the role of scrum master was developed to protect the team enough to complete work without distractions. This dedicated role also helps them improve their internal team processes.

Beyond creating these roles to help the teams, Scrum says you need to deliver quickly so you always know whether you’re on tract. In order to fail fast and learn fast, you need a fast feedback loop. Scrum set the boundary for teams to deliver values as anywhere from two to four weeks. You need to have business-approved and customer-ready product completed that often. Scrum also recognizes that if you’re going to successfully deliver this quickly the team needs to meet every day.

This is what’s known as a daily stand-up meeting. Finally, the last key element of this framework is the recognition that a healthy team needs time to reflect and think about what they can do to improve. So Scrum mandates a ceremony called the “retrospective” where they can assess themselves and decide what to improve. Really, that’s it for the frame work. The focus of Scrum is to maximize the efficiency of the team. Since scope is the point of flexibility for agile projects, Scrum’s focus is how to deliver the most possible value within the constraints of time and budget.

**ESSENTIAL ROLES FOR SCRUM TEAMS**

Scrum is lightweight framework that can incredibly flexible, efficient and powerful but much like a vehicle the best body style and frame will get you nothing without a powerful engine to move you forward. There are two key roles that exist on every Scrum team: Product Owner and the Scrum Master. Let’s start with the **Product Owner or PO**. The PO is the business representative on the team. They’re not part-time team members. They show up every day, because they’re contributing to the final product every day.

They review all the work team completes and either accepts it or asks the team to make changes to ensure the highest values is being delivered. Formerly, the business person was represented through requirements documents that were rarely if ever updated. On a Scrum team, the PO is always ordering the work and ensuring the team members clearly understand the details of the request but that’s only part of their job. They’re also interacting on a daily basis with the stakeholders.

It’s not enough for the PO to interact with the team, they must also be in tune with all the changes that are occurring in the business context. As a result the PO is the keeper of the product vision. He or she defines and manages the backlog of work to be done and the prioritization of those work items. Remember that Scrum allows the scope to be flexible. Since time and cost are locked, the PO is painfully aware that the work must be continuously sorted to highest value first.

They’ll also be pushing the team complete as much work as possible in each short delivery period. Now if you’re wondering how on Earth a team can keep up with these demands, you’re not alone. The founders of Scrum recognized the need to counterbalance the PO role, so they created the role of the **Scrum Master**. The Scrum Master protects is the facilitator who keeps the team within the guardrails of Scrum.

They balance the demands of the PO against the needs of the team. This role is the first safety value to ensure teams are performing at a sustainable pace. We don’t want them to get burned out before they reach the finish line. That’s big statement if you thing about it. Scrum is a framework that doesn’t value heroics by teams or individual team members. It values sustainability and open dialogue on what can cannot reasonable accomplished.

The Scrum Master is most visible spokesperson for the team. Scrum Masters value transparency. They’ll devise charts and boards to share the team’s progress with anyone who’s curious or interested in how they’re doing. They’re also the first escalation point when something gets in the way for the team. The Scrum Master will work to remove any blockers until they’re out of the way and the team can continue on. While the PO focuses on what needs to be done, the Scrum Master focuses on how the team does the work.

And one more thing. The Scrum Master also holds the team accountable for their commitments to the PO. They show trends in team performance over time to help the team improve their processes and practices. As you can see each role is absolutely critical to getting the framework to function properly. If you’re lacking one of these roles, Scrum will be far less effective than if you have them both.

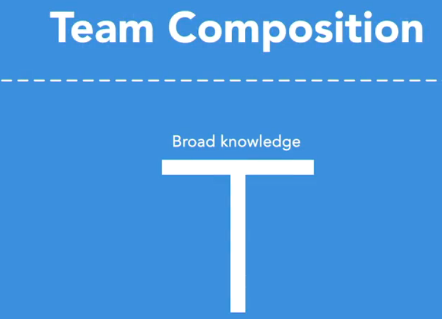
**ESTABLISH YOUR SCRUM TEAM**

Once I was in a position where I was given an assignment but I didn’t have all the tools I needed to complete the job successfully. I felt like I’d been set up to fail. So when you’re ready to set up your scrum team I hope you’re planning to do everything you can help them succeed. Scrum is all about daily collaboration and communication. Absolutely anything you can do to make that easier for your team will pay big dividends down the road. That’s why if it’s possible, co-locate your team members in the same room aisle or space to make that collaboration more effective.

In many companies that’s simply not possible due to building or distance barriers that’s okay. You can still be successful with scrum you’ll just have to work bit harder to ensure sufficient collaboration can take place. Some things you can try are video conferencing, dedicated chat rooms and conferencing phones in all your locations. These can easily help your team stay in touch. Now, let’s talk a bit about team composition. You’re first priority is to ensure you have dedicated team.

If your team members are multi-tasking between your project and another one, they lose a lot of efficiency and that slows down your delivery. Dedicating your core team results in better focus, higher efficiency and faster delivery. The ideal team size is seven members plus or minus two. I know this sounds specific but research indicates these numbers maximize people’s ability to create close relationship and collaborate most effectively.

Sure you can have more people but the communication channels it takes to keep everyone in the loops gets much, much harder for every additional person you add. In a perfect world, you’d have a team made up of five to nine people who have absolutely every skill you need to complete the work. Those are what we call **T-shaped resources**.

****

They have broad knowledge at a high level in several areas and deep knowledge in just one area, like a T.

For example, I worked with a very skilled Java Developer, the vertical of the T who also had database analyst skills, the cross bar of the T. so she could work on more than on type of deliverable. These types of resources can be hard to come by so you’ll want to line up some consultancy resources that can be used when your team needs specialized skills. Specialized skills that I’ve seen fairly frequently are architects, database analysts and security analyst.

These are people that you’ll need once in a while to work with your team. But you may not have enough work to keep them fully occupied. This is a very common and with the right level of up front commitment and ongoing communication it can be highly successful. A small unit that works closely together all the time will inevitably face internal conflicts. Some up front ground rules are needed to prevent trouble. We call these team norms and you need to help the team establish these before any work is even started.

These norms are commitments that the team members make to each other about how they’ll work together. How they’ll resolve disagreements and how they’ll reach design consensus. Just to give you an idea one of the most common scrum team norms is to agree to disagree but move forward with the decision the teams made. Another common norm is something like when laptops can be used during meetings to make sure everyone is focused on the conversation.

You can even have norms on how to hold each other accountable. You can get very creative with team norms but they must have the commitment of all team members for them to be useful. They must also be fairly applied and the team must trust each other enough to hold each other accountable to honoring the norms. If you take the time to work through these elements and help your team set the stage, success can come more quickly for them. You’ll be amazed at how quickly your team can work effectively once you’ve set them up for success.

**GETTING READY TO WORK**

**SET THE VISION ON YOUR PROJECT**

I’d never go on vacation by showing up at the airport and randomly selecting a destination based on the flights they’ve got that day. I don’t know about you, but I’d rather choose my destination and my activities before I start my trip. That way, I’ll get exactly what I want. Your scrum project is exactly the same. The vision is the map that will guide you and your team to your destination. The product owner establishes this before any work begins. The vision tells you, your team, everyone really, where you’re going and what valuable product or enhancement you’ll deliver at the end.

The destination contained within the vision is what we call a minimum viable product or MVP. The MVP is about developing a product just enough to get it out the door early adopters. They can then get feedback from those end users. This approach is basically saying that if we do a small set of work to get out usable product out the door, we’ve met the goal. There are couple of good reason for approaching the work this way.

First, the faster we get something into our stakeholder’s’ hands, the faster we get feedback and understand what else is needed. Second, by keeping this fast delivery in mind we minimize the possibility of scope creep or gold plating where unnecessary things are added along the way. Discipline and execution is still required, but now you have a way to watch for it. Once you have your vision in hand, it’s time to start decomposing the vision into its functional parts.

Here’s vision we can use. Help busy professionals have more time and feel healthier by providing a mobile app where they can quickly order an affordable, tasty, healthy lunch and have it delivered or pick it up themselves. The PO can do this in advance, but it’s absolutely essential that the team fully understand these functions. Only with that awareness will be team truly understand their role in delivery to the vision.

The first level of decomposition is to identify your themes. Themes are just a broad grouping of similar work. Restaurants do this every day. Appetizers, salads, entrees and desserts are the themes they work with. These aren’t simply to help customers read the menu. They’re also useful in organizing the kitchen. You’re probably making dessert separately from salads, right? The same is true for your themes. Using our sample vision of a mobile app for customers to order lunch our app themes might be: profile, order payment and delivery among others.

These themes help us in two ways. First, they help trigger ideas about what needs to be developed to meet the MVP for that theme. Second, they help us group our work together so we can be efficient and minimize risk, like ensuring we have security built before we complete the profile development. Once we’ve identified our themes, we break them down further into features. Features are the next small slice of work we can use to help us stay organized.

Using our example, we could break the theme of profile into features like log in, save password, recent orders, favorites and recent locations. Knowing that we’re targeting an MVP, our product owner may say that the only feature we need for this release are security, log in and save password. The others while nice to have aren’t required to complete the initial release of our product. In this way you and your team can use the vision and the MVP to stay focused on your destination and remain organized along the way.

**WRITE USER STORIES**

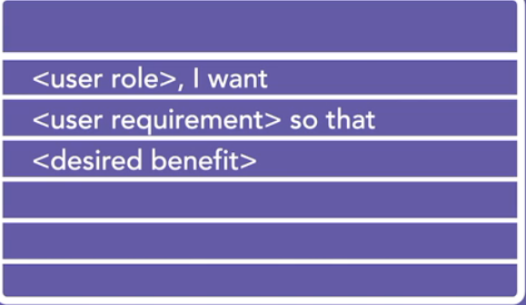
We’ve talked about themes features as groupings of work. These are really useful constructs to help us get organized but they’re still too big for a team to work on and deliver value in small timeframes. The people we’re developing for our customers or users. Each interaction they have with our product is a use case that tells the team a story about how they’re going to use our product. We call these user stories and this is the level of detail we need to know what to delivery.

The user story then is the tactical level of work that can be delivered quickly. At the same time, they’re not so small that they deliver no value. Anyone on the team can write user stories but it’s usually the PO. As the stakeholder and representative for the customer it’s their duty to ensure everything in the backlog adds value for the customer so they usually write the user stories. When writing user stories, its best to use Bill Wake’s invest acronym as a guide.

First, good user stories are independent. It can be delivered separately from other stories and have value by itself. If you only have time to do one thing this story can stand alone. They’re also negotiable. Until the story is committed for work, it can be rewritten changed or canceled at any time. User stories must be value. It delivers value to the PO stakeholders and customers.

Simply put, it’s meaningful. Next, they also need to be estimable. You must be able to estimate the size of the story in story points. That means that the story is descriptive enough so you know what has to be done to finish it. Only then can you understand the effort required user stories must also be small the story is enough to be completed in one sprint. And last testable.

The story provides enough information that you can develop tests for it. While these seems like a lot of accomplish with a user story, we have a format that helps us write good ones. Here it is.



This is simple format tells us so much. By understanding which customer we’re talking about, we get a better understanding of the need and it helps you focus on an activity within our product. Clearly defining the need and most importantly the benefit expected from that requirement helps trigger the conversations the team must have in order to be clear on what is value.

Using our mobile lunch ordering app as an example of a good user story could be as a mobile customer, I want to create a profile so that future orders are faster to place. This is known as a functional user story, meaning it serves a function for the end user. You’ll also write non-functional stories. Those stories help support the functionality the end users need without directly benefiting them here’s an example.

As a developer, I want to upgrade the database software to the latest version so that we have a supported product. Now, it’s not good enough to have a good user stories each one must also have **acceptance criteria or AC**. The AC is the most powerful tool the team has to reduce errors in getting a story to done. Whenever user story is written, the PO and the team collaborate to determine the AC for the story.

So each story has its own unique AC. In the case of our functional story about creating a profile the AC could be that customer name is captured and saved, next customer email address is captured and saved, and also, customer phone number is captured and saved and customer password is captured and saved. Additional AC for this story would ensure that invalid phone numbers, addresses and passwords are rejected.

Your AC should be as explicit as possible, so all parties on the team know what done for the story really is. User Stories are the tactical tool scrum teams use to deliver the work for their product. Writing good user stories can be challenging but with practice its gets easier every time.

**SET BOUNDARIES FOR SUCCESS**

At this point, you should have your themes and features broken down into user stories. Also, you’ve estimated your stories using points so you know about how much work is waiting to be done. This you product backlog. Your next step is to create boundaries that your team can use to complete this work. These guidelines include the team’s definition of done, prioritizing your backlog and establishing your sprint cadence. First, your team needs to define done for their stories.

I know each story has its own acceptance criteria but this definition is a little bit more general. This definition of done states the minimum requirements for all stories. Since these apply to everything in your backlog, you don’t want to repeat these things in the AC of every story. Creating this overarching definition of done helps save time while allowing more precision in the story specific AC. Examples of done criteria for all stories are for a story on your team to be considered done, it has been tested in the pre-release environment or how about for a story on our team to be considered done, it has been code reviewed and all errors fixed.

Next, your product owner needs to be prioritize the backlog. **This is known as backlog grooming**. It simply means that stories are continually sequenced in value order. The more valuable the outcome of the story the higher it is in the backlog. Remember, Scrum allows flexibility in scope, but is locked in duration so the PO wants the most valuable stuff delivered first. That way, even if time runs out, a valuable product has still been delivered.

Finally, you need to establish your spring duration or cadence. Scrum says that your spring can be anywhere from one to four weeks in length, with a preference towards the shorter time scale. Remember, the Scrum best practices say we should fail and learn fast. With that being the case you should establish the shortest feasible sprint length you can. When I work with teams, I generally recommend they go with a two-week sprint cadence.

This duration really helps the team stay focused on what’s happening within the sprint. A two-week sprint is in the Goldilocks zone. If your sprint is too short, teams tend to panic and quality degrades. If your spring is too long, the team will unconsciously relax their pace. They feel like they have plenty of time, but this middle zone of two weeks is just right. It creates a sense of urgency within the team without inciting panic.

Establishing these boundaries helps the team understand what done means for every story, the value of specific stories and the timeframe in which they’ll be delivering. They now have the execution framework they need to be successful.

**STORY POINTS AND ESTIMATION IN SCRUM**

There are two kinds of estimation we all use every day. There’s actual estimating and relative estimating. Actual estimating is what you use when reading a map. It’s 25miles from Point A to Point B. this is a very specific. Then, there’s relative estimating, which is comparing things to each other to get a general idea of something. Like a giraffe is twice as tall as a zebra or cake is the same width as a pie.

In Agile scrum, we use both kinds of estimating. We use relative estimation to get a rough size of our work by comparing user stories to each other. This gives us an overall sense or estimate of how big something is. Stories themselves are rough guides to how the user wants to interact with our product. Because it’s a rough statement of need, we can’t be too specific on how big it is. Also, since this is just a rough cut, we don’t want stakeholders to thing we know precisely what it’s going to take to get that done.

Relative estimating helps us maintain the mindset that it’s just an estimate, not a commitment. At the same time, when we commit to tasks, we better be pretty darn clear on what it takes to do that task. So when it’s time to do the work we need to be specific on how long it will take so that’s when we switch over to actual estimation. Scrum teams use planning poker to guide them relatively estimating their stories.

Story points are the unit of measure we use to convey the relative sizes. Here’s a great detailed course on how to play planning poker with your scrum team. Estimation is meant to be lightweight and fast. It should take days or even hours to determine how much effort you’ll put into something. While it might take a little bit of practice, once you get that hang of it, you’ll fly through it.

**CREATE A ROADMAP AND RELEASE PLAN**

You’ve done all this work preparing your themes, features and backlog. You also know how long your springs are, now it’s time to estimate when things will be worked on. Scrum has tools for this as well. We use two different mechanism for this the **roadmap and the release map**. The roadmap is very high level and is intended to apply your themes over time. This way everyone has a general sense of what the focus will be in a given time frame.

Using our example from earlier, creating a mobile lunch ordering app, we identified themes of profile, order, payment and delivery. Now we need to decide the best order to work on these themes. While payment may be the most important to our business partners from a dependency perspective, it probably makes more sense to prioritize the profile and ordering themes first. Like everything else in Scrum this a guideline not a rule, but following this idea here’s what the roadmap would look like for these themes.

As you can see these themes aren’t necessarily in their value order. Once you’ve completed this you can organize your stories around the theme timelines you’ve created this is how you create your release plan. The release plan is the next layer of detail. Its high level plan that’s meant to connect the roadmap to the sprints. It provides visibility to how we’re going to deliver. In Scrum you must have fully functional stories completed at the end of every sprint.

You’re not, however, required to release them to the stakeholders at the end of every sprint. This means you can complete all the work for two or three sprints before releasing the combined results together. You’ll use your story points to help your team decide that they do within each sprint. Once your team has been together for a while they’ll reach what’s known as a **stabilized velocity**. Meaning that everyone will know about how many story points they can complete in each spring.

Until the velocity stabilizes you’ll estimate the number of points to target for each sprint. For example, let’s say that our team is just starting out and we collectively thing we can handle about 10 points per two weeks spring and we want to release after three sprints, that means about 30 points per release. Now let’s take a look at our roadmap and the stories we’ve organized over time based on your themes. All we need to do is select the highest priority stories by size to fit into each sprint.

It should look something like this. You might notice I’m not planning those in priority order. At this level of detail it’s about maximizing the number of points per sprint. So if the next highest priority story. Story C for eight points is too big for sprint one we go down the list until we find one that will fit. Story D five points. Always remember that your roadmap is an estimate or when the team will complete these stories.

The roadmap is intended to be updated at the end of every spring. The team will be learning things and writing new stories through the sprint. So the roadmap is meant to be living document. It’s just a guide, but it’s a guide that will help you stay focused on getting the project home.

**GETTING DOWN TO WORK**

**SPRINT PLANNING IN SCRUM**

As your team prepares for the spring more detailed collaboration is necessary and that’s exactly what you do in your spring **planning meeting**. Full participation is critical so you’ll need all your development team members, your scrum master, and product owner. These responsibilities can’t be delegated. Using the prioritized backlog, the PO presents the highest value stories to the team in order. The team needs to feel confident and comfortable enough to ask questions.

The goal here is that everyone on the team fully understands the intent of the story specific AC for that story. It’s also helpful to post the team’s definition of done for all stories in the meeting room. Remember, the team’s definition of done applies to everything, so having that information handy is going to help ensure all the tasks are identified. I can’t emphasize enough how important the question and answer aspect of sprint planning is.

At the end of sprint planning, the team is going to commit or make a promise that they’ll finish all this work. Errors is understanding must be avoided and this is the forum to clear the air on the stories. Once everyone has clarity on the details of the story, the team writes down the tasks needed to get each story done. Let’s look at our lunch ordering app as an example. For the story create user ID, the task would be, write new use ID to the database and should take about half an hour.

Let’s take a look at tasking our stories. Each task should be clear and estimated using time. At the sprint level, points are used to identify stories for inclusion in the sprint. At the task level, time I used to fill up our capacity, hours that each team member can take on. Now that you know the number of hours for all the planned work in your sprint, you need to verify that the team in an eight hour working day, people usually have only about six hours of productive time.

The other two hours are generally filled with phone calls, interruptions, and emails and such. We’re trying to prevent over-committed so this comparison of necessary versus available hours is a good confirmation step. Finally now that you’ve got the stories clearly explained and the tasks defined for each one, it’s time to commit to the work. Scrum takes the commitment step very seriously. Each person on the team is asked whether they commit to the stories in the sprint and each person is expected to say either yes they do or no they don’t and why.

If someone can’t commit the PO and the team need to work together to change the shape of the sprint until everyone can commit. Sprint planning is a key collaborative effort in scrum. You’ll repeat this meeting at the beginning of each sprint and you’ll become better at every time.

TRACK YOUR SCRUM PROGRESS

If you’re working on a critical project it’s pretty common for stakeholders to ask your team members how things are going. This is good. People are invested in the outcome of you project. It can be a distraction for your team members have the full perspective on how things are going and others may not. Scrum addresses these challenges by posting information radiators. An **information radiator** is anything that you post on walls or team sites that help everyone understand what you’re doing and how it’s going.

These can take as many unique forms as needed for your environment and project. I’ve seen teams visibility post the vision for their project the team norms, their team’s definition of done, their roadmaps and the release plan. At a minimum teams post their task board and their burn down chart. So let’s talk about those in a bit more detail. The task board can take any form you’d like but it has a few key components.

It shows the stories committed to in the sprint, the tasks and their current status and what tasks have been completed. This is a simple example of what a task board looks like. It clearly shows the committed stories and the related tasks. It goes further by showing what tasks have and have not been started. Finally it shows which ones are done. This simple tool very easily tell everyone what the team is working on. This board becomes the heart of your sprint execution.

The team will gather around it to track what’s happening in their sprint and help each other if it looks like someone is struck. Another primary tool for sharing information on your progress is the sprint brundown chart. This is used by the team to measure how well they’re executing the sprint. Where the task board tells you about task completion, it doesn’t tell you how that compares to where you are in the sprint. A burn down does that. Here’s an example of burn down.

As you can see the burndown tells you exactly where you are in the sprint by day and how much work you can committed to do in the whole sprint or hours. Ideally you’ll burn down in a linear fashion. The actuals line tells you exactly how you’re doing in comparison to the ideal. This is a very powerful tool showing the team and all the stakeholders how the team is progressing. Easy to create. Easy to use and understand. These are the hallmarks of great information radiators.

These will serve your team and well and your stakeholders will appreciate having a clear status.

THE DAILY SCRUM OR STAND UP MEETING

The scrum master as the scrum process owner will establish your daily scrum. This is also commonly referred to as the daily standup meeting or stand up for short. In order for scrum for work, it relies on the three C’s: collaboration, communication and cadence. The standup includes all three. It’s the foundational element of collaboration. It includes everyone on the team: the developers, the testers, and the PO and scrum master.

It occurs at the same time every day, so your scrum master will select a time for the standup that works for everyone. For teams that are co-located or within a few times zones, this is pretty straight forward. For international teams, this can be a bit more challenging. But with today’s video capabilities, this is surmountable as well. I’ve seen teams where members in one country film their standup meetings and send it over for team members in other countries to watch as part of their standup.

The bottom line is that your daily standup is one of the non-negotiable of the scrum framework. It’s usually held in front of the team’s task board, whether it’s electronic or physical. This is the time when task are moved across the board. At the same time your standup is not a progress report. No one should be delivering the blow-by-blow of their activities just the overview. Stakeholders are also encouraged to attend but ask them to hold their questions and comments until the end of the scrum.

Daily scrum is short. It’s limited to 15 minutes. It can be shorter, but I cannot be longer. The whole team stands to keep it fast, thus the shorthand name stand up. Really if everyone I sitting down, it starts to feel like a meeting and it can drag on like meetings tend to. The scrum master also limits updates to three questions per person. “what did you do yesterday?” “what are you doing to do today?” and “is anything blocking your progress” as individuals go through these questions, the whole team understands their progress as a unit and sees how their work first into the whole.

At the same time, if a team member didn’t complete something the day before or for several consecutive days, this is the team’s opportunity to hold them accountable. This isn’t done in a mean-spirited way. It’s simple hard to face your peers and admit you’ve missed your goal. It’s a daily opportunity to ask for help. One of the most powerful aspects of scrum is the immediacy in which issues are resolved.

If someone on the team says they have something blocking them the whole team has the ability to step in and help. It can’t be resolved within a day by the team, the scrum master steps in. if the scrum master can’t fix it in a day, it goes to the PO to help solve the next day and so on. These escalations then continue until the blocker is removed and progress can be made again. The daily scrum is the expected cadence for communication and collaboration.

Scrum teams collaborate and communicate all day long. The daily scrum is just the formal version so everyone gets to see and hear the teams collective. Progress.

BACKLOG GREEOMING IN SCRUM

In scrum the product owner is accountable to the team and the stakeholders. The stakeholders want to ensure they get what they need from your project. A key aspect to keeping the right priorities throughout the project is the ongoing refinement of the product backlog. This means that the PO is out there meeting with stakeholders to understand whats needed. They’re constantly moving the work around as the business needs and requirements shift. Through the project your backlog is constantly changing as new features and stories are being discovered and added.

Other stories are being changed and maybe even removed. As new stories are discovered the PO works out the details and AC for each one. But at least once per sprint the whole team comes together to see what new items have come up. This is usually a 30 to 60 minute backlog grooming session. It usually happens around the midpoint of the sprint. The PO reads the new stories to the team and the team asks about any details that are missing.

The PO then goes to find answers to the team’s question. Once done, the PO decides what the priority of the new stories are and adds them to the backlog where they think the stories fit. Now this might sound a little scary but there’s safety net in place to protect the team and the sprint commitment. The PO can only add stories to the future sprints. The sprint that’s in progress is off limits. So yes, your backlog will change through the life of the project.

But no, your sprint commitment can’t be changed once the sprint is underway. This helps the team keep their commitments but also allows the Scrum team to adapt to the changing business needs throughout the life of the project. The other advantage of ongoing backlog grooming sessions is that the team gets to ask the PO for clarification, well ahead of the sprint planning and commitment meeting. If the details can’t be ready in time for the sprint planning meeting, the story can’t be added to the sprint.

This ensures there are not surprises once a sprint is underway. These interactions also help the PO known where there are gaps in the requirements. It empowers the PO to fill those gaps in a way that doesn’t delay the team. The team gets to keep working on the current sprint while the investigation happens. Backlog grooming then is one more way in which business needs can be continuously identified and prepared for delivering within a spring.

It’s a great tool to foster understanding and collaboration across the whole team.

WRAPPING UP A SPRINT

GET STORIES DONE IN SCRUM

In Scrum, the team takes their sprint commitment very seriously. They’re striving each day to collaborate and get their stories to done. Remember, in Scrum, done means usually product that meets the acceptance criteria. Since that’s the goal, the team is developing and testing the entire time, to ensure all the story AC are met. The AC approval through, has to come from the product owner. Since the product owner is the business representative on the team, their acceptance or rejection is the final word.

That means that throughout the sprint team members are checking their work with the PO. Each story that’s accepted can then move across the task board to the done column. Some teams hold a brief more formal session known as the sprint review. During this ceremony, the team and the PO meet to the review the sprint as a whole. This is a checkpoint where everyone looks at each story from the sprint and sees which ones were done. Anything that isn’t accepted as complete is reviewed, prioritized and moved out to another sprint.

In some cases, the team has discovered new information about a story and it needs to be split into smaller ones. Once those are split, they’re also prioritized by the PO and then moved to the appropriate sprint. Fianally, the team collectively understands and agrees on what was done, and can be demonstrated to the larger stakeholder group. This meeting is meant to be short and simple. What did we do, what do we still need to do in a future sprint, and what’s ready to show off? This session is another step in Scrum collaboration.

The team acknowledges their progress and keeps their eyes on what’s coming. This collaboration helps the team maintain awareness of the whole project. It keeps the team focused on the bi picture while delivering a usuable product in every sprint.

DEMO THE TEAM’S WORK

In Scrum, we strive to deliver a working product at the end of every sprint. But what’s the point if no one knows about it? Scrum’s answer to this is the demo. As you know, the PO is accountable for accepting or rejecting what’s been delivered. Whatever they accept, is ready to be demonstrated or demoed to a larger audience. Remember, the PO is accountable to the other stakeholders as their representative to ensure they get what they want.

The demo is how the PO and team make sure the stake holdres are happy with what we’re delivering. The demo is a powerful ceremony for Scrum teams. The demo builds trust between the team and the stakeholders. Which comes in a few ways in this context. First and foremost, this is a forum for direct conversation with the stakeholders. This group has a vested interest in the success of the team and the product. It’s a great opportunity for the team to receive direct feedback.

The stakeholders get to see for themselves the work being done on their behalf. They provide feedback to the team, both praise for what’s been done, and suggestions for changes also known as new stories. The PO will capture these and after the demo set about adding details and vetting the new stories for their backlog. Sometimes, the stakeholders will see something that’s demoed and decide they don’t want it after all. That’s completely okay. Beter to hear it now and catch it right away.

The team can then pivot to what the stakeholders do want. Another advantage of the demo is to let the stakeholders know who is working on their project. They get to see for themselves the skills and dedication each team ember brings to every sprint. This is an opportunity to build relationships between the team and stakeholders. This visibility gives the stakeholdres a broader more balanced perspective on what it takes to crate their product.

Additionally the stakeholders will see the teams willingness to receive feedback and adapt their changing needs. Finally the demo shows overall progress towards the final goal. Your PO will keep your product road map and releas plan up to date after ever yspring. This is where you show them to your stakeholdres. Your team is bringing the stakeholdres along on the journey with them. The stakeholders can also provide feedback on the timing and contents of each planned release.

Again, another trust and relationship building exercise. You might be wondering if you have the demo at the end of every psring. The answer is no. you don’t. but you need to demo as often as possible. It only helps you to build trust with your stakeholders and prove the progress being made for them. Your product will be more robust and widely accepted when you receive regular feedback to help perfect it. Once of the best things about Scrum is its transparency in aking everything visible to everyone.

Be sure you demo regularly to keep your product and stakeholders in close alignment.

ASSES THE TEAM

For a scrum team, 100% of every spring is focused on the needs of the stakeholders and the end users, but the principles behind agile say that teams need to reflect regularly on how to be more effective and adjust their behaviours according. In scru, this principle has taken shape as the team retrospective, or a retro for short. This is the one time every sprint when the focus is not on the product, it’s on the team itself.

Since the focus is on the team and team processes, the scrum master facilitates this ceremony. In order to have a successful retro, you must have a safe environment, as a result this is a closed door session. The team must know that only dedicated team members will be present and that the team norms will be observed. This sense of safety is essential to ensuring the open dialogue that’s needed to really assess honestly the way the team is working together.

Usually the agenda for the retro is simple. It consists of three questions. What worked well? What did not work well? And what will we improve? While everything on this list is important be sure to start with the team successes first. I know this sounds backwards. We’re here to get better, why focus on what we already do well? But in order to help the team stay away from responding defensively you need to appreciate them.

Fill their emotional bank accounts before you start sharing the improvement areas. When you focus on question one, pay attention to everything the team did well that’ within their control. This includes examples of great collaboration both inside and outside the team. Recognize things the team has done to help each other out. Be sure to focus on the team itself not the stories they’ve completed. By focusing on success, the team creates a positive behavior loop.

They’ll want to become better so they have more to celebrate the next time. When moving to question two, what didn’t go well, be sure to focus on things that you can change not on outside groups or tools that are beyond your control. For example, perhaps your team needs to use a specific testing tool and the tool is slow. Your improvement area wouldn’t be to fix the tool, it would be to find a better way to use the tool more effectively.

After you have a good list of improvement areas, you can move to question three, what do we want to focus on? All of the items identified for improvement, more to an improvement backlog. It’s not feasible to work on everything at once, so the teams need to decide which one or two items to focus on in the next sprint. During the next spring, the scrum master will hold the team accountale to the improvement commitments as well as their story commitments in this way the scrum master hekps protect the teams commitments to themselves too.

All the other tools scrum teams use are about the product. The retro is powerful because its about the team itself. Remember scrum is a framework that only be successful when the team is healthy and focusing on improving themselves as well as their product.