User Stories

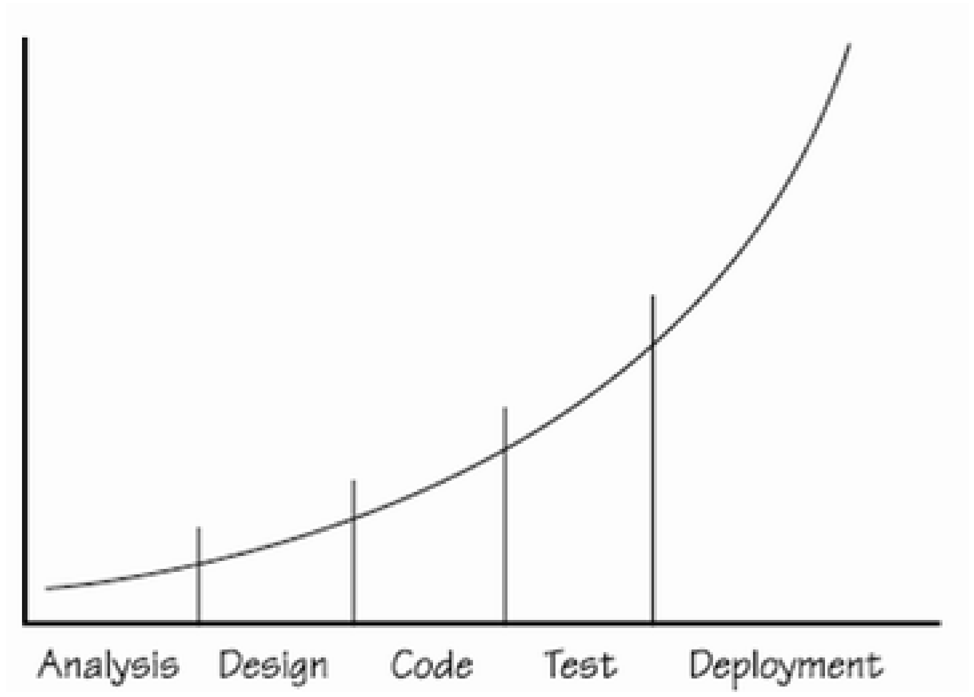


Figure 1 - Barry Boehm's Cost of Change Curve - 1981

Introduction

The real challenge in writing software isn't the time spent writing the code itself. Instead it's the time spent on deciding what software we should build and perhaps just as importantly what we shouldn't build. Traditional development focuses on getting everything right upfront. It's based on Boehm's Cost of Change curve - whose thesis is that change gets more expensive over time. Basic architectural principles (i.e. isolation), modern coding practices (Test Driven Development, Refactoring, Acceptance Test Driven Development, etc.) and modern tools (IDE's with built-in refactoring tools, Ruby vs. Fortran/C) have largely flattened the curve.

We spend so much time up front gathering traditional requirements trying to get the details right that we're reluctant to change them or throw them away as the user's needs evolve.

User Stories are an Agile approach to solving this problem; they're lightweight, simple requirements. We don't try to get the details right up front because inevitably they will change. They typically replace Use Cases and other heavy weight requirements. Our goal is to place the focus on the needs of the User. Instead of trying to be all the words that need to be said for a requirement a User Story is the very opposite. It's intended to start a conversation between the people who will implement the Story and the Customer/Product Owner.

User Story Background

The term “User Story” was first used Kent Beck in 1996 and became popular through their inclusion in the first Extreme Programming Project. They’ve been further refined by Ron Jeffries, Chet Hendrickson, Bill Wake and Alastair Cockburn. User Stories, while not required by Scrum, are often used by Scrum Teams as a way to represent Product Backlog Items. There are other ways to represent Product Backlog Items that are beyond the scope of this article.

The Three Components of User Stories

There are three components of User Stories: often referred to as the three C’s: Card, Conversations and Confirmations (from Ron Jefferies: http://xprogramming.com/articles/expcardconversationconfirmation/) :

- **Card**: A token (with a story title/description), used for planning and acting as a reminder to have conversations. The card is basically just a title or some descriptive text about the Story. You can use a card, or some descriptive text, like a sentence, to represent a story as a token to remind you to have conversations about the Story. It is a good practice to keep the title or descriptive text to as few words as possible. The purpose of the card is to remind us to have conversations.
- **Conversations**: Conversations that discuss the story details and result in one or more test confirmations.
- **Confirmations**: Acceptance Criteria that can be turned into automated acceptance tests. These automated tests are vital; and they are what enable the simple and light approach implemented by the first two C’s: Card and Conversations.

The rest of this article includes key points and illustrative examples of User Stories.

World's Smallest Online Bookstore

We will use this example to help illustrate the major points of User Stories.

Product Vision

Smallestonlinebookstore.com caters to the view that Amazon is wrong; an infinite supply of books is too much. Readers don't want an infinite supply; just the right choice for their next book. They don't want to spend hours agonizing, instead they want to spend that time reading. All books are read and reviewed by our staff. In addition we verify that public reviews come from real readers and not authors/publishers. When you have questions or problems we have real staff on hand to answer all of your questions. Our goal is to save you time (not sifting through hundreds of books) and money (by not selling you boring books).

Users

The “user” in User Stories can be any end user or stakeholder that is trying to get business value from the system. Examples from the Smallestonlinebookstore.com could be: Book Buyer (First Time, Frequent, Casual), Warehouse Staff, Site Team, Publisher (Small, Large), Director of Marketing.

Initial User Stories

This is the initial product backlog that was created by the Team in their release planning session.

As a first time book buyer I want to purchase my first book so that I can read it.

As a first time book buyer I want to find the perfect mystery novel so I can while away the time on my next plane flight.

As a frequent book buyer I want to buy my book with a minimum of hassle so that I don't waste my time.

As a first time book buyer I want to find another book by my favorite author Dan Airely so that I won't waste my time reading junk.

As a shipping clerk I want shipping labels printed on a self adhesive label so that I can easily add it to the final package.

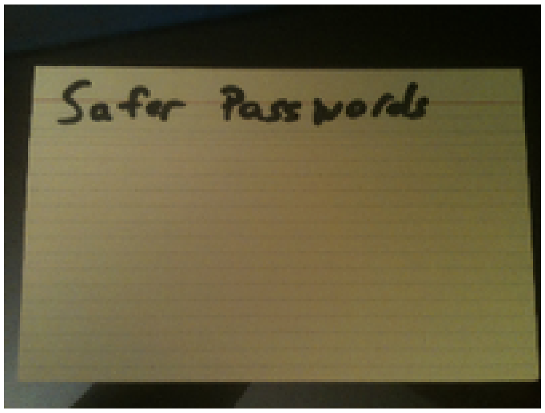
As a system administrator I want to be able to reset lost passwords with a minimum of hassle so I don't waste my time on trivial things.

Other Styles

Below is the same User Story described with 3 different styles:

#1 As a frequent book buyer I want strong passwords so that my credit card information remains secure.

#2



#3 In order to keep my credit card information secure as a frequent book buyer, I want the site to use strong passwords.

Conversations

User Story conversations are the second component of user stories. They can happen at any time and the nature of these conversations will vary widely from Team to Team. The main focus of the conversations will be to discuss details of the story and to create test confirmations (Acceptance Criteria that can be turned into automated tests).

Who participates in these conversations?

This varies widely as well, but frequently the Product Owner is involved since they have the last say on product vision and requirements. Sometimes users, stakeholders, testers, programmers, or even the whole Scrum Team are involved. To the extent possible, it is always highly encouraged that you get lots of key Stakeholders and Users involved in these conversations.

When do these conversations happen?
On an ongoing basis at any time, but especially as a part of Product Backlog Refinement. Product Backlog Refinement is an ongoing activity throughout a Scrum project, and includes adding, changing, removing, splitting and estimating Product Backlog items. It also includes defining Acceptance Criteria. Scrum Teams often allocate up to 10% of the time in the current Sprint to prepare for the next Sprint, and this preparation can include Product Backlog Refinement.

What is the expected outcome of these conversations?
User Story conversations (whether they be during Product Backlog Refinement or not) should result in shared Team understanding, and the Team and Product Owner coming up with test confirmations for each story. While conversations and minor changes to stories will continue inside of the Sprint, every reasonable attempt should be made to bring the Team to a shared understanding of the vast majority of story details and test confirmations (at least the Acceptance Criteria) before the Sprint begins.

Conversations at Smallestonlinebookstore.com

Returning back to our example product, the Team generally implements the conversation component of User Stories in these ways:

- The Development Team and Product Owner meet to do Product Backlog Refinement once a week for an hour or two in order to refine the stories that will be in the next Sprint.
- At spontaneous and often random times, conversations will happen with Stakeholders, Team members, etc. If there are any material updates to the story or its test confirmations, those involved in the conversation must be sure to communicate the updates to the rest of the Team.

Confirmations

Typically, test confirmations are created in two steps:

- Acceptance Criteria are communicated, and
- The Acceptance Criteria are turned into automated tests that verify that each has been met.

Acceptance Criteria

The goals of Acceptance Criteria are:

- To clarify what the Team should build (in code and automated tests) before they start work
- To ensure everyone has a common understanding of the problem
- To help the Team members know when the story is complete
- To help verify the story via automated tests.

When are Acceptance Criteria created?
As mentioned above, they should be very well known before implementation of the User Story begins. In addition, experience shows us that discovering these Acceptance Criteria at least a few days **before** the Sprint Planning meeting makes for a shorter, more productive meeting. Discovering them in advance of the Sprint also usually means the Team's commitment is closer to their real capacity.

Discovering numerous Acceptance Criteria after the Sprint Planning meeting usually leads to over-commitment and waterfall type behavior. Typically, Acceptance Criteria are created during User Story conversations and Product Backlog Refinement.

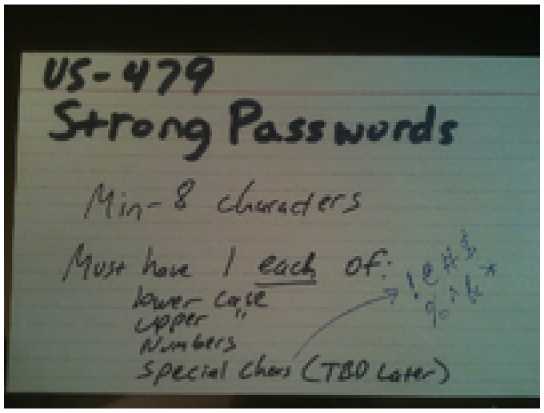
How do we communicate Acceptance Criteria?

There are numerous ways to communicate them that are beyond the scope of this article, but below are four examples for the story (introduced above) related to strong or safer passwords: **As a frequent book buyer I want strong passwords so that my credit card information remains secure".**

1) Example using "Test that..." style of Acceptance Criteria in an electronic form, like on a wiki.

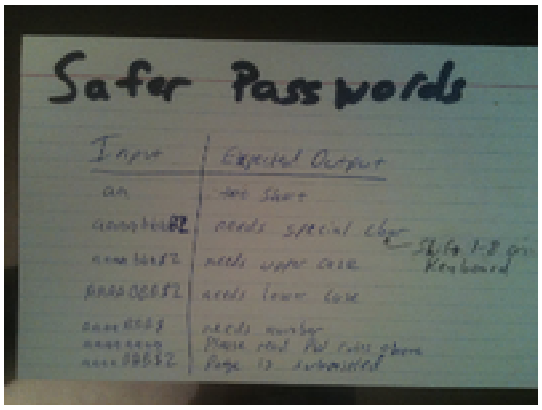
- Test that the system allows passwords with all of the following characteristics:
  - At least 8 characters
  - Contains at least one character from each of the following groups:
    - Lower case alphabet
    - Upper case alphabet
    - Numbers
    - Special Characters (!,@,#,$,%,^,&,*)
- Test that the system disallows passwords that are lacking any one of the above characteristics.

2) Example on a card where the Team had lots of conversations, took some notes on the card, and felt very confident that everyone had a shared understanding of what the Acceptance Criteria would be from looking at the notes on the card.



US-479
Strong Passwords
Min-8 characters
Must have 1 each of:
lower case
upper "
Numbers
Special chars (TBD Later)
!@#$
%^&*

3) Example using "Specification By Example" style of Acceptance Criteria on a card.


Safer Passwords

Input	Expected Output
aa	too short
aaaabbbb12	needs special char ← Shift 1-8 on Keyboard
aaaa bbb$2	needs upper case
AAAA BBB$2	needs lower case
aaaa BBB$	needs number
[illegible]	Please read the rules above
aaaa BBB$2	Page is submitted

4) Example using "[Specification By Example]" style of Acceptance Criteria in an electronic form, like on a wiki or spreadsheet.

Data	Expected Result	Expected Message
Aa9ab$	Fail	Too Short
AAbbCC11	Fail	No Special Characters
$$$bbb111	Fail	No Upper Case
AAA%%%1111	Fail	No Lower Case
AAAA%%%%bbbbb	Fail	No numbers
IsThis$AGood11	Pass	

Should we create documentation to communicate the Acceptance Criteria?

User Story practitioners should be careful about creating elaborate documentation to describe Acceptance Criteria, as we don't want to fall into traditional, wasteful, requirement habits. The User Story practice strongly emphasizes and prefers discussions (conversations) over documents. The point of User Stories is to communicate efficiently, and adding documentation can lead to waste and confusion. User Story practitioners should strive for the minimum amount of documentation that could possibly work! Having said that, it is OK to create very light documentation around Acceptance Criteria -- but again, the Team should strive for the minimum documentation that can possibly work.

Automated Tests

The second typical step to test confirmations is automated tests. Automated acceptance tests are what allow the User Story practice to thrive without the need for extensive documentation, such as that needed for traditional requirement practices.

Who automates the tests?

It does not matter who automates the tests, so long as someone automates tests that verify every Acceptance Criteria. Automated tests are what help flatten out the cost of change curve that was discussed at the beginning of this article.

When are the tests automated?

The **ideal** situation is to have them automated before the programmers begin implementing the story. That way, the programmers can get immediate feedback by running the tests against the code as they progress. The second best situation is when someone is automating the tests while the programmers are implementing the User Story, in parallel. Teams can also automate tests after the User Story has been implemented if they are disciplined about always doing a thorough job. On the other hand, automating after the fact often leads Teams to short change the automation of tests and this can lead to technical debt buildup and waterfall type behavior. Automated tests act as a long term, real-time verification of correct system behavior, and since the User Story practice encourages very little documentation, leaving the automated testing piece out can be very wasteful and risky.

User Story Key Points

- User Stories are relatively small: a few days' effort for one or a pair of Team members
- User Stories are focused on the what (the needs of the user) not the how (the technology).
- User Stories are the starting point for an ongoing collaboration between the Product Owner and the entire Scrum Team.
- User stories are best described in terms that users and stakeholders familiar with the domain would understand.
- Once implemented with supporting Automated Acceptance tests, User Story documentation can be disposed of.
- Many Teams use a template to help formulate the description of a User Story (the card): As <a user> I want to <do something> so that <value statement>. I.E. the user is able to achieve something they value with the functionality represented by the "I want statement". Not all Teams use this template, and as we discussed above, there are several other ways of communicating the User Story description.
- Not everything in the Product Backlog need be a User Story
- Not everything a Scrum Team does is associated with a specific User Story or Product Backlog Item, though all work of the Scrum Team should be made visible to the entire Scrum Team.

INVEST

Good User Stories can be measured against a simple set of criteria (from Bill Wake: http://xp123.com/articles/invest-in-good-stories-and-smart-tasks/):

Independent - dependencies between stories limit the flexibility of both the Product Owner and development Team. The Product Owner should be able to ask for stories in whatever order they make sense.

With some stories: "As a Canadian Book Buyer I want my book shipped to my Canadian address so that I can order painlessly" and, "As an American Book Buyer I want my book shipped to my US address so that I can read it quickly". These stories are independent because they can implement in whatever order the Product Owner sees fit. However, the Team may tell the Product Owner that whichever one is implemented first it will be more expensive as we have to build the infrastructure to support addresses.

Subsequent address stories will be cheaper because the hard work has already been done.

Negotiable - the elegance of a User Story is that the precise details are left until later. It gives the Product Owner and Team a chance to delay unnecessary decision making until implementation begins. It allows the Team to discover new options right up until they're done.

Valuable - Each Story needs to deliver some small sliver of value all on its own. In other words, the customer has to be able to see the value. This pushes us towards slicing our work into **vertical**[1] chunks and not technological layers.

In addition, this prevents creating "Infrastructure" User Stories. Scrum and XP take the view that Infrastructure should only be built to solve the current problem.

Estimable - if the Team through lack of experience or differences in technical understanding can't estimate a Story, they shouldn't fake it - instead they should run a short experiment to gain that experience. These experiments are called Spikes (see the associated GASP article on Spikes - hint to Ron we need one).

Sized Appropriately - Stories at the top (~3 sprints) should be small, ***so small that the Team should be able to get 5-10 similar sized stories completed every sprint***[2]. Stories in the middle of the backlog (between 4 - ~10 sprints out) should be larger. The Team might only complete 1-2 of these in a Sprint. Further out and the Stories are very large.

Testable - It is clear how you will test the Story.

Epics & Themes

Epics are Stories that won't be implemented until further into the future (i.e. >4 Sprints away), that are often too large or too vague to be completed in one Sprint. In a way, it's good that Stories this far out are often vague and/or large, because it saves us the effort of refining Stories whose needs will change often, or even disappear, in the coming weeks.

A **Theme** is just a collection of User Stories that have some unifying trait or concept. It is sometimes useful to talk in terms of Epics and Themes when doing longer range planning like release planning and roadmapping.

Conclusion

[1] Vertical refers to a story that slices its way through all the software layers i.e. Database, Business Logic, UI. Horizontal refers to specific software layers i.e. the Database, Business Logic etc. For more detail see: http://scrumftw.blogspot.ca/2008/10/slices-verticals-user-stories-and-scrum.html

[2] Interestingly this seems to vary with team size and more than sprint length: "Based on data I analyzed on successfully finished sprints, I determined that a team should average around 1 to 1-1/2 user stories (product backlog items of any sort, really) per person per sprint. So, a six-person team should get somewhere around 6-9 user stories done per sprint." http://blog.mountaingoatsoftware.com/should-the-daily-standup-be-person-by-person-or-story-by-story

One way to tell if your Team is implementing the User Story practice well is to look at the questions below. If your answer to any of the below is “No”, then your Team needs to inspect and adapt your User Story practices.

- When we demonstrate and deliver the software functionality of a User Story, is it only a rare occasion when we fail to meet Product Owner, user, or stakeholder desires?
- Do we have automated tests that verify nearly all Acceptance Criteria from previously implemented user stories?
- Are we using the least amount of documentation that could possibly work, while also satisfying the above two questions?

Backlog of possible changes:

Issue #	Description	Status
1	*"Card: A token" - Not sure what is meant by a "token"; is it literally a card or a just a verbal representation of something ? This was vague to me and I know I could Google it, but I don't think you want me to have to .*	*Won't fix - delete after 1st public review.*
2	" User Story practitioners should strive for the minimum amount of documentation that could possibly work!" - Absolutely love this sentence! Talk about an amazing argument for Scrum. Our old way of doing the software development process included about 8 documents - all sorts of sequence and process diagrams, as well as business specifications and requirements documents. This is about 100 pages for our website alone. And it all had to be done before coding began, in theory. It made you dread any new project. Also, whenever anytime anything changed, all documentation had to be updated. In Scrum, who does the actual documentation of a system for the user (like a "user manual?) , or is there any? I assume that would be left to a technical writer or project expert?	1.1
3	"Leaving the automated testing piece out can be very wasteful and risky. " - I obviously understand why it 's risky not to test, but why wasteful? Not intuitive as to what that means.	1.1
4	"Not everything in the Product Backlog need be a User Story. " - What does and what doesn't? Is it items that are purely technical that don't need to involve the user that don ' t need to be in your Product Backlog, i.e., "Create SQL Server 2008 customer tables"? Or what defines that?	1.0?
5	"Not everything a Scrum Team does is associated with a specific User Story or Product Backlog Item . " Again, what is the criteria for deciding what a Scrum team does that doesn't live in the Product Backlog?	1.0 - same as above

6	"If the dependency is just around sizing/effort then it's ok for the team to say: 'This type of story requires a templating engine, the first one will be sized 13, all subsequent stories will be sized 2 '." - Huh?	Done
7	"Scrum and XP take the view that Infrastructure should only be built to solve the current problem." - Does this mean that nothing is built to be reusable? That we ONLY take into account the current project and not develop a framework or worry about extensibility to other projects?	1.1
8	"Sized Appropriately - Further out and the stories are very large. " - How large? And maybe an example?	1.1
9	I think it would also help provide a bit more structure if you tell me what you're going to tell me beforehand. In this chapter I'll cover the an overview of user stories-- the three c's, testing, and.... This comes from my days as an instructional designer. It seems like overkill, but it is helpful in framing things for the reader.	1.0?
10	does this audience know what scrum is? (Should we consider documenting what we expect to be assumed pre-requisite knowledge?)	1.1
11	can you talk about the format of conversations...informal stand ups? Frequency?	1.1
12	I like real world examples. I think you've done a good job with the bookstore, esp. the sample cards, but I'd integrate it further. For example-- a sample conversation gives me an idea of how a conversation might look. Even if its a brief look into a conversation. I think the more integrated the example is, the better the flow and more cohesive.	1.1
13	if this is for users transitioning from waterfall, I might give more examples of how user stories are effective in breaking down communication barriers, what common challenges are in transitioning to stories, etc	1.1
14	Handling of non-functional requirements and user stories	1.1
15	Add references to external sources i.e. Mike's book; Bill on Story Splitting, ….	1.2
16	Extend the use of the examples into INVEST	1.1

17	A slightly advanced technique, but only slightly, is to build the infrastructure spread equally over the two stories. If only one is done, this leaves you with a little catchup, but even then it can be a good move.	1.1
18	**Placeholders for Better Self Study Questions** Does a User Story contain enough information to be implemented right away? Who should talk before implementing a User Story? What doesn't a User Story include? How much detail is in the User Story? Why are User Stories memorable? Why is that helpful? Why are the following poor quality User Stories: As a frequent book buyer I want to search by title. As a user I want to ship my book home quickly so I can start reading As a frequent book buyer I want my credit card stored in an Oracle 11x database so that	1.1
17		
18		
19		