AGILE
MADE EASIER

Improve
Project Visibility

Simplify
Collaboration

Centralized
Management

Accelerate
Agile Adoption

Agile Story Writing

eWorkshop Workbook

VERSIONONE

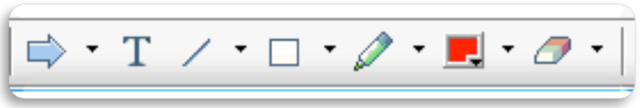
WORKBOOK

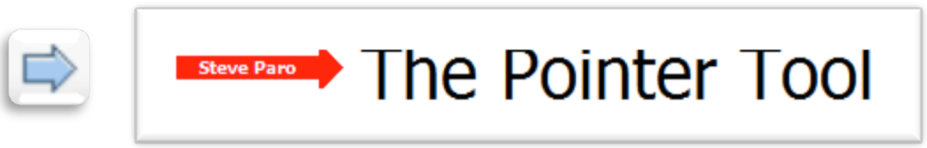
Table of Contents

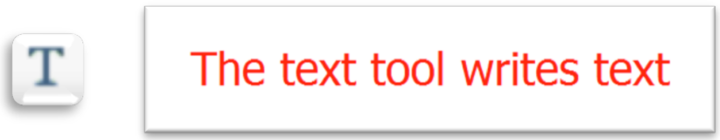
Live Session Logistics

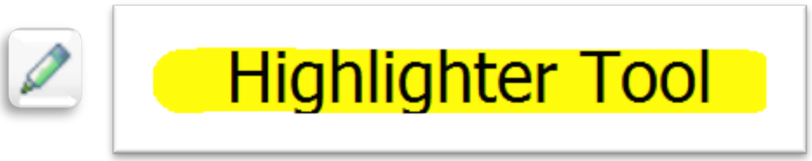
As much as possible, it is our desire to replicate a live, classroom experience. The WebEx Training Center application provides several communication and collaboration tools to facilitate this. To get the most out of this session, you should be prepared to actively participate in the main session conversation, breakout sessions and exercises. In addition to chat and audio conferencing the tools described below will help you get the most out of this session.

Annotation Tools

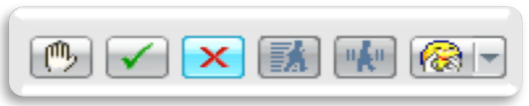


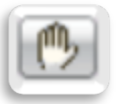


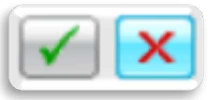


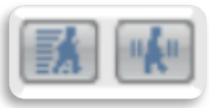


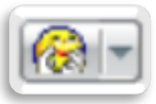
Feedback Tools




Raises hand for a question


Answer Yes or No


Go Faster or Slower

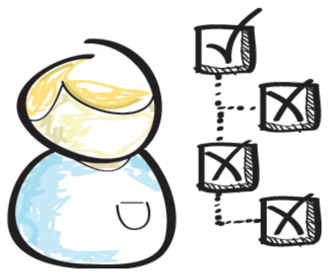

Feedback Emoticons

What is a User Story?

User stories are a common method of documenting and communicating requirements in agile projects. Unlike traditional requirements which attempt to define a detailed and seemingly contractual solution, User Stories simply seek to describe a problem or desired goal from the perspective of a specific user, and the benefit that solving the problem will produce. Focusing on the problem allows us to evolve the right solution and the right time.

User Role

It's all about perspective. To truly understand a problem, it's necessary to know the experiential context from which the problem statement has been defined. If the problem is that there is too much salt on the road, understanding who we are solving the problem for will help greatly in determining the actual solution we come up with. While the owner of a brand new car will surely want to avoid getting too much road salt on her shiny new toy, the consequences of crossing a salt laden road will be far more dire for Sammy the Garden Snail.



Functionality

It's not enough to simply deliver a solution. To truly provide value we must deliver the right solution. Too often requirements will simply describe a detailed solution, but they make no mention of the problem that needs to be solved. A requirement that states "The report shall be a left justified, grid format using courier font against a #FFF8C6 colored background. Displayed attributes in order are FName, LName, TrnCode, TrnDesc, TrnType, TrnAmount, queried based on date, time and account. Report shall include summary section at 70px from bottom." may give a lot detail; it leaves the actual problem open for interpretation. A requirement that states "I need to be able to view all new road salt order activity by customer and date" defines the goal of the requirement and leaves our options open to implement the most effective solution when the appropriate time comes.

Valuable

If a feature solves a problem that nobody needed solving, does it provide value? This is actually far more than just a rhetorical question. There is a very real cost associated with every single feature that we will build. Unless those finished features provide value, time spent working on them can only be described as waste. By defining the actual benefit realized by solving the problem as defined from the perspective of the user, we can be reasonably assured that there is real value to delivering the requirement. Equally important is that this allow the team building the solution to understand the 'why' behind the feature and use it as a test to whether a task can be attributed to adding value.

The Three C's

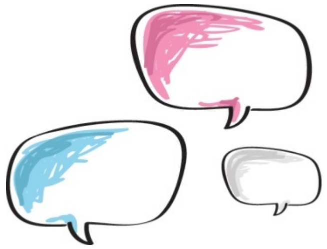
The 3 C's Include the Card, Conversation and Confirmation

Card

Early Agilists chose an index card to represent a story because of its small form factor. It kept the statement of requirement brief. The requirement was stated in the customer's language on the face of the card. On the back were listed the acceptance criteria. The card is a reminder of the need to have conversations about the requirement.

Conversation

They serve to flesh out the details of a story. Detailed requirements are discovered as a requirement moves closer to implementation, the conversations and the granularity of the decomposition of the requirement increase. This occurs through the conversation between the product owner / customer, Subject Matter Exerts and the development team.



Confirmation

This is an expression of the acceptance tests that will provide the confirmation criteria and document details for the story. The acceptance tests are used to ensure a requirement is completed to the specification of the product owner.

"While the card may contain the text of the story, the details are worked out in the Conversation and recorded in the Confirmation."

Mike Cohn, User Stories Applied: For Agile Software Development

Acceptance Criteria

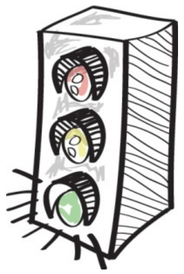
One of the most critical components of agile requirements is Acceptance Criteria as it describes, from the unique customers perspective, how that customer will know when the requirement has been satisfied. Without an understanding of the acceptance criteria it becomes difficult to truly understand the intent of the requirement. We will not know what's assumed or implied nor we will understand how the requirement will be tested or if it will in fact solve the problem we've set out to solve.

Document the Conversation

All agile requirements should start with a conversation. As we discuss the requirement we will better understand the problem we are trying to solve and the solution that is right for the customer. It is from this conversation that the acceptance criteria are born

Written by the Customer

With an understanding of the customer's needs comes an understanding of just how that specific customer will actually use the solution we come up with. The customer isn't going to care so much about the minute technical details of how the problem is solved, only that the problem is in fact solved. It is from this voice and perspective that the criteria should be written.



Valuable

"I'll know it when I see it" is not an acceptable expectation in regards to when a requirement is done, but without explicit acceptance criteria, it's all we've got. Understanding what it takes to complete a requirement is critical for many reasons. Without it, we can't effectively estimate and commit as we won't fully know what all is expected. Without it, we can't be sure that we're validating our work the way our customer intends to use it. Without it, it will be harder for work to be accepted as requirements will essentially be open-ended. Just as we wouldn't head out on a trip without first knowing our destination, we shouldn't be preparing to commit to working on requirements without knowing what it means for them to be considered done.

Why use User Stories?

Crystal Ball not Required

A User Story approach doesn't assume a crystal ball allowing all requirements to be thought of and fully flesh out up front as in more traditional waterfall based product development processes.

Avoid the Blame Game

In traditional waterfall approaches the Product Manager delivers to the Development Team a requirements document usually in the form of a Marketing Requirements Document and / or Product Requirements Document. The Development Team basically rewrites the document from their perspective.

Not Contracts

Traditional requirements documents looked like contracts and felt very final and they trigger legalistic stances between different parts of product development organizations. The Conversations that emerge the understanding of a story don't feel that way. If we have a conversation today, act on it and then learn something next month, we talk again.

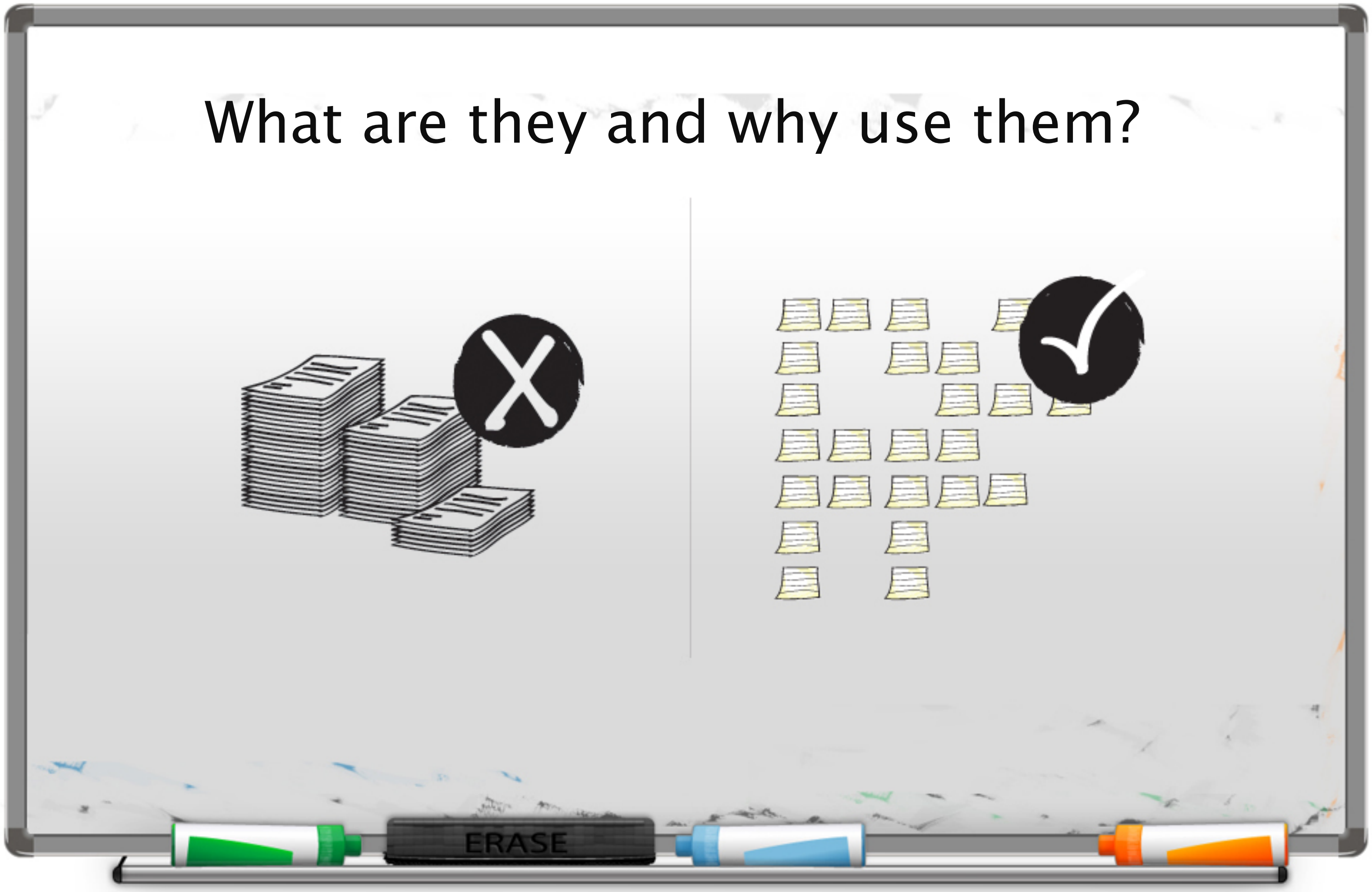
Encourage Conversation

Traditionally, all too often, there has been too little conversation between the product owner role and the development team to clarify and emerge the understanding of the requirements. Because of the inherent lack of viable precision / knowledge when the specs are written, and the lack of conversation, it can set a team up for a blame game session at the end of each release because the right solutions aren't delivered, quality is low and or the schedule is missed. This pattern causes wasted effort on the collection of metrics to defend different camps rather than having them collected to actively manage projects.

Nothing is Final

Much of this pain can go away when using User Stories effectively in an Agile approach. We utilize conversations to emerge an understanding of requirements and design instead of heavy upfront documentation approach. With User Stories we enjoy the reality that in fact nothing is final.

Exercise 1: Stories



User Roles and Personas

To be able to write effective stories for a proposed system or piece of software we must understand the primary User Roles these stories will serve.

What is a User Role? It is a collection of attributes that define and characterize a population of users and their intended interactions with the system.

Identify

How do we identify the Primary User Roles? Hold a User Role Brainstorming Session to identify an initial set of User Roles. In this session the assembled cross-functional team will identify a time box and then commence writing on cards or Post-Its all of the user roles they can think of for the solution space in which they are working. The team would then Organize the initial set of roles to enable the visualization of the relationships. This is done as a group usually using a white board or a wall and the team will arrange the user role cards or Post-Its on the wall representing logical groupings of relationships.

Consolidate

Consolidate the User Roles into a primary set of roles that will be the focus of the system

Refine

Refine the User Roles by modeling them through the definition of attributes of each role

User Roles and Personas

Continued…

What about Personas?

What

Persona is the name given to a more elaborately defined User role. Personas provide an imaginary representation of a user role to a development team. They are given a name and a picture. Stories are written directly for a persona and while they are being developed it is not uncommon for the team to role play the personal to test the stories.

Why

The benefit of Personas, especially in UI development is that they are easier for teams to relate to and make for a closer tie to the customer and more ownership of the User Experience.

How

Personas do requires adequate market & demographic research to be useful in representing a user role. Therefore before undertaking the use of Personas make sure you understand the commitment in time and cost and weigh this against the estimated benefit.

For more information on Personas see About Face 3: The Essentials of Interaction Design – Alan Cooper.

Trawling for Requirements

Robertson & Robertson 1999

Cast the net

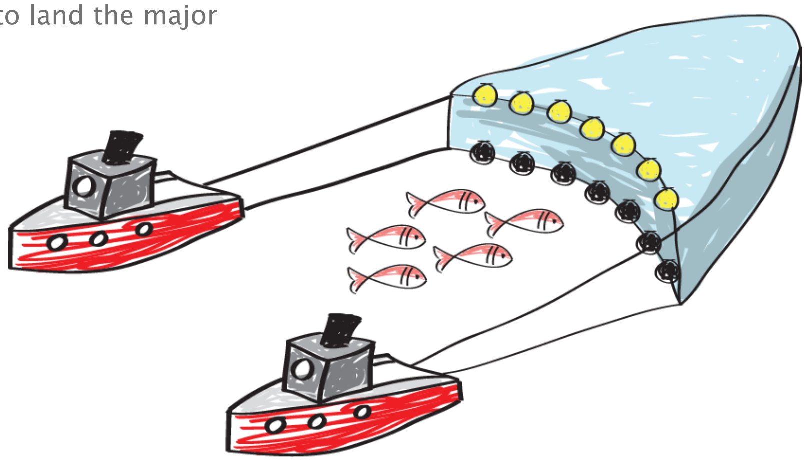
It is rare that Users already know all the requirements and we merely need to elicit them. It's effective to think of user requirements being captured like fish are when a fishing boat trawls for fish with nets. The fishermen utilize different gauge nets to catch different sizes and types of fish. Likewise, product owners, business analysts and other roles will trawl for requirements listening and testing for different types and sizes of requirements. The first pass would utilize a large gauge net to land the major requirements.

Adjust the gauge

We use progressively finer 'nets' at each pass of our requirement gathering efforts with the finest used at the Iteration pass. Knowing where to find requirements takes skill. To identify, capture and understand the requirements and their conditions for satisfaction, those gathering the requirements must ensure they speak to the right people, with the right perspective and also expect to

Evolve

Requirements for systems evolve. As you begin to show a customer user what their stated requirements look like through design work, prototypes or even working code, their understanding will change and they may well evolve the requirement based on this understanding. You've probably heard before from a customer or user "well that is what I asked for but now that I see it, it is not what I want. Also, it is important to not throw discoveries away because what was unimportant to a customer this month may well become a priority next month due to a better understanding or changing circumstances.



Requirements gathering Techniques

Different situations and contexts in the problem domain for which you are building a system / solution can call for different requirements gathering techniques. Some of the most valuable techniques are User Interviews, Observation, Questionnaires, and Story-Writing Workshops. Let's take a brief look at each of these techniques

User Interviews: Keys to successful interviews

Use your User Roles

Ensure you interview representatives of your key user roles. Interviewing people that are not representative of your key user roles is not only a poor use of time but also will distort the requirements and not enable delivery of the solution desired by the customer.

Ask open-ended and context free questions

"Would you like the new storage management application be browser based?" This is a closed-ended question. It doesn't provide sufficient detail for it to be answered and it assumes that users would know tradeoffs between a rich desktop client and a thin, browser based clients.

Better Question: What would you be willing to give up to have the new storage management application run in a browser? This question is open-ended and more context free. Context free questions don't have an implied answer.

Cast a wide net

Eventually you'll have to ask very specific questions to trawl for very detailed requirements but to start, that isn't what you want because you don't want to artificially limit the answers from the users… you might miss something important!

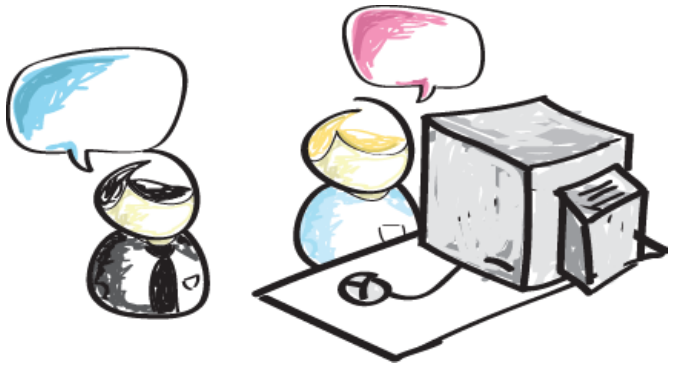
Requirements gathering Techniques

Questionnaires and Observation

Questionnaires

Questionnaires are excellent for getting information on Stories you already have captured in your backlog. This information if often useful for gaining a better understanding of the story, the conditions of satisfaction as well as identifying information that is useful for making prioritization decisions.

They are appropriate to use when you have large user populations to cover when addressing specific questions. They are not appropriate as a primary approach for trawling for requirements because they are not interactive. There is no opportunity for follow-up and clarification questions.



Observation

Observation is very valuable for getting grounded in the users' real needs. It is a great technique for identifying improvement opportunities in process and systems. Unfortunately opportunities to conduct observation can be difficult and expensive to arrange. For instance Usability Studies where teams can actually observe the users of the system they are building going through their work routines. Nothing motivates improvement by a team than seeing a customer struggling with what they thought was an easy to use solution.

If you can have the opportunity and means to conduct observation based requirements gathering session, this may well justify more frequent releases to demonstrate to users / customers that you've heard them and to test that you do in fact have alignment with the user on their needs and their capabilities as users.

Requirements gathering Techniques

Story Writing Workshops

The purpose of Story Writing Workshops is to enable a cross functional team to generates as many stories as possible in a fixed period of time. This follows in the pattern of most Agile work as being 'time-boxed' and teams need to get use to getting through the whole process in the time-box rather than dwelling on just the first step. A story writing workshop should have a strong facilitator that can ensure equal participation.

The Story Writing Workshop approach should combine brainstorming with "napkin" based prototyping of the solution that is the desired outcome of the customer. A Napkin based prototype is a low-fidelity prototype based on paper or a white board that maps very high-level interactions with the planned software

The goal is to identify conceptual workflows, not try to generate actual screens and data fields.

Use the prototype to find missing stories by asking questions as you're walking through it.

Questions to Ask

What will the user most likely do next?

What mistakes could the user make at this point in the workflow?

What could be confusing to the user at this point?

What supportive information might the user need at this point?

Be sure to maintain a parking lot of issues that when addressed may uncover changes or other new stories.

It is recommended that the team in a Story Writing Workshop pursue one path of the systems workflows at a time. This is to avoid the team from getting disoriented and losing their place in the discussion when the discussion tries to span across the breadth of the system. One other important aspect of the prototype that is built up during the workshop is that it should be seen as a 'throw away prototype' once it's served its purpose for aiding in the discovery of stories, it should literally be thrown away so as to not stagnate the conversation that will be needed to emerge the understanding of the requirements of the system.

The Story Writing Workshop will want to utilize the Key User Roles of the system being worked on that the team has identified in their earlier efforts.

The focus of the workshop should be on quantity not quality of stories - Use cards for speed and flexibility even if you will eventually house the stories in an electronic repository.

Exercise 2: Gathering Stories

Conduct a story writing workshop!

Your breakout group is a cross-functional team that has been tasked with bringing to market the next great social media platform, as described in the vision statement below. You are attending a story writing workshop already in progress. The customer has already identified several user stories, your objective is to using the leading questions, identify as many missing stories as possible within a 10 minute timebox.

Vision Statement:

For family and friend who want to stay in touch and not miss out on important events while protecting their privacy, the “Circle of Trust” is a web-based service that provides status updates, event planning, photo sharing and video conferencing. Unlike other social media services, our service protects people’s privacy while providing a cutting edge social experience.

Exercise 2: Gathering Stories – Steps

1. Designate a team member to be the note taker. This team member will use the text tool, designated by the "T" icon, to type on the screen.

2. Designate a team member to be the "customer" who can answer questions about the stories about the following stories:

   A. I need to be able to search for people I'd like to add to my network.

   B. I need to be able send messages to people in my network.

   C. I need to be able to block certain people from trying to add me to their network.

3. Using the worksheet provided in the session, fill in the missing stories for each of the identified user stories using the leading questions below. Have the note taker record answers.

   A. What will the user most likely do next?

   B. What Mistakes could the user make?

   C. What could be confusing?

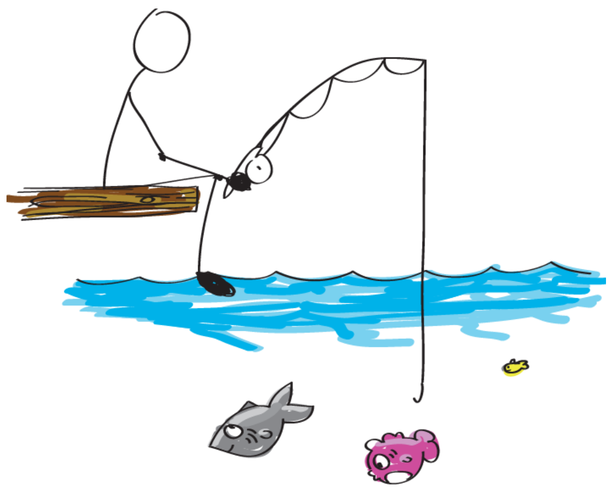
   D. What supportive information would be helpful?

Example: Need to be able to add photos
- ✓ Need to be able to tag people in photos
- ✓ Need a video demonstrating how to load new photos
- ✓ Need to be able to organize loaded photos into albums
- ✓ Need to only be allowed to only add valid image format files
- ✓ Need to be able to notify people when photos have been added

Writing good stories

Goal Stories

Consider starting with Goal Stories that describe for each Key User Role the Goals they have for using our software. These Goal Stories should be stated at a high level. Once you have the high level goals then you can turn your attention to breaking down the Goal Stories into stories of a finer level of granularity. This occurs as conversations about the stories emerge further understand of the requirements and their conditions of satisfaction.



Avoid architectural boundaries

One of the challenges for folks new to working with stories is deciding how to break their stories down. Technical folks tend to break down stories along architecture boundaries based on their experience. The problem with his approach is that it tends to isolate the layers of the architecture for extended periods of time before the solutions being built within each layer are integrated to provide a full solution. This makes architectural changes much more risky to implement because you don't know if the architecture design is adequate until much later in the game when there may be little time left to do something about a problem.

Breaking Stories down along architecture boundaries refers to writing stories that describe how a story is implemented in each layer of the architect. For instance:

- *As a VersionOne customer I want to fill out a form on line to register for a class (UI layer)*
- *The Information on the class registration form is written to a database (Persistence Layer)*

Slice the Cake

Bill Wake described in 2003 an approach for writing stories that he called "slicing the cake". This involves writing stories that exercise all the layers of a solution's architecture. This approach has several advantages including reducing risk to the overall implementation effort because it reduces the chances of finding dysfunction within and between layers later in the game.

Also, delivering slices can enable an organization to ship its product sooner should the need arise. If architecture layers are developed separately and then integrated later, this opportunity either does not exist or it comes with considerable more overhead to getting the product wrapped up and out the door.

Story Template

Improve consistency, understanding and value through the use of templates.

Story Template Form

When teams begin learning how to write stories it is generally seen as a good practice to utilize a common template for doing so. This ensures that stories are in a consistent format and contain their needed information. Perhaps the most popular story template is in the form:

As a <**Key User Role**>, I would like to <**Statement of function** / **utility**> so that I can <**statement of value**> and <additional statements of value>

Example: As a VersionOne user, I would like to be able to view dates for future e-learning courses, so that I may continue to learn about agile development.

Acceptance Criteria Template Form

When capturing stories teams will also capture the Stories' acceptance criteria, also known as conditions of satisfaction. These are the User's / Customer's functional acceptance criteria that the team will use to test whether a story is 'Done' per these functional acceptance criteria. Just as there is a template for stories, there is also a template for acceptance criteria.

Given <some condition>, **When** <User takes some action> **Then** <other possible actions> (And) <results> .

Example: Given I have selected the search option, when I enter a title, then I am able to run the search and see the results.

INVEST

INVEST is an acronym used to remind those writing user stories of how to tell if they meet the generally accepted criteria of good Agile stories.

Independent

- Avoid dependencies with other stories
- Write stories to establish foundation
- Combine stories if possible to deliver in a single iteration

Negotiable

- Stories are not a contract
- Too much detail up front gives the impression that more discussion on the story is not necessary
- Not every story must be negotiable, constraints may exist that prevent it

Valuable

- Each story should show value to the Users, Customers and Stakeholders

Estimable

- Enough detail should be listed to allow the team to estimate
- The team will encounter problems estimating if the story is too big, if insufficient information is provided, or if there is a lack of domain knowledge.

Sized Appropriately

- Each story should be small enough to be completed in a single iteration
- Small detailed stories for the near future
- Larger stories are okay if planned further out (Epics)

Testable

- Acceptance criteria stated in customer terms
- Automate whenever possible
- All team members should demand clear acceptance criteria

Exercise 3: Writing Good Stories

Identify the missing I-N-V-E-S-T initials from the stories below. Some stories may be missing more than one initial.

1. As a grandparent, I want to view photos of my grandchildren.
2. As a parent, I want to approve who my kids network with so I can protect them from internet predators.
3. I want to plan cool events.
4. As a parent, I want to modify the grandparent viewing photos story so that I am notified when they view the photos, so that I'll know what they thought.
5. As a teenager, I want to be able to sync my contacts with my iPhone, Facebook and AIM accounts so that I don't have to do it manually.
6. As a service administrator, I want to be able to suspend accounts with a single click, so that I can lockout people who violate our security agreement.

ERASE

Size Stories to the Planning Horizon

As the planning horizon for a story gets shorter the granularity of the details and understanding of a story should increase through emerging conversations.

Immediate

At the point of planning for an iteration enough details should be available to size and plan a story into an iteration.

Future

Stories that are further out on the planning horizon can remain less precise, in fact defining a story in detail before it is needed is frequently seen as wasteful because of the likelihood that circumstances may change and make the detailed definition of the story moot and then have to be redone.

Iterate

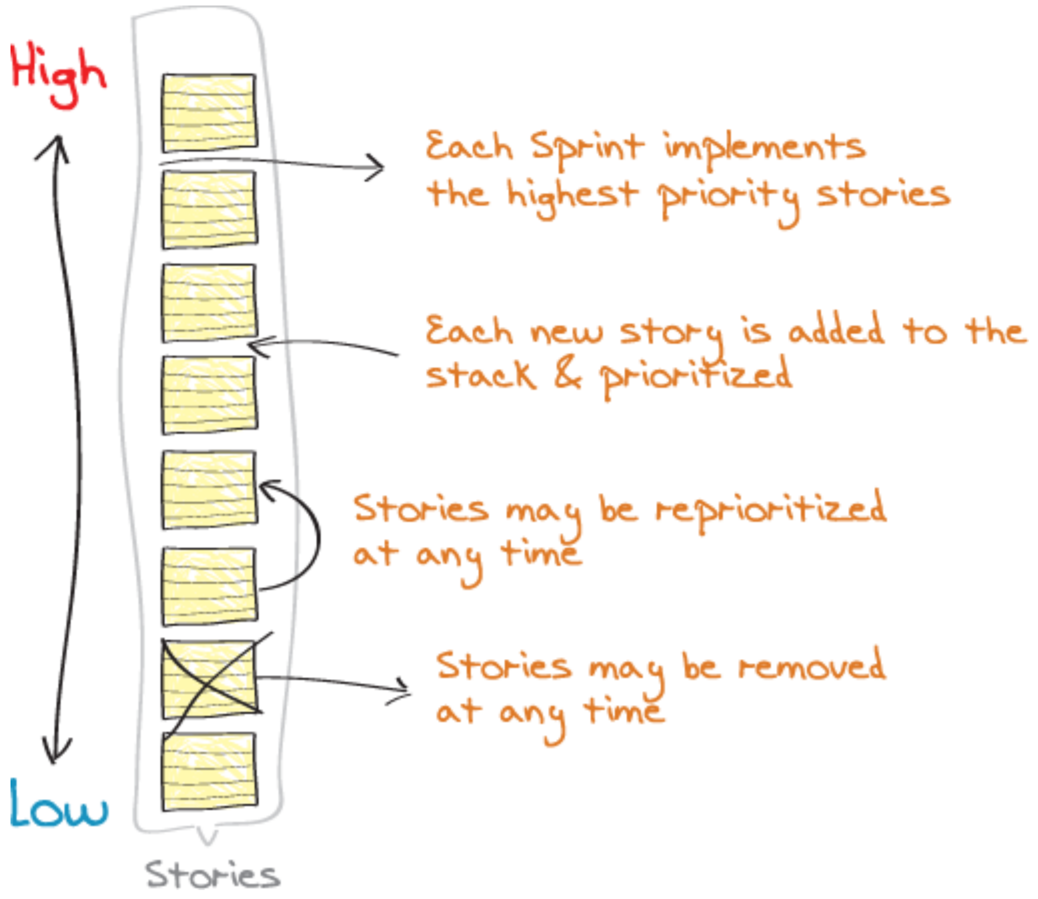
Stories are discovered throughout the product development process. They should be refined by what is learned while building the system during each iteration.

Product Backlog

The collective list of stories that represent all the capability desired in the solution being asked for by the Customer is called the Product Backlog.

Product Backlog

The Product Backlog is ranked in order of importance with the item at the top of the Product Backlog representing the most important customer requirement that the team could deliver next. Items at the top of the backlog will be defined in a fairly granular manner, while items further down the backlog, hence further away on the implementation horizon, will often be higher level expressions of value being sought by the Customer.


High
Low
Stories
Each Sprint implements the highest priority stories
Each new story is added to the stack & prioritized
Stories may be reprioritized at any time
Stories may be removed at any time

Further eWorkshops and Training

If you enjoyed this Agile eWorkshop, please consider registering for other classes in this series by visiting us at http://www.versionone.com/elearning. Students attending eWorkshops are eligible for 1 PDU per hour of instruction.

Agile eWorkshop: Story Writing

Learn how to write good stories, gather user requirements and properly size stories. Using agile best practices, our experts will provide hands-on exercises to prepare you for writing stories. Participants will leave the workshop with in-depth instructions for story writing, as well as the knowledge to help ensure the success of your stories.

Agile eWorkshop: Estimation

Learn about agile delivery models, how to estimate and how to use your estimates and velocity to determine how long a project may last. Using agile best practices, our experts will provide hands-on exercises to prepare you for estimating stories, features and task. Participants will leave the workshop with in-depth instructions for estimation, as well as the knowledge to help ensure the success of your estimation techniques.

Agile eWorkshop: Release Planning

Learn how to prepare a release plan, conduct a release meeting and manage the plan during the release. Using agile best practices, our experts will provide hands-on exercises to prepare you for release planning. Participants will leave the workshop with in-depth instructions for release planning, as well as the knowledge to help ensure the success of your releases.

Agile eWorkshop: Sprint Planning

Learn how to prepare a sprint plan, conduct a sprint meeting and manage the plan during the sprint. Using agile best practices, our experts will provide hands-on exercises to prepare you for sprint planning. Participants will leave the workshop in-depth instructions for sprint planning, as well as the knowledge to help ensure the success of your sprints.

Agile eWorkshop: Metrics

Learn how to read, use and act upon agile metrics. Using agile best practices, our experts will provide hands-on exercises to prepare you for using agile metrics. Participants will leave the workshop with in-depth instructions for using metrics, as well as the knowledge to help ensure the success of using agile metrics to aid your projects.

For a complete list of all VersionOne Training, please visit http://www.versionone.com/training.

Sources & Further Reading

- *User Stories Applied: For Agile Software Development*, by Mike Cohn, Addison-Wesley Professional
- *The Inmates Are Running the Asylum*, by Alan Cooper, Sams
- *Mastering the Requirements Process, Second Edition*, by Suzanne Roberston; James Robertson, Addison-Wesley Professional
- *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise*, by Dean Leffingwell, Addison-Wesley Professional
- http://www.agilesherpa.org/