**Agile Methodology, Scrum Practice, and Effective User Story**

**Introduction**

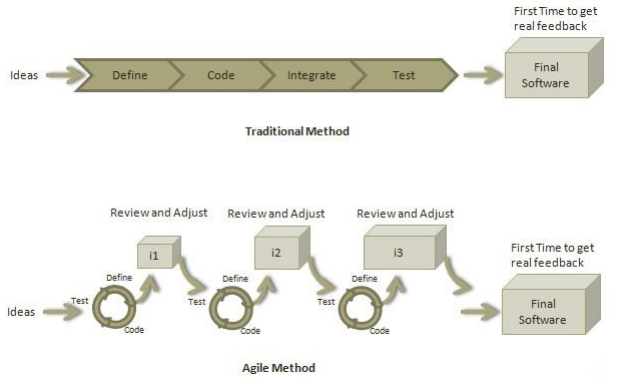
Without user knowledge, we tend toward analytical approaches to information architecture planning and user interface design — structures that make sense to us and the way we think about the information we want to convey, but that don’t always make sense to our audience. To balance out that analytic thinking, we need to introduce narratives to our planning procedures. Specifically, we need to be thinking through every design decision based upon a **need** >> **experience** >> **outcome** narrative. ***What does the user need? What kind of experience does our solution create? Does our solution work?***

We need a method for folding user knowledge back into the design process so that we properly understand user needs, create the best user experience, and reliably bring about the right outcomes.

A user story is a brief statement that identifies the user and her need. It is a tool used in Scrum practice and Agile development. It helps to create a system or product that really meets the needs of those it is meant for, and helps the product designer and developer to accurately engineer a system that is relevant at deployment, using proven methods for efficient product development.

**What is Agile Methodology**

Agile is a software development methodology to build a software incrementally using short iterations of 1 to 4 weeks so that the development process is aligned with the changing business needs. Instead of a single-pass development of 6 to 18 months where all the requirements and risks are predicted upfront, Agile adopts a process of frequent feedback where a workable product is delivered after 1week to 4weeks iteration.



The image above captures the entire idea behind the Agile software development methodology.

The following are true in Agile development:

1. Individuals and interactions over Processes and tools
2. Working software over Comprehensive documentation
3. Customer collaboration over Contract negotiation
4. Responding to change over Following a plan

That is, while there is value in the items on the right, the items on the left are more valued.

Twelve Principles of Agile development

1. ***Customer Satisfaction***: highest priority is given to satisfy the requirements of customers through early and continuous delivery of valuable software.
2. ***Welcome Change***: changes are inevitable during software development. Everchanging requirements should be welcome, even late in the development phase. Agile processes should work to increase customers’ competitive advantage.
3. ***Deliver a Working Software/product***: deliver a working software frequently, ranging from a few weeks to a few months, considering shorter time-scale.
4. ***Collaboration***: business people and developers must work together during the entire life of a project.
5. ***Motivation***: projects should be built around motivated individuals. Provide an environment to support individual team members and trust them to make them feel responsible to get the job done.
6. ***Face-to-face Conversation***: face-to-face conversation is the most efficient and effective method of conveying information to and within a development team.
7. ***Measure the Progress as per the Working Software***: working software is the key and it should be the primary measure of progress.
8. ***Maintain Constant Pace***: agile processes aim towards sustainable development. The business, the developers, and the users should be able to maintain a constant pace with the project.
9. ***Monitoring***: pay regular attention to technical excellence and good design to  
   enhance agility.
10. ***Simplicity***: keep things simple and use simple terms to measure the work that is not completed.
11. ***Self-organized Teams***: an agile team should be self-organized and should not depend heavily on other teams because the best architectures, requirements, and designs emerge from self-organized teams.
12. ***Review the Work Regularly***: review the work done at regular intervals so that the team can reflect on how to become more effective and adjust its behavior accordingly.

From these principles, 3 characteristics can be drawn for Agile development methodology, they include the following:

1. Iterative, incremental, and ready to evolve
2. Face-to-face communication
3. Feedback loop.

To accomplish tasks, and track progress of the development cycle it is vitally important that an agile team adopt the culture of having a daily stand-up meeting, no matter how the team is established and regardless of its office location. Daily stand-up, as the name suggests, is a daily status meeting among all the members of an agile team. It not only provides a forum for regular updates but also brings the problems of team members into focus so that it can be quickly addressed.

**What is Scrum Practice**

Scrum is a framework for developing and sustaining complex products. It is a framework within which people can address complex adaptive problems, while productively and creatively delivering products of the highest possible value. With Scrum, it is easy to divide large organizations into small teams, and far-reaching projects into a series of short time horizons called sprints.

Scrum is not a process or a technique for building products; rather, it is a framework within which you can employ various processes and techniques. Scrum makes clear the relative efficacy of product management and development practices so that you can improve.

The Scrum framework consists of Scrum Teams and their associated roles, events, artifacts, and rules. Each component within the framework serves a specific purpose and is essential to Scrum’s success and usage. The rules of Scrum bind together the events, roles, and artifacts, governing the relationships and interaction between them.

Scrum Theory

Scrum is founded on empirical process control theory, or empiricism. Empiricism asserts that knowledge comes from experience and making decisions based on what is known. Scrum employs an iterative, incremental approach to optimize predictability and control risk. Three pillars uphold every implementation of empirical process control: transparency, inspection, and adaptation.

1. Transparency: significant aspects of the process must be visible to those responsible for the ***outcome***. Transparency requires those aspects be defined by a common standard so observers share a common understanding of what is being seen. For example: a common language referring to the process must be shared by all participants; and, those performing the work and those accepting the work product must share a common definition of “**Done**”.
2. Inspection: Scrum users must frequently inspect Scrum artifacts and progress toward a Sprint Goal to detect undesirable variances. Their inspection should not be so frequent that inspection gets in the way of the work. Inspections are most beneficial when diligently performed by skilled inspectors at the point of work.
3. Adaptation: If an inspector determines that one or more aspects of a process deviate outside acceptable limits, and that the resulting product will be unacceptable, the process or the material being processed must be adjusted. An adjustment must be made as soon as possible to minimize further deviation. Scrum prescribes formal events for inspection and adaptation.

Scrum Events

Events are used in Scrum to create regularity and to minimize the need for meetings not defined in Scrum. All events are time-boxed events, such that every event has a maximum duration. Once a Sprint begins, its duration is fixed and cannot be shortened or lengthened. The remaining events may end whenever the purpose of the event is achieved, ensuring an appropriate amount of time is spent without allowing waste in the process. Events are specifically designed to enable critical transparency and inspection. There are 4 major Scrum events, and failure to include any of these events results in reduced transparency and is a lost opportunity to inspect and adapt

***Sprint*** is the heart of Scrum. It is a time-box of one month or less during which a “Done”, useable,  
and potentially releasable product Increment is created. Sprints best have consistent durations  
throughout a development effort. A new Sprint starts immediately after the conclusion of the  
previous Sprint.

During the Sprint:

1. No changes are made that would endanger the Sprint Goal
2. Quality goals do not decrease
3. Scope may be clarified and re-negotiated between the Product Owner and Development Team as more is learned (Scrum team is discussed below).

Sprint contains and consists of the following formal Scrum events which include the following:

1. Sprint Planning: The work to be performed in the Sprint is planned on this event. The collaborative work of the entire Scrum Team creates this plan. Sprint Planning is time-boxed to a maximum of eight hours for a one-month Sprint. For shorter Sprints, the event is usually shorter. Sprint Planning answers the following:
2. What can be delivered in the Increment resulting from the upcoming Sprint?
3. How will the work needed to deliver the Increment be achieved?
4. Daily Scrum: is a 15-minute time-boxed event for the Development Team to synchronize  
   activities and create a plan for the next 24 hours. This is done by inspecting the work since the last Daily Scrum and forecasting the work that could be done before the next one. The Daily Scrum is held at the same time and place each day to reduce complexity. During the meeting, the Development Team members explain:
5. What did I do yesterday that helped the Development Team meet the Sprint Goal?
6. What will I do today to help the Development Team meet the Sprint Goal?
7. Do I see any impediment that prevents me or the Development Team from meeting the Sprint Goal?

The Development Team uses the Daily Scrum to inspect progress toward the Sprint Goal and to inspect how progress is trending toward completing the work in the Sprint Backlog. The Daily Scrum optimizes the probability that the Development Team will meet the Sprint Goal. The ***Sprint Goal*** is an objective set for the Sprint that can be met through the implementation of Product Backlog. It provides guidance to the Development Team on why it is building the Increment. It is created during the Sprint Planning meeting

1. Sprint Review: is held at the end of the Sprint to inspect the Increment and adapt the Product Backlog if needed. During the Sprint Review, the Scrum Team and stakeholders collaborate about what was done in the Sprint. Based on that and any changes to the Product Backlog. During the Sprint, attendees collaborate on the next things that could be done to optimize value. This is an informal meeting, not a status meeting, and the presentation of the Increment is intended to elicit feedback and foster collaboration. The Sprint Review includes the following elements:
2. Attendees include the Scrum Team and key stakeholders invited by the Product Owner
3. The Product Owner explains what Product Backlog items have been “Done” and what has not been “Done”
4. The Development Team discusses what went well during the Sprint, what problems it ran into, and how those problems were solved
5. The Development Team demonstrates the work that it has “Done” and answers  
   questions about the Increment
6. The Product Owner discusses the Product Backlog as it stands. He or she projects likely completion dates based on progress to date (if needed)
7. The entire group collaborates on what to do next, so that the Sprint Review provides  
   valuable input to subsequent Sprint Planning
8. Review of how the marketplace or potential use of the product might have changed  
   what is the most valuable thing to do next
9. Review of the timeline, budget, potential capabilities, and marketplace for the next  
   anticipated release of the product.

The result of the Sprint Review is a revised Product Backlog that defines the probable Product Backlog items for the next Sprint. The Product Backlog may also be adjusted overall to meet new opportunities.

1. Sprint Retrospective: is an opportunity for the Scrum Team to inspect itself and create a plan for improvements to be enacted during the next Sprint. The Sprint Retrospective occurs after the Sprint Review and prior to the next Sprint Planning. This is a three-hour time-boxed meeting for one-month Sprints. For shorter Sprints, the event is usually shorter. The purpose of the Sprint Retrospective is to:
2. Inspect how the last Sprint went with regards to people, relationships, processes, and tools
3. Identify and order the major items that went well and potential improvements
4. Create a plan for implementing improvements to the way the Scrum Team does its work.

By the end of the Sprint Retrospective, the Scrum Team should have identified improvements that it will implement in the next Sprint. Implementing these improvements in the next Sprint is the adaptation to the inspection of the Scrum Team itself. Although improvements may be implemented at any time, the Sprint Retrospective provides a formal opportunity to focus on inspection and adaptation.

The Scrum Team

This consists of a Product Owner, the Development Team, and a Scrum Master. Scrum Teams are self-organizing and cross-functional. Self-organizing teams choose how best to accomplish their work, rather than being directed by others outside the team. Cross-functional teams have all competencies needed to accomplish the work without depending on others not part of the team.

The team model in Scrum is designed to optimize flexibility, creativity, and productivity. Scrum Teams deliver products iteratively and incrementally, maximizing opportunities for feedback. Incremental deliveries of “Done” product ensure a potentially useful version of working product is always available.

Roles in a Scrum team

1. A Scrum Master is a team leader and facilitator who helps the team members to follow  
   agile practices so that they can meet their commitments. The responsibilities of a scrum  
   master are as follows:
2. To enable close co-operation between all roles and functions.
3. To remove any blocks.
4. To shield the team from any disturbances.
5. To work with the organization to track the progress and processes of the company.
6. To ensure that Agile Inspect & Adapt processes are leveraged properly which includes:

* Daily stand-ups,
* Planned meetings,
* Demo,
* Review,
* Retrospective Meetings, and
* To facilitate team meetings and decision-making process.

1. Product Owner is responsible for maximizing the value of the product and is the sole person responsible for managing the Product Backlog. Product Backlog management includes:
2. Clearly expressing Product Backlog items
3. Ordering the items in the Product Backlog to best achieve goals and missions
4. Optimizing the value of the work the Development Team performs
5. Ensuring that the Product Backlog is visible, transparent, and clear to all, and shows what the Scrum Team will work on next
6. Ensuring the Development Team understands items in the Product Backlog to the level needed.

The product owner is commonly a lead user of the system or someone from marketing, product management or anyone with a solid understanding of users, the market place, the competition and of future trends for the domain or type of system being developed. The key is that the person in the product owner role needs to have a vision for what is to be built.

1. Development Team consists of professionals who do the work of delivering a potentially  
   releasable Increment of “Done” product at the end of each Sprint. Only members of the Development Team create the Increment. Development Teams are structured and empowered by the organization to organize and manage their own work. The resulting cooperation optimizes the Development Team’s overall efficiency and effectiveness. Development Teams have the following characteristics:
2. They are self-organizing. No one (not even the Scrum Master) tells the Development Team how to turn Product Backlog into Increments of potentially releasable functionality
3. Development Teams are cross-functional, with all of the skills as a team necessary to create a product Increment
4. Scrum recognizes no titles for Development Team members other than Developer, regardless of the work being performed by the person; there are no exceptions to this rule
5. Scrum recognizes no sub-teams in the Development Team, regardless of domains that need to be addressed like testing or business analysis; there are no exceptions to this rule
6. Individual Development Team members may have specialized skills and areas of focus, but accountability belongs to the Development Team.

Although the agile Product Owner prioritizes the product backlog during the sprint planning meeting, the team selects the amount of work they believe they can do during each sprint, and how many sprints will be required.

Summary of Scrum

Scrum is an agile development practice, and an iterative and incremental software development framework for managing product development. Teams use scrum to manage work, and implements the Agile principles as a set of artifacts, practices, and roles

In scrum, ***features*** are written from the perspective of the end users, and are known as ***User Story***. A collection of User story makes up the ***product backlog***, which gives an overview of the functions of the system. These features are narrowed based on present needs, both real and predicted, and the narrowed features with which development can commence make up the ***release backlog***. From release backlog, we get a list of ***Sprints*** which are short duration milestones throughout the project. A tracking tool i.e. ***burndown chart*** is put in place to monitor the progress of each sprint. It is also important to monitor the progress of each Team member (esp. Development team), and this is achieved by setting up what is called a daily ***standup meeting*** (which is meant to last 15mins). After this cycle, in which there is a ship ready product, it is important to have a ***sprint retrospective*** meeting, where all the deliverables, and processes are reviewed.

**What is User Story**

A user story is a tool used in Agile software development to capture a description of a software feature or a listed acceptance criteria to fulfil certain requirements from an end-user perspective.

Effective user stories express the needs of users and support effective communication and collaboration between product owners and agile teams. They are prompts for communication, shifting the focus from writing about requirements to talking about them, which help to understand users and develop the right products.

All agile user stories include a written sentence or two and, more importantly, a series of conversations about the desired functionality. A ***user*** salesperson could say

As a <user type>, I want to <function> so that <benefit>.

* As a consumer, I want shopping cart functionality to easily purchase items online.
* As an executive, I want to generate a report to understand which departments need to improve their productivity.
* As a salesperson, I want to create a quarterly forecast so that I may share my sales goals.

***Story Points*** is a unit used by the agile team to estimate relative sizes of user stories and features.

Characteristics of a User Story

1. It is common to Agile methodology
   1. Useful with scrum
   2. Popular with new teams
2. Allows to defer detailed user requirement until the last responsible moment: closest to implementation
3. Designed to be brief: meant to encourage interaction between customers and the dev. team
4. High Level
5. Place holder for conversation

What is not a User Story

1. Detailed requirements:

⇒ user story defers design/implementation requirement.

1. Set in stone

⇒ user story is designed to be flexible

Why Use User Stories?

1. Keep yourself expressing business values
2. Avoid introducing detail too early that would prevent design options and inappropriately lock developers into one solution
3. Avoid the appearance of false completeness and clarity
4. Get to small enough chunks that invite negotiation and movement in the backlog
5. Leave the technical functions to the architect, developers, testers, and so on

Well-formed stories will meet the criteria of Bill Wake's **INVEST** acronym:

1. Independent: We want to be able to develop in any sequence.
2. Negotiable: Avoid too much detail; keep them flexible so the team can adjust how much of the story to implement.
3. Valuable: Users or customers get some value from the story.
4. Estimable: The team must be able to use them for planning.
5. Small: Large stories are harder to estimate and plan. By the time of iteration planning, the story should be able to be designed, coded, and tested within the iteration.
6. Testable: Document acceptance criteria, or the definition of done for the story, which lead to test cases.

User stories are often written on index cards or sticky notes, saved on a whiteboard or online tool e.g. Trello, and arranged on walls or tables to facilitate planning and discussion. As such, they strongly shift the focus from writing about features to discussing them. In fact, these discussions are more important than whatever text is written.

A user story is defined incrementally, in three stages:

* The brief description of the need
* The conversations that happen during backlog grooming and iteration planning to solidify the details
* The tests that confirm the story's satisfactory completion

User Story groups and sizes?  
User stories come in different sizes. A small story, called simply a “user story,” is one that is well understood and can be implemented within a sprint—something like: “As a manager, I want the monthly sales report to be paginated so that I can quickly refer to a specific part of the document.” A story should be small enough to be coded and tested within an iteration—ideally just a few days.

When a story is too large, it is called an epic. Backlog items tend to start as epics when they are lower priority. For release planning, epics should be broken down into smaller chunks, but not so small that you have moved into detailed design.

A theme is a collection of related user stories. Let’s say I write a group of stories about monthly reporting that all should do with formatting. By giving those stories a theme, let’s say “Monthly Report Formatting” we put a virtual (and sometimes literal) rubber band around the stories and can refer to them as a group.  
  
How Detailed Should a User Story Be?  
***Too broad***

* A team member can view iteration status.

***Too detailed***

* A team member can view a table of stories with rank, name, size, package, owner, and status.
* A team member can click a red button to expand the table to include detail, which lists all the tasks, with rank, name, estimate, owner, status.

***Just right***

* A team member can view the iteration's stories and their status with main fields.
* A team member can view the current burndown chart on the status page, and can click it for a larger view.
* A team member can view or hide the tasks under the stories.
* A team member can edit a task from the iteration status page.

A Meaningful Agile User Stories

1. Key stakeholders must achieve a shared understanding of the deliverable
2. Stories should convey the opportunity, issue, need or value that it will deliver
3. The story title should be short and convey the deliverable without reading the details
4. Good stories deliver an atomic increment in business
5. Stories need to be completed in a Sprint
6. Good stories have sufficient acceptance criteria
7. The team should be able to estimate the story

Who Uses User Stories?  
***Creation***: The customer, customer proxy, product owner, and anyone else who identifies a need for the product can contribute user stories.  
***Ownership and maintenance***: The product owner owns the user stories and is responsible for writing, gathering, maintaining, and prioritizing.  
***Usage***: Developers, testers, technical writers use user stories to be able to know what to implement and when they are done. Product owners track overall progress based on the status of the user stories. Management tends to track user stories rolled up to epics or features.

What Are the Top Mistakes That People Make?

1. Too formal or too much detail. Product owners with good intentions often try to write extremely detailed user stories. If a team sees a story at iteration planning that looks like an IEEE requirements document, they often assume that all the details are there and will skip the detailed conversation.
2. Technical tasks masquerading as stories. Much of the power of Agile comes from having a working increment of software at the end of each iteration. If your stories are really just technical tasks, you often do not end up with working software at the end of each iteration, and you lose flexibility in prioritization.
3. Skipping the conversation. Stories are intentionally vague before iteration planning. If you skip the acceptance criteria conversation, you risk moving in the wrong direction, missing edge cases, or overlooking customer needs.

User story grooming session

It is highly recommended to go through the user stories with a group of stakeholders and some of the team members. It helps to describe what’s needed for the item to be ready for development and to which priority. Sprints are meant to allow you to deliver finished parts of the product. As simple as they may seem, it requires a proper planning, and there is a need to have perfect input and to specify acceptance criteria.

***How to run grooming session for the user stories.***

1. Send a recursive invitation for grooming session. Depending on your sprint duration, if you are having two long sprints then it’s ideal to have grooming session every month or every two weeks, while sprint is in the middle.
2. Prioritize the backlog as good as you can, ask the participants to go through the user stories before coming to meeting, so there can be a detail level of discussion
3. You can invite people from technical team, not all members need to be there, but some senior members, architect, or someone with good knowledge about the user story in question should be invited, then there should be people from business, sales or stakeholders, and the internal customer who requested those user stories.
4. It is important to run the meeting as timebox for 1hr, or as biweekly shorter meeting. It is not a good idea to spend 3 hours for grooming session, as it is not very productive.
5. Go through the user stories in detail, try to finalize the open questions, perfect your mockups, and describe and verify your user flow.
6. If entertainment has been budgeted for, order for lunch or healthy snack.
7. Take meeting notes and write clear action points like what to be updated and by who etc.

Prioritizing User Stories

A survey result conducted by an important international organization, states that 64% of the functionalities included in software products are never or almost never used! ***How come?!*** One way to avoid such a waste of resources, is to assign priorities to the user stories and to develop them starting from the highest.

The Product Owner (PO) plays an important role in Scrum practice, and one of its duties is prioritizing the features/user stories. Once the requirements gathering phase is finished, the product owner should have in hand a list of index cards in which are written the user stories. Prioritizing the user story is necessary to:

* Maximize and accelerate the ROI
* Reduce risks

To maximize the ROI, the PO must have a clear understanding of what the customer wants in terms of: ***objectives, necessities, and opportunities* to reach**. In one word, she must understand what is the actual Customer Business Value to gather.

Reducing risks, on the other hand, is primarily a matter of:

* Removing impediments
* Giving answers and explanations to any doubts or open points
* Deepening user stories that are not clear
* Executing any critical project activities soon enough.

There are many methods for setting user story priorities, but one to consider is that which takes care of the customer business value and the risks associated with each feature. This is done by assigning to the features an index (e.g. 1 to 10) so that it is easy to draw a point in a x/y graph.

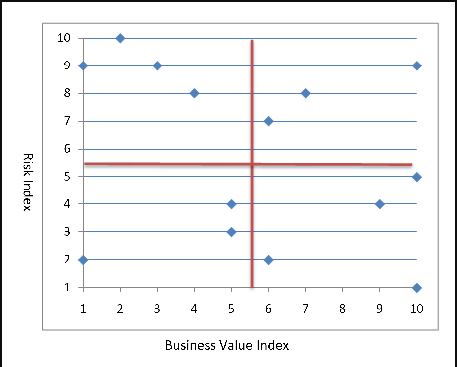
Graph of Customer Business value against Risk can be achieved by using an Excel sheet:

1. Create a new worksheet containing three columns: User story ID (or title), Business Value Index, Risk Index.
2. List all the user stories and for each one, first assign a business value index: 1 for the lowest and 10 for the highest.
3. Higher values should be assigned to each activity that directly aims to one or more of the objectives to be achieved.
4. Lower values should be assigned to the activities not so important in achieving the goals like ancillary activities, maintenance or administrative activities, reports, etc.

Once a business value index is assigned for each feature, is time to provide risk indexes. See the table below:

|  |  |  |
| --- | --- | --- |
| Features ID | Business Value Index | Risk Index |
| ID1 | 10 | 1 |
| ID2 | 6 | 2 |
| ID3 | 1 | 2 |
| ID4 | 7 | 8 |
| ID5 | 6 | 7 |
| ID6 | 5 | 3 |
| ID7 | 1 | 9 |
| ID8 | 5 | 4 |
| ID9 | 3 | 9 |
| ID10 | 4 | 8 |
| ID11 | 2 | 10 |
| ID12 | 10 | 9 |
| ID13 | 9 | 4 |
| ID14 | 10 | 5 |

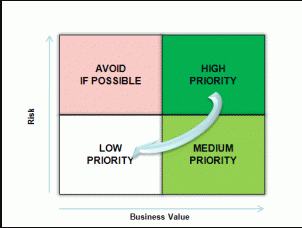
Now use excel to create a graph having all the points: see the image below



Consider the table below; which has Risk on the vertical axis and Business value on the horizontal axis. It will serve as a reference for estimating and prioritizing the user story.

|  |  |  |
| --- | --- | --- |
| Risk | High Risk  Low Value | High Risk  High Value |
| Low Risk  Low Value | Low Risk  High Value |
| Business Value | |

Using the reference table on the graph obtained, we can visually represent the priorities of the user story, hence we can choose which to commence with, which to avoid, which is low priority, and which is low priority.



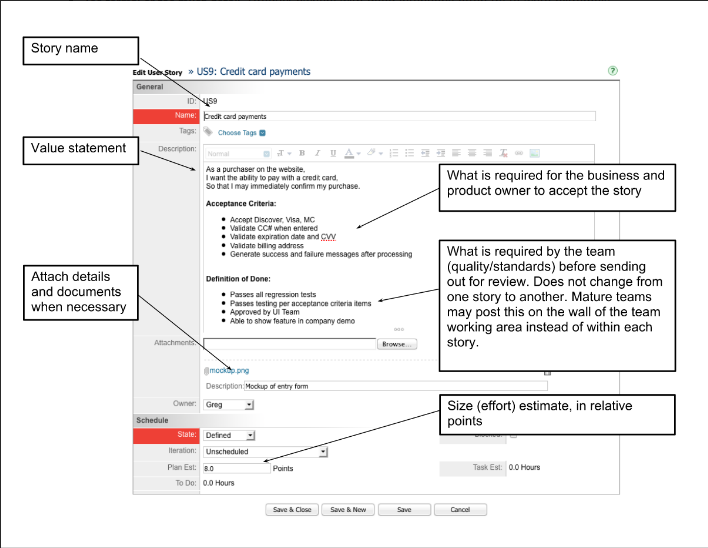
Writing a User Story

User story is an agile technique to capture product functionality. Working with user stories is easy, but telling effective stories can be hard. The following ten tips help to write a good user story:

1. Users come first
2. Use personas to discover the right stories: personas are fictional characters that are based on first-hand knowledge of the target group. They usually consist of a name and a picture; relevant characteristics, behaviors, and attitudes; and a goal, which is the benefit the persona wants to achieve.
3. Create stories collaboratively
4. Keep stories simple and concise
5. Start with Epics
6. Refine the stories until they are ready
7. Add acceptance criteria
8. Use Paper Cards: this approach provides three benefits: First, paper cards are cheap and easy to use. Second, they facilitate collaboration: everyone can take a card and jot down an idea. Third, cards can be easily grouped on the table or wall to check for consistency and completeness and to visualize dependencies. Even if the stories are stored electronically, it is worthwhile to use paper cards when writing new stories
9. Keep stories visible and accessible
10. Don’t solely rely on user stories: Creating a great user experience (UX) requires more than user stories. User stories are helpful to capture product functionality, but they are not well suited to describe the user journeys and the visual design. Therefore, it is necessary to complement user stories with other techniques, such as, story maps, workflow diagrams, storyboards, sketches, and mockups.

User Story Format

User stories are short, simple descriptions of a feature told from the perspective of the person who desires the new capability, usually a user or customer of the system. The figure below describes a simple format of a user story



**Summary**

In this article, we have covered the importance and benefit of Agile development methodology, Scrum practice and user story. This article is meant to improve your teams understanding of an effective user story, and their ability to develop and build systems using scrum practice in Agile methodology. Following the steps and guides outlined in this article will help in planning, scheduling, and tracking of project tasks, thus leading to an effective and efficient software development, and a quick product delivery. You must have in mind that the measure of project success is a ship ready product at each phase/iteration in the development cycle.

References

1. <http://www.emilianosoldipmp.info/2011/07/prioritizing-user-stories/>
2. <https://help.rallydev.com/writing-great-user-story>
3. <https://www.benlinders.com/2017/how-to-write-effective-user-stories/>
4. <https://blog.taiga.io/user-stories-demystified.html>
5. <https://michaellant.com/2010/05/21/how-to-easily-prioritize-your-agile-stories/>