Scrum, User Stories, and More!

CSCI 5828: Foundations of Software Engineering
Lecture 22 — 11/06/2014

Goals

- Cover Material from our User Stories Book
  - Chapter 15: Using Stories With Scrum
  - Chapter 16: Additional Topics with Respect to User Stories

Using Stories with Scrum

- User Stories is a practice / requirements technique that originated with the agile life cycle known as Extreme Programming
    - Because it plays well with the principles and values of Agile life cycles, they can be used in other life cycles
- One very popular agile life cycle is Scrum
    - In this chapter, Mike Cohn explores how user stories can be integrated into Scrum

Scrum

- Agile life cycles like Scrum and Extreme Programming have two important properties; they are **iterative** and **incremental**
    - **iterative life cycles**: makes progress on a system through successive refinement
        - A team takes a first pass at a system knowing it is incomplete in many different areas
            - In subsequent iterations, detail is added until each aspect of the system is completely functional
- **incremental life cycles**: delivers a working system in increments
    - each increment is a complete subset of functionality
    - each increment is fully coded and tested

Scrum Basics

- Scrum projects are organized around 30-day iterations called **sprints**
  - At the start of each sprint, the team determines the amount of work they can accomplish during that sprint
  - Work is selected from a prioritized list called the **product backlog**
  - The work scheduled for a sprint is placed on the **sprint backlog**
  - Each day, a **daily scrum** is held; a brief status meeting that allows the team to inspect its progress and adapt its work as necessary

Scrum Team

- A Scrum team typically consists of four to seven **developers**
    - like all Agile projects, developers are expected to work on all aspects of the system under development
- Scrum teams are augmented by two additional people
    - **Product owner**: A product owner plays the role of the customer; the product owner's job is responsible for adding jobs to the product backlog and prioritizing the items on that list
    - **Scrum Master**: A Scrum master plays the role of project manager; however, the key responsibility of the Scrum master is one of leadership than of managing
        - the Scrum master serves the team rather than directs it: removing obstacles and helping the team observe Scrum's rules and practices

Product Backlog

- The product backlog is a master list of all desired functionality for the system under development
  - At the start of the project, the product owner and the team work together to generate a list of “obvious” work items
    - typically, it is easy to identify more than enough work to get the first sprint started
- The product backlog is then allowed to grow and change as more is learned about the product and its customers
  - Work items can be technical tasks as well as more user-centric statements of functionality

Sprint Planning Meeting (I)

- A **sprint planning meeting** is held at the start of each sprint
  - The meeting lasts a full day and is attended by the product owner, scrum master, developers, and any other relevant stakeholders
- During the first half of the meeting, the product owner describes the highest priority features that are remaining on the product backlog; the team will ask questions about these items to make sure they understand them
  - During the second half of the meeting, the team works alone to decide what items will be worked on during the sprint
- The team and the product owner define a **sprint goal**, a short description of what the team plans to achieve
  - at the end of the sprint, the team will judge their success against the sprint goal rather than any particular set of items on the product backlog

Sprint Planning Meeting (II)

- Once the sprint starts, **only the team may add work** to the sprint
  - The product owner cannot change their mind; the company CEO cannot make requests for additional features, etc.
- The team commits to finish the work they selected; the company commits to letting them get that done
  - if they get ahead, the team can work with the product owner to add a few more items to the current sprint
- If for some reason, the company cannot honor their commitment, the sprint ends abruptly and the team starts again with a new sprint planning meeting

Sprint Planning Meeting (III)

- When an item is transferred from the product backlog to the sprint backlog, the team works to expand the item (if necessary) into a set of tasks
  - A single item from the product backlog may expand into one or more items on the sprint backlog, allowing the team to more effectively share the work
    - This is similar to how user stories get expanded into tasks at the start of an iteration

Sprint Review Meeting (I)

- Each sprint is required to deliver a potentially shippable product increment
  - At the end of 30 days, the team has produced a working, tested, and usable software prototype
  - The prototype must be potentially shippable, allowing the company the option of releasing it to customers or using it internally
- At the end of each sprint, a **sprint review meeting** is held
  - During this meeting, the team demonstrates the software prototype
    - There are rules that forbid the use of slides and restricting prep time for this meeting to two hours
  - This meeting is not meant to be a distraction but rather the natural result of the work of the sprint

Sprint Review Meeting (II)

- The entire team (developers, product owner, Scrum master) is expected to attend this meeting
    - Other stakeholders may attend if they are interested
- During this meeting, the current state of the project is assessed against the sprint goal;
    - ideally the sprint backlog was reduced to zero
        - BUT, it is considered more important to meet the goal of the sprint, even if one or two features were not completed

Daily Scrum Meeting (I)

- Scrum was the first documented agile life cycle to include a short daily meeting; the technique quickly spread to other agile life cycles
- The daily scrum is typically held as early as possible after the entire team has arrived to work (typically 9 AM)
  - Everyone is required to attend (devs, product owner, Scrum master)
- The scrum meeting is short: typically 15 minutes; no longer than 30 minutes
  - To keep it short, most teams require it to be a standing meeting

Daily Scrum Meeting (II)

- During the daily scrum, each team member (including the product owner) answers three questions
  - What did you do yesterday?
  - What will you do today?
  - What obstacles are in your way?
- It is important that this meeting stick to presenting the answers to these three questions; the scrum is not used to resolve issues it is to identify them!
  - It is also used to get developers to commit to work that will get done that day; the team commits to each other, not the manager or the company
- Important issues that need resolution can be handled outside the scrum via small group meetings that will naturally occur throughout the day

Daily Scrum Meeting (III)

- The daily scrum is a “heartbeat activity” designed to
  - give everyone on the team an overview of where the project is at
  - to allow more senior management to get progress reports
    - these managers are not allowed to ask questions or redirect work!
  - to allow people to adjust to respond to problems/obstacles that have come up
    - someone who is ahead on their work can agree to work with someone who is working on a task that is taking longer than expected
- It is not used to update estimates; that can be done on a whiteboard or via software during the day

The Rules of Scrum (I)

- A sprint planning meeting is held at the start of each sprint
  - Each sprint must deliver working and fully tested valuable code
- The product owner prioritizes the product backlog
  - The product backlog may grow or reprioritized at any time
- The team collectively selects the amount of work brought into the sprint
  - Once a sprint begins, only the team can add work to the sprint backlog
- A short scrum meeting is held every day so the team can answer the three questions
  - Only the team members may speak during the daily scrum

The Rules of Scrum (II)

- The result of a sprint is demonstrated at the end of the 30 days at a sprint review meeting
  - Working software is demonstrated during the review; slides are not allowed
  - No more than two hours may be spent preparing for the review

Adding User Stories to Scrum (I)

- As should be expected, to add users stories to Scrum, you add a rule that states that the product backlog can consist only of user stories
    - Now that each item in the product backlog expresses functionality that is valuable to the product owner, it becomes a lot easier for the product owner to assign priorities
- At the start of the project, filling the backlog can make use of all the techniques associated with user stories
    - Define user roles
    - Use these roles to generate stories for each role
    - If there's functionality that you know is needed but you don't know a lot about it, write it as an epic and prioritize it for later

Adding User Stories to Scrum (II)

- Stories in the sprint planning meeting
  - Each story in the product backlog is designed to deliver value to the user
    - We now decide which stories go on the sprint backlog and expand them into tasks as usual
- We also avoid any problems with tasks being on the product backlog
  - Tasks are development centric work items that might contain terminology unfamiliar to the product owner
    - Why waste time explaining what a task means to the product owner, when instead you can keep that person focused on desired functionality and work out tasks and task assignments on the sprint backlog

Adding User Stories to Scrum (III)

- Stories in the sprint review meeting
  - With stories, it is much easier for the team to demonstrate what functionality made it into the software prototype
    - The story may have been decomposed into 10 tasks but if all ten are completed then the team can show that the feature it described is now present in the system
- Stories in the daily scrum
  - Stories can help developers see the “big picture” when they are “down in the weeds” in the daily scrum
    - Working on tasks gain new meaning since they can be associated back to particular story and thus valuable functionality
    - Obstacles to be handled can be prioritized in terms of the stories they are blocking

Additional Topics

- Chapter 16 highlights a few additional topics related to user stories
  - handling nonfunctional requirements
  - paper or software?
  - user stories and user interfaces
  - should stories be kept after they are completed?
  - user stories and bug reports

Nonfunctional Requirements

- Teams new to working with user stories will often feel that EVERYTHING about a project should be specified with user stories
  - Heads down a wrong path; stories are meant for functional requirements
- Typically, teams will encounter constraints or goals that do no relate to functionality but to nonfunctional concerns
  - performance, portability, scalability, accuracy, reusability, maintainability, interoperability, availability, usability, security, capacity
- These types of concerns can be handled with an index card that is tagged with the word “Constraint”; the card then simply states the constraint
  - All queries will display their first result in less than one second
- Constraint cards are NEVER assigned to an iteration; they are goals to be achieved; sometimes these goals can be tracked with tests
  - Take a hit on velocity during an iteration to have space to write those tests

Paper or Software?

- Should user stories be written on index cards or entered into a software system
  - The book presents reasons for either choice
- Paper
  - Constant reminder that stories are imprecise
  - Encourage interaction during planning sessions
  - Constrains amount of text that can be associated with a story
- Software
  - Easier to track and sort stories
  - Enables requirements traceability (if required for the project)
  - Helps teams that are distributed across multiple sites

User Stories and User Interfaces

- The author acknowledges that agile life cycles are not oriented to designing applications where the user interface is the primary concern
    - Agile life cycles are meant for iterative refinement and with user interfaces iterative refinement leads to small or large changes in the UI by the end of each iteration
        - The problem? Users hate it when the UI changes on them!
- To address this concern, the author describes a way to include paper prototypes into the process associating them with groups of user stories
    - You add steps in which the paper prototypes are reviewed by the customer to get feedback and to include them in the process where changes to the UI are considered
        - You then code up the stories as normal

Retaining Completed Stories

- One of the tenants of Extreme Programming was that completed user stories were put into the shredder
  - The idea was that we don't allow our selves to become weighed down with excess documents
    - if we're done with the document, get rid of it

- However, the author recommends retaining completed stories
  - they provide a history of how the project evolved
  - they allow that history to be conveyed to outsiders
  - if you are asked about "the requirements of the system", you can provide them a union of all user stories as the response!

Bug Reports and User Stories

- A very short section in the book because
  - Each bug report can be treated as a user story
    - Fixing the bug will provide value to the user, so it counts!
  - If there are a lot of small bugs, they can all be combined into a single story and assigned as usual

Summary

- In this lecture, we
  - learned about Scrum, one of the most popular Agile life cycles out there
  - learned how user stories can (very easily) be added to Scrum
    - and the advantages of doing so
  - addressed a variety of other topics related to user stories
    - non-functional requirements
    - should stories be written on paper or stored in software?
    - how can user stories be used to create user interfaces
    - should stories be kept after they are completed?
    - user stories and bug reports

Coming Up Next

- Lecture 23: The Design of Design, Chapter 6-16
- Lecture 24: The Design of Design, Chapter 6-16