AGL Academy

A community effort by Agile government professionals to help educate and empower those who seek to implement Agile processes into their own agencies.

Powered by Agile Government Leadership

By bringing applied Agile practices to government, we want to redefine the culture of local, state and federal public sector service delivery across all aspects of government. We will work with Agile professionals and organizations to support their work in getting Agile infused into government processes. We will foster a spirit of openness and mentor those new to Agile so that they have the necessary practical advice, resources, tools and community support for successful deployment. Through Agile Government Leadership, we will create a responsive, engaged government that more efficiently and effectively serves its citizens.

Connect with AGL

- Website
- Subscribe
- LinkedIn
- Twitter

Agile for the Government Project Manager

Welcome.

This material is designed for the Project or Program Manager (PM) in government who wants to lead their team to start practicing Agile.

Here you will find a step-by-step course to get you and your team started. It includes readings, videos, general introductory info, advice specific to government settings, and workshops you need to run in order to start providing the benefits of Agile development to your stakeholders.

You have several options for completing this course:

- Use the Trello software (explained in the next section) so that your team can get experience with Agile processes while learning
- Use this document (contains links to outside reading material and resources)
- Use our website (http://www.agilegovleaders.org/academy)

You will work through a series of tasks each week. These start out as simple readings for you to discuss with colleagues, then progress to workshops you must perform that model the Agile process on a small scale to give you hands-on experience. With a little effort and cooperation from Project Managers on your team, in a few weeks you will be ready to begin a serious Agile project.

Before you begin, please note:

- This course is intended for a government Program Manager, or Project Manager, who is in charge of a software project. It is not aimed at executives that have many Program Managers reporting to them.

- Although there are many Agile Methodologies, this document is about Scrum, a common and standard Agile practice that is widely used.

- As a government professional you may have to deal with procurement and other issues. Although we discuss these challenges, this curriculum teaches you Agile project management, not Agile procurement. We assume that your development team is willing to work in an Agile, iterative way, whether they are your employees or your contractors.

- This is an introductory course -- there are many other important topics in Agile development which we will mention at the end when we point you in the direction of the "Next Level".

We at Agile Government Leadership hope this empowers you and your team to wow your citizens with efficient delivery of phenomenal digital services. We welcome and appreciate feedback from you on what could allow us to iteratively improve this curriculum for the next PMs who take on this challenge.

Contributors

This course was designed by Agile government professionals with combined experience at the federal, state, and local levels, in addition to the private sector. Contributors to this course are members of the AGL Working Group: Elizabeth Raley, Bill Haight, Tim Nolan, Son Tran, Robert Read, and Doug Birgfeld.

Trello - Your Tool For Agile Learning

Included in this course is the option to use Trello to guide your team through the learning process. This will serve a dual purpose:

- It gives your team a chance to try out Trello, which is a free and easy-to-use Agile tool. You may find yourself using it for many future projects!

- It formats the course in a project-focused way that gives clarity to the process. We strongly recommend that you follow the course using the Trello board, but if you decide not to, the material is available on our website and in this document.

Why Trello?

As you will learn, the "user story" is the fundamental unit of work in Agile methodologies. Tracking stories on a storyboard allows for transparent communication among the whole team. In this curriculum, the stories (or tasks) are not the production of software but the gaining of understanding. As you work through the tasks, you can use the Agile Team Curriculum Board (below) to track your own team's progress.

In a Scrum process, a story is not considered "Done" until it has been demoed and has passed a quality assurance process. Our quality assurance for these curriculum tasks is simple: you have to have a conversation with a colleague about the content of each reading task, video, or workshop on the Trello board.

Ready to get started with Agile learning?

If you and your team already use Trello, you can simply access the board below and begin: https://trello.com/b/m7LowhmZ/template-agile-team-curriculum

If you're not using Trello already, you can sign up here: https://trello.com/signup

You can watch a quick video demo here: Trello for the Agile Government Curriculum

Lesson 1: Introduction To Agile

GOAL: By the end of this lesson and after completing all the items below, your team should have a shared terminology and understanding of Agile and Scrum, along with the motivations for using them in government projects.

What Is Agile?

Agile is an iterative and incremental method of managing and developing projects with a team. It focuses on customer collaboration, responding to change, and frequent releases of working software.

The Agile Manifesto is a single, simple, brilliant sentence that you should refer to often:

> "Through this work we have come to value:
>
> - Individuals and interactions over processes and tools
> - Working software over comprehensive documentation
> - Customer collaboration over contract negotiation
> - Responding to change over following a plan
>
> That is, while there is value in the items on
> the right, we value the items on the left more."

Let's take the manifesto and put it into the context of real work, real customers, and the real world. Agile takes the best of the world's history of getting things done and rolls it into a single framework. For instance, we know that people work better in teams; we know that trusting relationships have better results than legal documents; we know that predicting the future is super hard; we know that the proof is in the pudding. We also know that 3 things 100% done are more useful than 10 things 75% done. And finally we all know the phrases: "one day at a time", "one game at a time", and "keep your eye on the ball".

Agile takes these intuitive truths and brings them into the world of development, software and beyond. Agile is not a prescriptive dogma. It is a set of activities and methods that can be used individually to make things better, and collectively to make things great.

Agile takes practice and disciple. But just like that play you were in, or the concert you practiced for, you will be happy with the ovation at the end. It's worth it.

Why Agile in Government?

The Agile Manifesto follows 12 Principles that focus on value, collaboration, motivated people, change, rapid delivery, and simplicity. These same principles apply to government regardless of whether your agency is Federal, State or Local. The principles offer a pragmatic approach for working with customers, citizens, employees and each other.

An Agile government provides excellent services and value because managers trust their teams to do the work and to collaborate directly with customers. Instead of micromanaging and frustration, there is daily communication and collaboration. This culture of transparency rubs off on the customers within an Agile government -- and soon, the culture of the agency begins to shift.

With Agile processes paving the way for delighted customers, a government's focus can be on providing genuine happiness instead of settling for "good enough" or shrugging off another failed IT project. Eventually, a government practicing Agile will adopt a permanent Agile mindset. The 12 Principles behind the Agile Manifesto will become self-evident instead of being something to strive for.

The easiest way to identify and explain what Agile offers to your agency is to replace it with "flexible". We want our government to be flexible to meet the changing needs of our customers and citizens. We want our government to be flexible to produce the greatest value within the allotted time frame and budget.

Being Agile (flexible, transparent, efficient) should be the goal of every government agency.

Reading List

Study each of these resources and discuss what you're learning with a colleague.

Agile Government Handbook

AGL has compiled a handbook that outlines Agile in the government space.
Study Time: 90 minutes.

Government Challenges to Using Agile

This article discusses government-specific challenges that you may face in moving your team to Agile methods.
Study Time: 10 minutes.

The Scrum Guide

Scrum is a framework for practicing Agile. This guide lays out the Scrum roadmap and will serve in a full understanding of how to practice Scrum.
Study Time: 30 minutes.

Agile Terms

New vocabulary will come up while talking about Agile and Scrum. This is a cheat sheet to ensure the team has a shared understanding of what is being discussed.
Study Time: 20 minutes.

Video List

Watch these videos to gain a deeper understanding of the how-to behind Agile and scrum. Continue to discuss your findings with team members to ensure that you understand the concepts.

Agile 101 Case Study Discussion

Study Time: 1 hour

How to Run an Agile Project

Study Time: 1 hour

Scrum Training Series

(Individual videos below) Study Time: 3 hours

Scrum is a framework for Agile that addresses many of the issues that we face daily in website development: changing business needs, changing technology, high value needs and more. It provides us with a roadmap to follow in order to run an Agile project.

- Intro to Scrum
- Backlog Refinement
- Sprint Planning Meeting
- Daily Scrum Meeting
- Sprint Review Meeting
- Sprint Retrospective

Agile Assessment

Determine your team's current Agile capabilities using this Agile Assessment. This will help you see how Agile will be supported or challenged by your agency's leadership, culture, policies, etc.

CONGRATULATIONS, YOU'VE COMPLETED LESSON 1!

Lesson 2: Writing And Estimating User Stories

GOAL: By the end of this lesson and after completing all the workshops, your team should be able to write user stories and work together to put estimates on them.

Workshop: How To Write User Stories

- **Estimated Time:** 60 minutes for workshop for whole team, 2 hours prep time for Agile Project Manager
- **Materials Needed:** Index cards or Post-It notes, pens
- **Outcome:** Everyone knows the standard story format, knows some of the INVEST acronym, and can a write a user story.
- **Test**: Can you storify a simple Task?

Introduction

One of the most important aspects of moving to Agile is understanding "stories". The story represents the fundamental unit of communication and tracking progress. It takes practice to write good stories, and a workshop allows you this practice.

A User Story must provide value to some user. An Agile process is driven by the completion of stories, each of which provides tangible, demonstrable value to the user. A sprint consists of a set of conscientiously prioritized stories. Many people use the INVEST acronym to help understand what makes an effective story:

- Independent
- Negotiable
- Valuable
- Estimatable
- Small
- Testable

Experience will show that it's best to use a format for each story that identifies who the user is, what they need, and for what purpose. Such stories are written in this format:

"As a ___, I need a ___ in order to ___".

Example: "As a jazz fan, I need a tuning knob in order to find a jazz station on this radio."

Keys to a Valuable Story:

- Owners must have courage to ask for what they really want.
- A story must have value to someone. It must make the product better in some way.
- "Clean up the bugs we introduced in the last sprint" is NOT a story because it does not add anything to the product.

Keys to Successful Negotiation:

- Engineers and Product Owners in the same room
- Try to find a cheap solution that gets 90% of the value done
- Can ONLY be done through intimate conversation

Running The Workshop

Believe it or not, one of the critical technologies for a successful workshop is either index cards (3x5 or 4x6) or Post-It notes. Cards are better for a team smaller than 7 which can gather around a table, and Post-It notes are better for larger teams.

Part of the reason you use paper technology is so you can easily move stories around, reorganize and reprioritize them, and throw them away when done. The small size of the cards and notes ensures that you will not write too much into each story.

A story is a promise to have a conversation later between the end-user and developers. Your goal in writing stories is not to work out details, but to discover the most important goals for your project and to organize a project into discrete, testable chunks.

Potential project goals that will help you practice story-writing:

- Clean out the garage
- Develop a static website that informs people about dietary impact on breast cancer
- Develop an app for playing "Conway's Game of Life"
- Develop an app for playing tic-tac-toe
- Write a how-to guide for planting a garden specific to your locality

Choose a project goal for the workshop from the list above. If you are running these workshops with your development team it may serve you throughout this curriculum to have a consistent project for each workshop -- so choosing a site or app that you will eventually build in a later workshop is a good idea.

Make sure you have at least one end-user present, and at least one developer. One person must be designated the Product Owner -- their job is to know what the end users want and to prioritize the work. Another person must be the Agile Project Manager -- their job is to coach the team throughout the workshop and to keep the workshop on track.

You can also practice without a technologist. For this, you might want to choose the "Clean out the garage" project goal. Anyone can act as both the end-user (who needs work done) and the developer (who does the work) for this project. It makes a good one-person practice project.

It is absolutely essential that everyone has pens and cards. Sharpies are best, because they can be read from far away.

The phases of the 50-minute workshop are:

1. Review of the user story format (1 minute)
2. Review of the INVEST mnemonic (4 minutes)
3. Open brainstorming -- write at least 3 stories (5 minutes)
4. Backlog grouping exercise (5 minutes)
5. Backlog prioritization exercise (5 minutes)
6. Quick estimation (60 seconds per story)
7. Reprioritization (3 minutes)
8. Format checking (2 minutes)
9. INVEST checking (5 minutes)
10. Discussion of the workshop (5 minutes)

If time allows, you may repeat this workshop to create an even better 2-iteration workshop. You can go through the whole process again and build out more user stories.

Note that the Agile Project Manager should encourage changes to the stories and the creation of new stories at any time during the workshop, and during any phase. The goal is to get a creative set of stories. Generally, the process of story writing is a process of discovering things you didn't realize you needed or wanted, and sometimes it is the discovery of surprisingly simple solutions.

In greater detail, here is what happens in each phase:

Review of the user story format and INVEST mnemonic (5 minutes)

The Agile Project Manager is responsible for keeping fairly strict time. Agile teams must accept the discipline of time-limited iterations, on both large and small scales. In this phase you present

the structure of the workshop, the roles, the project goal, and a brief introduction to the story format and the INVEST acronym, which are ideally written on a flip-chart or somewhere where everyone can see them. You will not have time to fully explain this, but you will explain it more as the workshop progresses.

Open brainstorming - write at least 3 stories (5 minutes)

Everybody takes a sharpie and cards. Everybody writes stories for 5 minutes that they believe will advance the project goal. The Agile Project Manager should walk around answering questions during this time. People "publish" their stories on the wall or table as soon as they are written, but no criticism of stories is allowed during this phase. The developers participate in this phase just as the Product Owner and end-users do.

Backlog grouping exercise (5 minutes)

The cards are presented to the whole group. Stories that are duplicates or very closely overlapping are grouped together. Note that upon reading the stories of others in the group, anyone may write new stories on the spot (and in fact this should be encouraged during any phase). If a story is consensually deemed to be obsolete or going in the wrong direction, it can be thrown away or moved into a reserve pile. During this time stories can be rewritten or reworded to make them more Independent.

Backlog prioritization exercise (5 minutes)

The stories are force-ranked into order, with the highest priority at the top. There are no ties; a specific order must be chosen. Open discussion is allowed. In the end the Product Owner sets the actual priorities. His or her authority to do this is absolute.

Quick estimation (60 seconds per story) - as much as 15 minutes

In priority order, everyone produces a Small, Medium or Large estimate for the time to implement each story. Even the non-developers do this. We have a separate workshop for estimating, so this exercise is meant to practice evaluating if the story is estimable -- no need to spend too much time quantifying the story's size. The Agile Project Manager must keep the discussion to 60 seconds, which will be very hard. People will want to discuss the stories far more. Under no circumstances can you spend more than 120 seconds on a story. If 120 seconds

goes by, the development lead writes something down no matter what. If any stories seem too long or large, they can be quickly split into multiple smaller stories.

Reprioritization (3 minutes)

Given the estimates, the stories may be reprioritized or adjusted. This is usually an important learning phase. Usually, there are stories which are surprisingly easy or surprisingly hard, and, given their cost, the order in which they are to be accomplished clearly changes.

Format checking (2 minutes)

Confirm that each story is in the correct format: "As a ___, I need a ___ in order to ___".

INVEST checking (5 minutes)

Try to make sure that each story is testable by writing a description of the test on the back of the card. This should lead to a lively discussion between the developers and end-users.

Discussion of the workshop (5 minutes)

Spend 5 minutes quickly discussing not the stories, but the process of story writing. Talk about what worked and what didn't work. Make a list of action items that the team will do differently next time.

Things That Will Likely Happen

- Some people will have a hard time producing 3 stories.
- Some people may object to moving so quickly with so little explanation. Push through this. They will come around when they see the results.
- Some disagreement on product direction may be uncovered. The Agile Project Manager should respect this. The Product Owner has the final authority on prioritization. Ideally, the disagreement will result in a creative compromise which synthesizes the results.
- It is possible that this will all fall apart and you will have to try again. Be willing, it's worth it.

Workshop: Agile Estimating

- **Estimated Time:** 90 minutes for workshop for whole team, 2 hours prep time for Agile Project Manager
- **Materials Needed:** The user story cards from the previous exercise, and some type of Planning Poker tool.
- **Outcome:** Everyone understands basic Planning Poker and how to come to a consensus for a high level estimate.
- **Test**: Can you estimate a story quickly as a group? Can you explain why you've estimated it that way?

Introduction

It is difficult to estimate how long it will take for work to be done on a project, particularly within a Sprint or time box. An estimate is a guess. Your aim is to make an accurate guess. You must think abstractly to determine how you will approach your work. Think in terms of relative sizing instead of actual units of time.

To illustrate this, consider the Empire State Building. It's roughly 100 stories tall. Let's say you work in an office building that is 10 stories. It is daunting to compare how much work is required to build the Empire State Building versus your 10-story office building. However, if we look at this abstractly, you can say that the Empire State Building is ten times (10x) more difficult to build than a 10-story building. Trying to determine actual time would take far too much planning and may simply be impossible.

Relative estimating frees the team to focus more on how to solve the problem and less on actual duration. Duration is important to a degree, but the team should spend more time considering work items in relation to each other, especially if the story is complex or has unknowns. Duration can be determined as more of an afterthought.

To practice relative estimating, look at the stories in your backlog. Find a simple story and compare it to a more difficult one. Is it twice as hard? Three times? Ten times?!? You've done it. You have performed a relative estimate.

There are many Agile estimating techniques that don't require putting an actual hours estimate on a story. One way is to simply compare one backlog item to another. You can separate them into Small, Medium and Large stories or backlog items. The team can use this technique to evaluate a large backlog in a short period of time. You can expand your estimating to mimic T-shirt sizes -- XS, S, M, L, XL, 2X, 3X, etc. This estimating style is comparatively easy but does not allow you to quantify work to be done. Assigning a relative number to our estimates will be useful for computing velocity (how much work a particular team can accomplish in a Sprint).

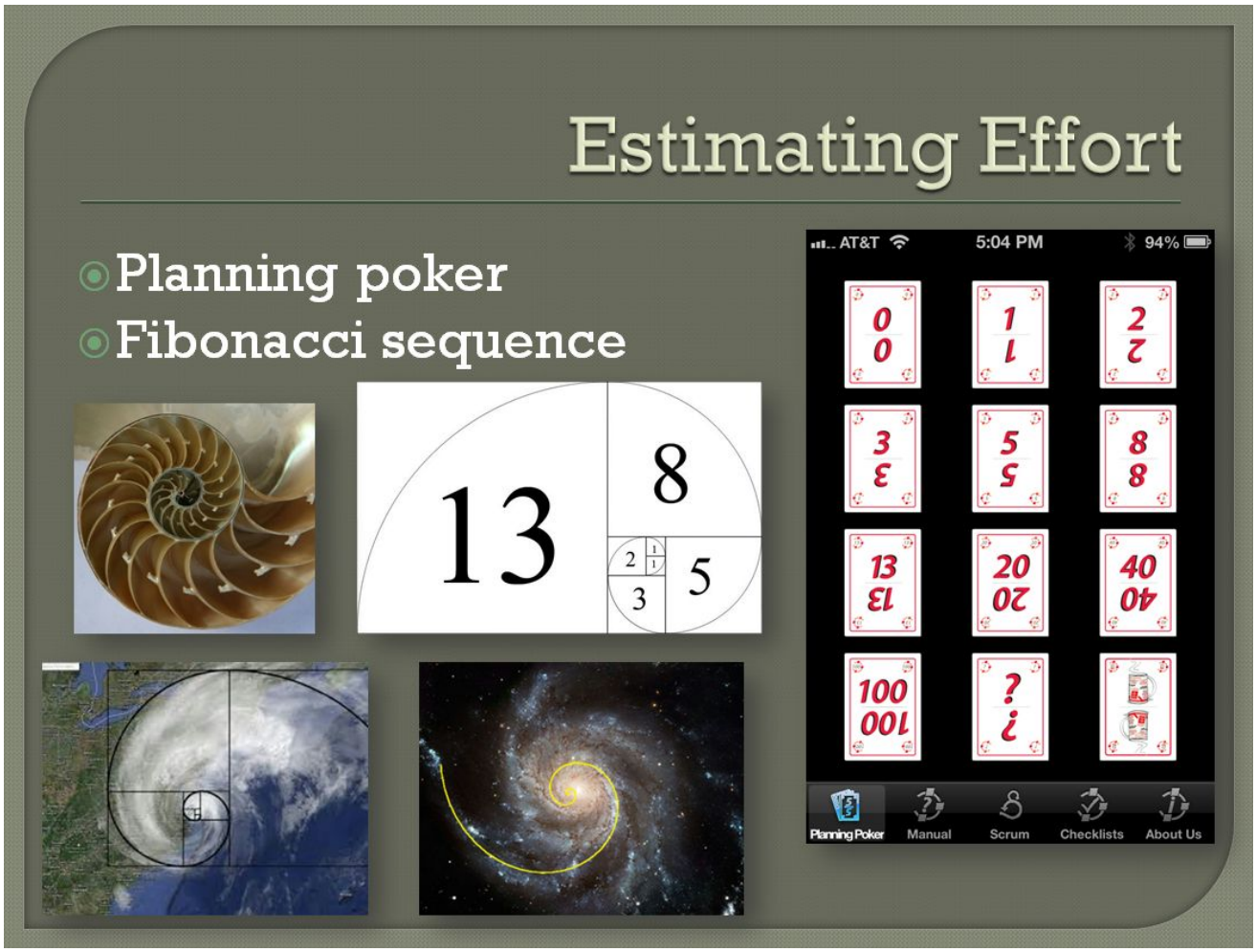
Some teams have very complex backlog items. Complex does not always mean big. To visualize this, consider the attempt to put a collar on a horse, dog, cat and canary. The horse and dog are the bigger animals, but the task of putting a collar on them is fairly simple. Putting a collar on a cat becomes more complex. Putting a collar on a canary is darn near impossible even though the animal is the smallest. Perspective is key in this estimating technique.

Planning Poker is a very popular method for estimating work items. Start your Sprint planning session with a deck of Planning Poker cards, download a Planning Poker app, or use your fingers. Planning Poker is based on the Fibonacci Sequence or golden spiral -- 1,2,3,5,8,13, etc. -- where we take the sum of the two values to the left, to get the new value (0+1=1, 1+1=2, 1+2=3, 2+3=5). Many people appreciate this technique because it mimics nature -- Planning in Nature.

Planning Poker takes time to master. Eventually, you may find that using this method during your planning sessions will be similar to a team building exercise. The team will become familiar with others' voting tendencies. They may even tease each other and have fun with the activity. Ultimately, the team will be able to estimate quickly and effectively.

Planning Poker is quantifiable. The planning estimates can be used to determine velocity for a Sprint, for instance. Velocity is the total estimation points (i.e., story points) completed in a sprint. (Each story is assigned a point value based on its level of difficulty). You can average your velocity over several sprints to get an accurate glimpse of your team's capacity. If your team is averaging 40 points per sprint then the next sprint should be around 40 points too, not much higher or lower.

Pick the estimating style that works best for your team. Feel free to try various methods or come up with one yourself. For this workshop we'll use Planning Poker so that your team can try it on for size.



Running The Workshop

Before the workshop begins, decide if you will use the Fibonacci sequence or T-shirt sizes for Planning Poker. Choose your Planning Poker tool. You'll work with the user stories that were created in the previous workshop.

Since this is the same group that participated in the User Story workshop you will have the same people in the same roles of Product Owner, Agile Project Manager, End User and Developers. The Agile Project Manager will continue to coach the team throughout the workshop and keep the workshop on track.

The phases of the workshop are:

1. Review Estimating (15 minutes)
2. Introduce Planning Poker (10 minutes)
3. Introduce the User Story (1 minute, repeat)
4. Discuss the Work (1 minute, repeat)
5. Reveal the Estimates (1 minute, repeat)
6. Discuss the Estimates (1 minute, repeat)
7. Estimate Again (1 minute, repeat)
8. Agree and Move On (1 minute, repeat)

As time allows, you will repeat these steps until you have enough stories estimated that would fulfill your sprint.

The Agile Project Manager should encourage changes to the stories and new stories to be written at any time during the workshop. As estimating conversations occur, the team may realize that assumptions were made that aren't represented in a user story. The goal is to practice having conversations around estimating and to come to an agreement on how much effort a story would take.

In greater detail, here is what happens in each phase:

Review Estimating (15 minutes)

The Agile Project Manager will review the ideas about estimating, including the Keys to Estimation:

- "A story is a promise to have a conversation" -- it does not specify all details that are needed to implement the story.
- An estimation is not a promise to complete the work in a certain amount of time.
- After this workshop, if you take more than two minutes estimating a story, you are taking too long.
- A story that takes more than one sprint is too long and must be broken up into smaller stories -- you won't know this until the estimation phase.
- Do estimation in a short meeting (90 minutes at most) with engineers AND story authors in the same room.
- Avoid rating your abstract story units to calendar time until you can measure from the velocity of a burndown chart for at least three sprints.

Introduce Planning Poker (10 minutes)

Planning poker is a tool that we'll use to do consensus based estimates. We've decided to either use the Fibonacci sequence or t-shirt sizes as our unit of measure.

Introduce the User Story (1 minute, repeat on next cycle)

The Product Owner (or another team member) introduces the first story to be estimated and describes the work item to be estimated covering the presumed duration, complexity, and any unknowns that may exist.

Discuss the Work (1 minute, repeat on next cycle)

The entire team participates in a discussion on how to get the work done. They will talk about what they will need to do to get the work done and ask questions to ensure they understand the user story fully. Once the discussion is over and all the questions have been asked, then it's time to focus on estimating again.

Reveal the Estimates (1 minute, repeat on next cycle)

Each team member will select the relative number that best represents the size of the work in their opinion. The entire team reveals their numbers simultaneously so that opinions are not swayed by early revealers. If everyone chooses the same number then that will be the estimate.

Discuss the Estimates (1 minute, repeat on next cycle)

If a broad range of numbers is revealed then you ask the outliers to explain their answer. The discussion continues until everyone has asked their questions.

Estimate Again (1 minute, repeat on next cycle)

Again, the entire team reveals their numbers simultaneously so that opinions are not swayed by early revealers. In most cases the numbers disclosed get closer to each other.

Agree and Move On (1 minute, repeat on next cycle)

The estimate exercise will conclude once your team reaches a consensus. You continue this process until you feel you've estimated enough work items to start working on your sprint (or run out of time).

Things That Will Likely Happen

- Some people will have a hard time estimating. We sometimes use the unpleasant "gun to your head" metaphor to insist that people MUST give an estimate.
- Some stories will require more conversation than others. Do your best to stick to the timebox and remember the keys to estimation above.
- Some disagreement on estimates will occur. The Agile Project Manager should respect this. The Development team has the final authority on the estimate. A Product Owner cannot say how long something should take but they can participate in estimating as an exercise. Ideally, the disagreement will result in a creative compromise which synthesizes the results.
- It is possible that this will all fall apart and you will have to try again. Be willing, it's worth it.

CONGRATULATIONS, YOU'VE COMPLETED LESSON 2!

Lesson 3: Running a 3-Sprint Workshop and Producing a Backlog

GOAL: By the end of this lesson and after completing the two workshops below, your team should know how to run a sprint and create a product backlog. By the end of this week your team will be prepared to start a real Agile project!

Workshop: Running 3 Sprints In One Day

- **Estimated Time:** 7 Hours for workshop for whole team, 14 hours prep time for Agile Project Manager.
- **Materials Needed:** a work room, workstations, wifi, demo mechanism, a storyboard, post-its or index cards, pens, a timer (details below for all these materials)
- **Outcome:** A demo-able, functional prototype, and a shared confidence in the process.
- **Test:** Did you get 3 sprints done following all ceremonies? Do you have a functional prototype of something? Do you feel confident in attempting 2-week sprints now that you have done three 2-hour sprints?

Introduction

It's time to plan and run a successful one-day workshop in which your team will perform three sprints. This is a model of the sprint structure that will be used for your future projects. It allows you to practice much of the Scrum ceremony within very compressed 1- or 2- hour sprints. As the Agile Project Manager, your job as a leader is to make sure that the Workshop happens and that you have a customer representative present. You may be the only person empowered to do this.

You may find it helpful to first read "The Story Of An Agile Workshop" and "How A Two-Day Sprint Moved An Agency 20 Years Forward".

In order to practice running an Agile project you are going to run a project on a very, very small scale -- but it will have all the features of a bigger project. It is critical that you execute all the features and follow the format. Perfect practice makes perfect performance. You and your team will compress 3 sprints -- each of which would normally take 2 weeks -- into a single one-day workshop.

In the next workshop you will be building a backlog of the stories that really matter to you. However, for the purpose of this workshop you may want to choose something in which your team is less emotionally invested.

A reasonable topic for this 3-sprint project is to produce a resource website about an important subject. Far-reaching and well-known topics -- like breast cancer or global warming -- are rich, valuable, and provide ample opportunity for development. Almost all developers and/or development teams should be able to produce a website quickly. This project cycle allows for 4 hours of development total, so it will need to be a site that can be done in that amount of time.

As the Agile Project Manager, your duty is to make sure that the process is followed and to make sure that time is strictly adhered to. In this kind of workshop you will be under tremendous pressure to extend a sprint deadline. You must NOT do this. When the time comes for the demo, the demo happens. If the developers have to say "We implemented zero stories in this sprint", then they say that in front of the whole workshop and the demo is over and you go on to the next phase.

Part of the beauty of this style of workshop is that it puts developers in close communication with end-users. This communication should be fostered. Usually, this is where the magic happens.

Nevertheless, there are some roles that must be respected. The workshop is formally divided into the development team and the "customer-facing" team. The customer-facing team includes the Product Owner, writers, and end users.

How the team members respect each other:

- The development team does not question the priorities set by the customer-facing team.
- The customer-facing team does not question the estimates or mechanisms of the development team.

In other words, the customer-facing team may not say “I think that should be easier than X.” The development team may not say “I think this story is more important to our users.”

The customer-facing team has absolute control of how stories are prioritized and whether a story passes or not at the time of demo. The development team has absolute control of the estimates and the technologies used.

Note, however, that story writing is always a shared responsibility. The wording of the stories must always be produced in close collaboration.

Duties Of The Agile Project Manager

- You must arrange a room large enough for all participants and in which the developers can comfortably work. This may require you to provide workstations, electricity, WIFI, or other telecommunications. You cannot have two rooms. It would be better to do it outside on the lawn or in the parking lot than in two rooms.
- A technical mechanism must be provided for everyone to see the demo. Depending on the size of the team and the nature of the development, this might be just crowding around a laptop, using a projector, or having a separate machine dedicated to the demo. It could conceivably be done entirely by web conference, if that is the nature of your application.
- You must keep the storyboard. Ideally this is a whiteboard broken into lanes showing the status of the stories. It is your duty to make sure the stories move correctly forward (and sometimes backward!) on the storyboard so that it always reflects the status of all the stories.
- You must keep the workshop on schedule.

- You must defend the rights of the teams, while fostering communication as much as possible. If a developer asks you a question about a story, you should get the customer-facing team to answer it. If a customer-facing person asks you a question about what is technically possible, you should refer this to the developers.
- You must insist on everyone being present for the end-of-sprint demos. The demos are the climax and goal of the entire workshop.

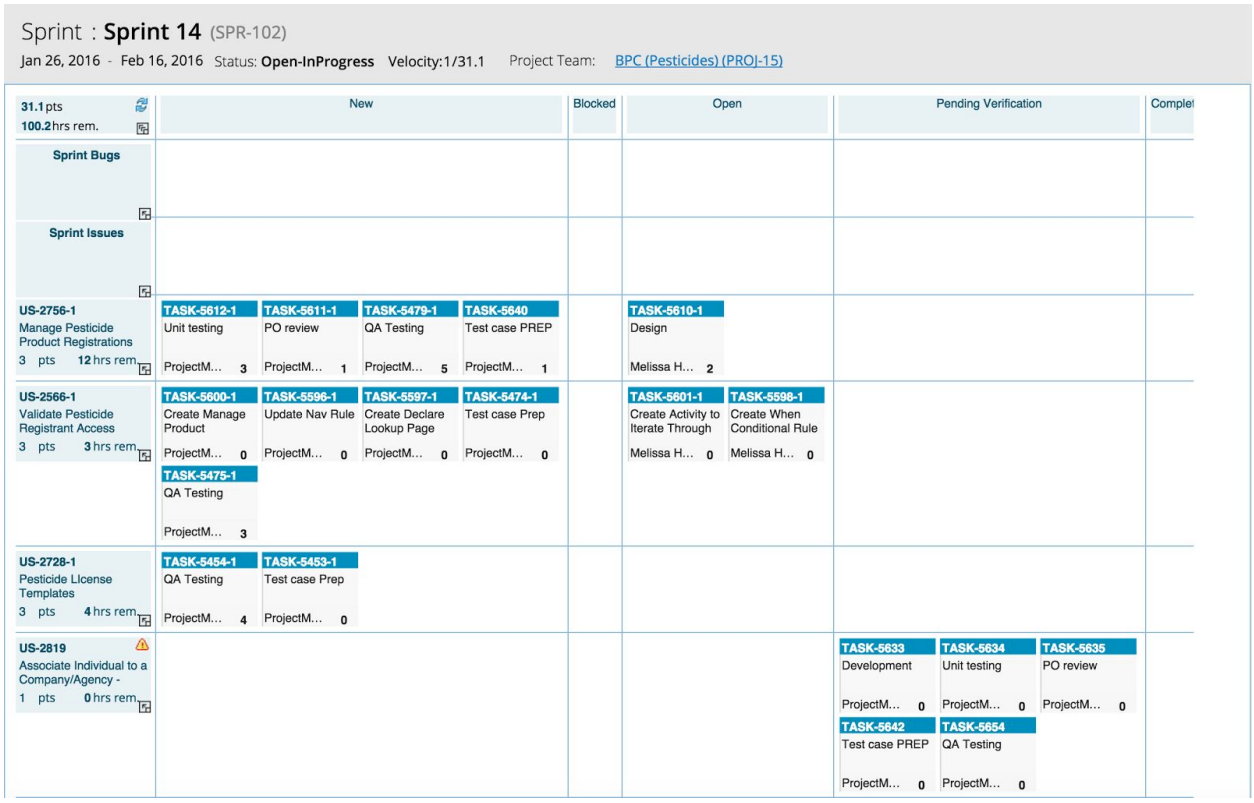
The Storyboard

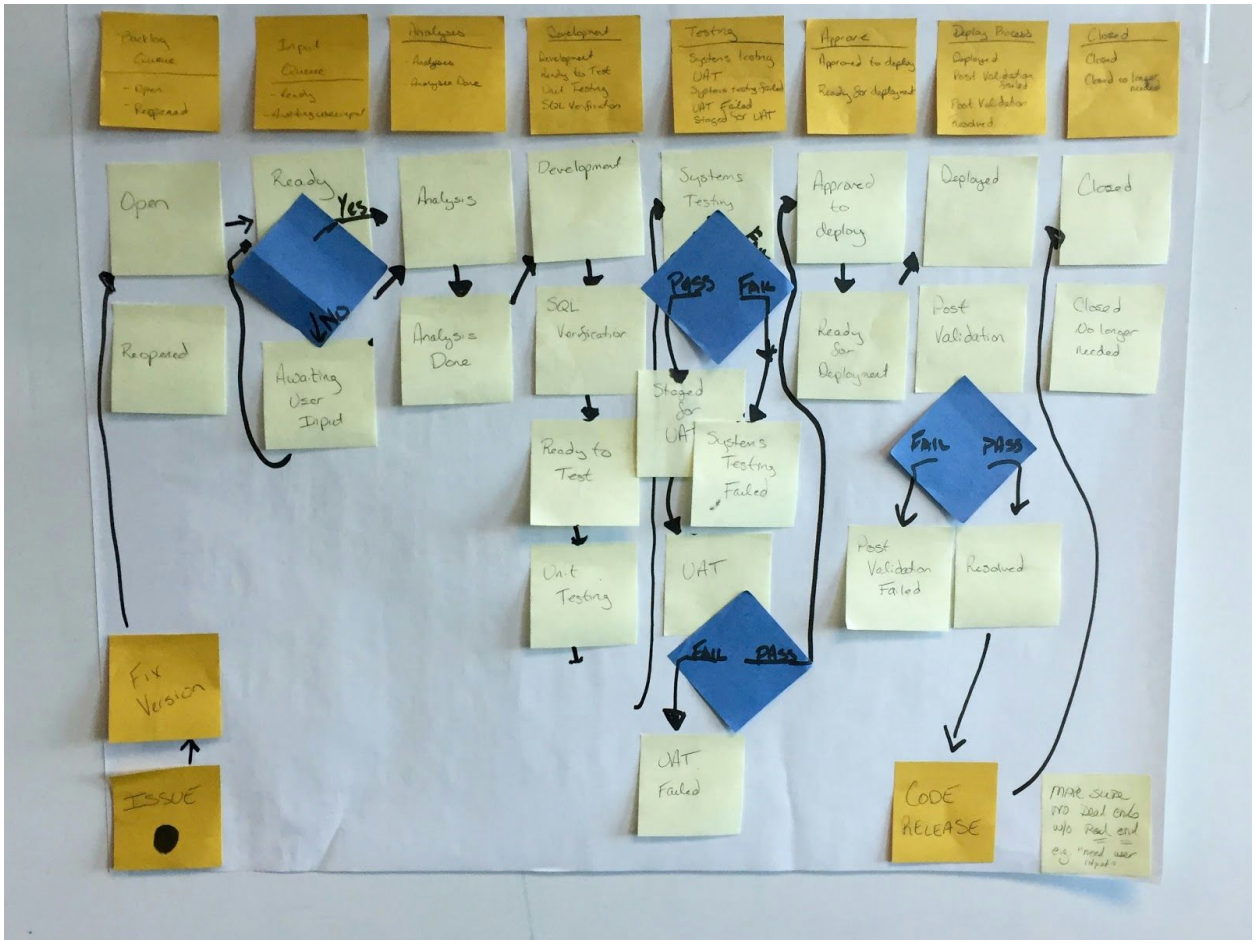
The Storyboard is a Big Visible Chart that is really the heart of the workshop. It must be where everyone can see it. A whiteboard with sticky notes works fantastically well for this.

The Storyboard has lanes. Each lane has a list of stories that are ready to be worked on by members of the team. A story is moved from one lane to another, usually but not always forward, by the people working on the story. The Storyboard looks something like this:

Backlog	In Sprint	In Dev	In QA	Done

Here are some actual working boards:


Sprint : Sprint 14 (SPR-102)
Jan 26, 2016 - Feb 16, 2016 Status: Open-InProgress Velocity:1/31.1 Project Team: BPC (Pesticides) (PROJ-15)
31.1 pts
100.2 hrs rem.
New
Blocked
Open
Pending Verification
Complet
Sprint Bugs
Sprint Issues
US-2756-1
Manage Pesticide Product Registrations
3 pts 12 hrs rem
TASK-5612-1
Unit testing
ProjectM... 3
TASK-5611-1
PO review
ProjectM... 1
TASK-5479-1
QA Testing
ProjectM... 5
TASK-5640
Test case PREP
ProjectM... 1
TASK-5610-1
Design
Melissa H... 2
US-2566-1
Validate Pesticide Registrant Access
3 pts 3 hrs rem
TASK-5600-1
Create Manage Product
ProjectM... 0
TASK-5596-1
Update Nav Rule
ProjectM... 0
TASK-5597-1
Create Declare Lookup Page
ProjectM... 0
TASK-5474-1
Test case Prep
ProjectM... 0
TASK-5601-1
Create Activity to Iterate Through
Melissa H... 0
TASK-5598-1
Create When Conditional Rule
Melissa H... 0
TASK-5475-1
QA Testing
ProjectM... 3
US-2728-1
Pesticide LIcense Templates
3 pts 4 hrs rem
TASK-5454-1
QA Testing
ProjectM... 4
TASK-5453-1
Test case Prep
ProjectM... 0
US-2819
Associate Individual to a Company/Agency -
1 pts 0 hrs rem
TASK-5633
Development
ProjectM... 0
TASK-5634
Unit testing
ProjectM... 0
TASK-5635
PO review
ProjectM... 0
TASK-5642
Test case PREP
ProjectM... 0
TASK-5654
QA Testing
ProjectM... 0



As you can see, there can be variations in how these boards are done, depending on what works best for the team. But they have much in common.

All storyboards must:

- Be big, visible, and clear
- Feature the highest priority stories at the top of the list
- Contain sections for Backlog, In Sprint, In Progress, Test, and Done (expressed in a way that makes sense to the team)

Remember to always keep the highest priority stories, visually, on the top of the list in each lane. The developers need to quickly reference the order in which they should tackle each story. Also, use language that the team understands most easily. If your team spoke Russian you

wouldn't do a board in Greek, so if the local culture needs to call "QA" something like "Pending Verification" -- that's fine.

The Product Owner can put a story into the backlog at any time. The PO and others on the customer-facing team should be constantly "grooming" the backlog for stories ready to be placed on the board, based on changing business needs or because they have been able to complete a story with enough detail to be worked on.

Stories move from "Backlog" to "In Sprint" at the beginning of the sprint based on priority (and the estimated available capacity of the team to fully complete the stories within the sprint's timebox). There should be about 2 sprints' worth of stories prepared in the backlog before the first sprint begins, along with some additional "stretch stories" that are ready to go, in case the team is working faster than predicted. Approximately 50% more story value than the minutes the team really has should be put "In Sprint".

When developers begin working on a story, place the story "In Dev". When a developer has finished coding a story, he or she should show it to a customer-facing person, who will place the story "In QA" if they agree the story is done.

At the demo, each Story which is "In QA" is demoed. If there is any question, do a simple voice vote of the customer-facing team only, and if they believe the story has passed, move it into the "Done" column. If not, move it back into the "In Dev" column.

Story Value And Points

Everything we do in life takes some effort, and some time. Your morning routine, for instance, is a set of stories that complete the goal of getting to work on time. By now you have that routine down pat, presumably. Let's break it down, though, for this example:

To get to work I need to complete these stories within the time box:

- Snooze a bit longer
- Pick out clothes
- Shower and Groom
- Dress
- Eat
- Drive to Work
- Park and Walk

Each of these story titles, in Scrum, would also have an explanation of what criteria must be met in order to be considered "done". For example, "Dress": must be able to tolerate winter cold, must have matching socks, must be covered top and bottom, etc.

Each story requires some amount of time and some amount of effort. Some things are easy, but take a long time (like driving to work), while other things require more effort but are quick to do (like parking and walking). We can assign a value to each task that reflects its relative effort -- this value is expressed in "story points".

Here is the previous list, with story points:

- Snooze a bit longer: 1pt
- Pick out clothes: 3pts
- Shower and Groom: 5pts
- Dress: 2pts
- Eat a cup of yogurt: 2pts
- Drive to work at rush hour on freeway: 8 pts
- Park and walk: 3pts

Assign points based on relative effort. In other words, do not count the minutes or the calories needed to do each story. Instead, consider the stories in relation to each other.

We know intuitively that pushing the snooze button is super quick and that driving to work is super long by comparison. So, the points reflect an order of magnitude. We learned in a previous workshop that one way to reflect orders of magnitude is a numeric pattern known as the Fibonacci sequence: 1,2,3,5,8,13, etc. -- where we take the sum of the two values to the left, to get the new value (0+1=1, 1+1=2, 1+2=3, 2+3=5). This is a very popular way of doing it, but there are all kinds of schemes that have the same effect.

If we add up all the points it will take 24 points of effort to get to work. My "get to work" Sprint has a 24 point value. We know the team (in this case, my car and I) have a capacity of 28 points, we know we can try to complete another story, like "stop and get gas" for my upcoming weekend trip. The "get gas" story is meant for the "go surfing this weekend" sprint, but if we have the time now, we might as well do it.

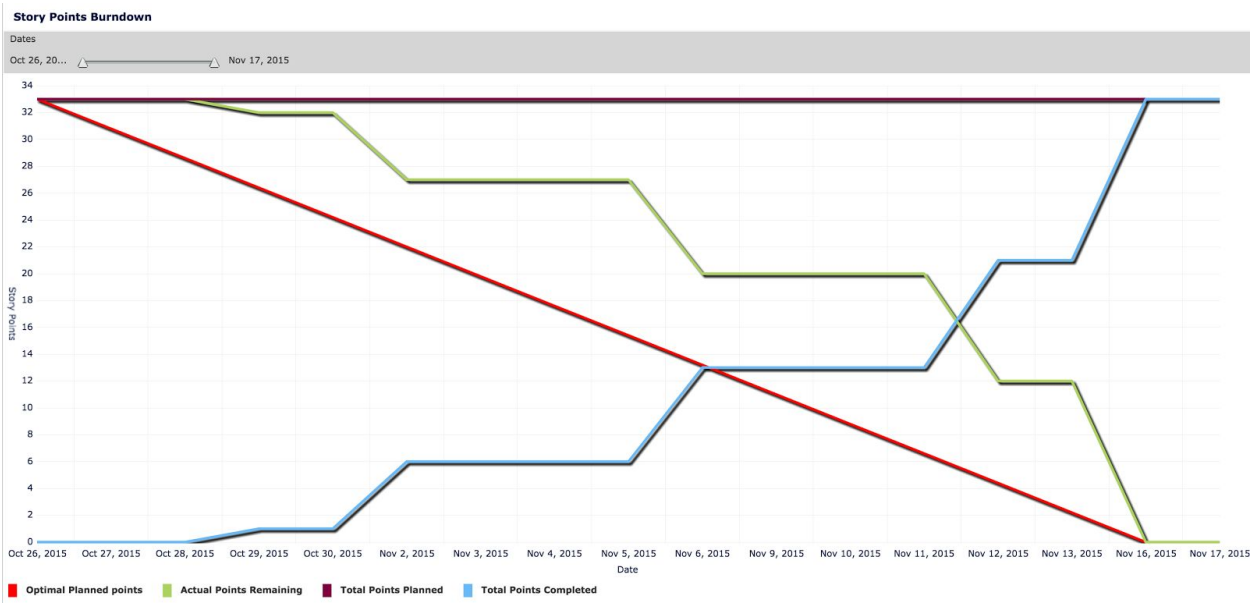
It could be that the "drive to work" sprint is accomplished more easily than predicted. In that case it's good to have extra stories ready to maintain our momentum and accomplish more. So we prepare 12 points of extra stories, just in case:

- Drop off package: 3pts
- Pick up snacks for weekend: 5pts
- Put in windshield washer fluid: 3pts
- Call Mom: 1pt

Now that you're familiar with story points, you can learn how to express the progress of stories using a Burndown Chart.

The Burndown Chart

The progression of stories and their assigned points, as they move incrementally from Backlog to Done, can be expressed in a burndown chart, shown here:



Burndown charts, like storyboards, can vary in appearance and have a number of tracked values, but they all have these things in common:

- Express a vector from the maximum anticipated story points to zero -- the red line. This vector is the perfect function of points completed or "burned" per day.
- Show a plot of of actual points complete for each day that passes. If you finish a 5 point story, 5 points are burned.
- Show the number of points planned. This number can go up or down if the team adds a "stretch story" or a story is removed because it's blocked by external circumstances
- Sometimes (as pictured above) there is a burn-up line, useful for predicting team patterns to find the "break even" point.

Every team has a maximum point value that they can take on. New teams tend to underperform and experienced teams tend to burn hot. It takes a few sprints to dial in on the team's point maximum.

Running The Workshop

The workshop format is critical, because it is necessary to have all of the features of 3 full sprints compressed into a tight schedule. Each of the 3 sprints must end at a designated time and ALL workshop participants must be present for all 3 demos. You may NOT postpone a demo. If you need additional time for story writing, add that onto the beginning of your workshop -- otherwise, simply use the stories that your team wrote in the previous workshop.

We strongly recommend that you provide refreshment to your workshop participants to keep them from wandering away and losing energy. A 3-Sprint workshop is a remarkably strenuous and taxing time. If you have a large number of people (more than 8) you should attempt to recruit an assistant.

Remember that your job is to keep the workshop on schedule. In particular, each demo must be performed at the scheduled time. Your are to foster communication between the developers and the rest of the team. It is not your job, or any one person's job, to answer all of the questions.

As the facilitator of the workshop, you should focus on the process, not the end result of what is actually developed. There must be a rigid agenda, with sprints One, Two, and Three beginning precisely at the planned time.

The overall structure of the workshop is:

1. An introduction and explanation of the process
2. Sprint One
3. Sprint Two
4. Sprint Three
5. A retrospective and reflection

Each sprint will consist of these phases:

1. A period of writing stories on notecards (10 minutes)

2. An estimation period with no more than 60 seconds spent on each story (10 minutes)
3. A development period that focuses on interaction between the customer and developers (80 minutes)
4. A demo (10 minutes)
5. A sprint retrospective which focuses on the process (10 minutes)
6. A break before the next sprint begins (15-30 minutes)

In greater detail, here's what should happen in each sprint:

Story Writing (10 minutes)

You should have stories already written from a previous workshop. If you choose to do a different project for this workshop it is recommended that you take time beforehand to write user stories, to prevent cutting into your workshop time. Use the time allotted for this first activity in the sprint to refine or add to the stories that have been created.

Story Estimating (10 minutes)

Although we prefer estimation in abstract "story points", we recommend that you estimate in minutes for this one-day workshop. That means any story with an estimate of more than 45 minutes is too big, and must be simplified or split into several smaller stories. You should spend no more than 60 seconds on estimating each story. If you already have your stories estimated from a previous workshop, then update story points to minutes and add estimates to any new stories. Ensure you are estimating in order of priority.

Development (80 minutes)

Here is where your storyboard comes into action. Your user stories will have been prioritized by the Product Owner and your team will move the number of stories they think they can accomplish within the 80 minute development period into the "In Sprint" column. As the developer picks up a story it gets moved across the board. At the completion of each story, a developer should demo the story immediately to a customer-facing person. Try to get them to do this before the card moves to QA on the storyboard.

Demo (10 minutes)

At the end of the development timebox, the developers will demo any cards that made it to QA. In a compressed timeframe, you may ask the crowd for a simple vote as to whether a story, after being demoed, should be considered passed or failed. It is common for suggestions to be made during a demo, which might be considered either bugs or potential improvements. In a full sprint you may develop a policy for this. In a one-day workshop we recommend that you simply ask the non-developers (the customer-facing team members) for a voice vote on whether each story should pass or fail. There is no shame in failing a story. It simply moves back to the "In Dev" column.

It is best to create a burndown chart as a big, visible wall chart. With the completion of each demo, add up the estimates of all passed stories and immediately add it to the burndown chart. Executives and other observers generally love the burndown chart, as it easily quantifies the amount of work being accomplished.

Sprint Retrospective (10 minutes)

The whole team discusses what worked, what didn't work, and how the process can be improved in the next sprint. The Agile Project Manager can capture this input and help coach the team in the next sprint.

Break (15-30 minutes)

Give the team a 15-30 minute break between sprints.

Things That Will Likely Happen

- You should expect a fair amount of confusion and skepticism, especially at first. However, this will abate as you push forward.
- There will be many, many questions that arise about each story. The workshop is forcing artificially fast story writing and very rapid estimation. Think of each story as a "promise to have a conversation".
- Sometimes developers will need to work with concentration without being interrupted. This is not one of those times. Communication is key to this exercise. The end product is less important than the team learning the process.
- Velocity may be uneven. It is fairly common for a sprint to fail to complete a single story.
- By about the second sprint, the customer-facing people will be waiting on the development team while the developers are furiously working. The customer-facing people should use this time to make sure the backlog of prioritized stories is exactly what they want.
- Expect that after each demo, some learning will surface that causes the Product Owner to change his or her priorities. Expect demos to result in new stories being written.
- Expect about three-fourths, but not all, of the stories to be passed by QA.
- Expect everyone to be astounded by what is accomplished in this single day.

When Things Go Wrong

To some extent, it is a positive part of the process for things to go wrong, if the team can learn from the experience. Here are some common problems you may face:

- The team can fail to break the goal into small steps that can be executed within the limited time of one sprint, resulting in nothing to demo.
- Developers can impose their own desires on what gets built.
- Customers can fail to assert priorities.
- A team can be so large that it is difficult for them to coordinate work within a sprint. If that's the case, break your participants into several smaller teams.

- The development team may devote too much energy to thinking about the final product and thus fail to do a good job presenting functionality in the first demo.

Workshop: Develop A Prioritized Backlog

- **Estimated Time**: 2 hours for the whole team (including customer representatives), 2 hours prep time for Agile Project Manager and Product Owner
- **Materials needed:** Tool of choice for your backlog (sticky notes, Trello, or other)
- **Outcome:** An estimated, prioritized backlog of stories that everyone can see -- estimated by engineers, prioritized by the customers.
- **Test:** Can everyone see the backlog? Is there consensus that these tasks will result in a useful functionality to the customer, however minimal it may be? Are the stories reasonably clear and in the standard format?

Introduction

You may have conducted your 3-Sprint Workshop on stories that are in fact aimed at your goal as a government program manager, or you may have used a simple exercise as practice. Now the time has come for you to create stories and a prioritized backlog that are definitely aimed at the program for which you are responsible. This is no longer an exercise -- this is work!

To begin your first full sprint with your full team, you need a fully prioritized and estimated backlog with enough stories to make several sprints' worth of stories that will provide actual value to your users.

Given the experience in story writing, estimating, and sprinting in the previous workshops, you have all the knowledge you need to make a fully prioritized and estimated backlog for the beginning of your real Agile project. As a government employee, you have the opportunity to make a greater impact with your project than many commercial firms ever hope to have. This is a is a good time to review the Agile Manifesto.

Remember, Agile processes are designed to help you serve your purpose, not to get in your way. However, most people have found that following Agile processes as taught in this curriculum allow you to produce software faster, with less risk, and with infinitely better customer satisfaction than older methods.

Running The Workshop

Choose the project that you will use for your first sprint and gather your Product Owner, designers, writers, and any other customer-facing stakeholders that will be important for identifying stories, along with your development team.

Drawing on what you learned from the User Story workshop, you will create your user stories and estimate them in preparation for your first Sprint Planning meeting. The format is the same as for the User Story workshop, but in this case you will have more time on the stories and will be more practiced at estimating.

The phases of the 2-hour workshop are:

1. High level overview of the project vision from the Product Owner (5 minutes)
2. Review of the INVEST mnemonic (5 minutes)
3. Writing of user stories (30 minutes)
4. Backlog grouping exercise (15 minutes)
5. Backlog prioritization exercise (5 minutes)
6. Quick estimation (60 seconds per story) - as much as 30 minutes
7. Reprioritization (5 minutes)
8. Format checking (5 minutes)
9. INVEST checking (5 minutes)

In greater detail, here's what happens in each phase:

High level overview of the project vision (5 minutes)

The Product Owner will tell the team about the high level vision for this project to set the scene for creating user stories that will meet project goals.

Review of the INVEST mnemonic (5 minutes)

The Agile Project Manager will give a brief introduction to the story format and the INVEST acronym, which are ideally written on a flip-chart or somewhere where everyone can see them.

Writing of User Stories (30 minutes)

Everybody takes a sharpie and cards. Everybody writes stories that they think advance the project goal. The Agile Project Manager and Product Owner should walk around answering questions during this time. People "publish" their stories on the wall or table as soon as they are written, but no criticism of stories is allowed during this phase.

Backlog grouping exercise (15 minutes)

All the cards are presented to the whole group. Stories that are duplicates or very closely overlapping are grouped together. Note that upon reading the stories of others, anyone may write new stories (and in fact this should be encouraged during any phase). If a story is consensually deemed to be obsolete or going in the wrong direction, it can be thrown away or moved into a reserve pile. During this time stories can be rewritten or reworded to make them more Independent.

Backlog prioritization exercise (5 minutes)

The stories are force-ranked into order, with the highest priorities at the top. There are no ties -- a specific order must be chosen. Open discussion is allowed. In the end the Product Owner sets the actual priorities. His or her authority to do this is absolute.

Quick estimation (60 seconds per story)

The Agile Project Manager must keep the discussion to 60 seconds per story, which will be very hard. People will want to discuss the stories far more. Under no circumstances can you take more than 120 seconds on a story. If 120 seconds goes by, the development lead writes something down no matter what.

It might seem that a longer time would produce better estimates. Experience has shown that this is not true, and that quick estimates are just as good as longer estimates.

If a single story is estimated to be of such effort that it exceeds a single sprint of work, split the story immediately into two or more simpler stories. It is always better to have at least 10 stories in a sprint.

Reprioritization (5 minutes)

Given the estimates, the stories may be reprioritized or adjusted. This is usually an important learning phase. There are often stories that are surprisingly easy or surprising hard, and given their cost, the order in which they are to be accomplished clearly changes.

Format checking (5 minutes)

Review that each story is in the correct format: “As a ___, I need a ___ in order to ___”.

INVEST checking (5 minutes)

Try to make sure that each story is testable by writing a description of the test on the back of the card. This should lead to a lively discussion between the developers and end-users.

Things That Will Likely Happen

- There will be many duplicate or similar user stories. You may refine as you go and consolidate cards.
- Some stories will have estimates that are very big. If a story is too big, it needs to be split into several stories. With practice this will become natural.
- You won’t get all the stories in this meeting to fulfill your first sprint, but you will come very close and the missing stories will be obvious as you are sprint planning.
- Estimating will be difficult -- keep to the timebox and remember that the estimates don’t have to be exact.

The Next Level

There are many important topics which have not been covered in this introductory study track:

- Automated testing
- Continuous integration
- Modern deployment tools
- Minimum Viable Products (MVPs)
- Lean startup continuous learning techniques

And many more! Once your agency begins to reap the benefits of Agile, you'll likely want to continue studying and pursuing this methodology that allows governments to bring delightful, effective services to citizens and customers.