CS314 Software Engineering
User Stories

Dave Matthews

User Stories in Scrum

https://www.scrum.org/
User Story

A brief statement of intent that describes something the system needs to do for the user.

User Story

- As a <role>, I can <activity> so that <business value>
  - <role> represents who performs the activity or receiving the value, even another system.
  - <activity> represents the action to be performed
  - <business value> represents the value achieved by the activity
- Stories appear in the product and sprint backlogs
  - Acceptance criteria are associated with a story
  - Tasks are associated with a story
User Story

• Negotiable expression of intent, not detailed requirement
• Short, easy to read, understandable to all
• Represents small increment of value
• Relatively easy to estimate
• Organized in easily rearranged lists
• Elaborated on as needed, avoids early specificity
• Require little or no maintenance, discarded when done
• Serve as inputs to documentation

Acceptance Criteria

• Functional tests
• Automated whenever possible
• Confirm the story has been implemented correctly
User Story Examples - Sprint 2

• As a customer, I want to provide a list of destinations so you can plan my trip to make the best use of my time.
  – uses geographic coordinates for destinations
  – reorders the destinations for a shorter trip

• As a customer, I want to view my trip itinerary on the web so I know where to go next and how far it is.
  – display a map showing the route
  – display destination information and distance for legs

Tasks

• Smallest unit of work necessary for completion of a story
• Multiple task associated with a single story
• Have an owner and an estimate
• Assures team understands the work to be done and can meet commitments
Splitting User Stories into Tasks

- Workflow steps – implement in incremental stages
- Business rule variations – break into several stories to handle complexity
- Major effort – break into several parts, with most effort in the first part
- Complex/simple – break complex stories down into simpler versions
- Data variations – build the simplest version, add variations and sources
- Data entry methods – build the simplest UI first, richer UI later
- Defer system qualities – do simple first, add other “-ilities” later
- Operations – split large vague operations (manage, control) into multiple
- Use-case scenarios – split complex user/system and system/system interactions
- Break out a spike – gain better technical or functional understanding

User Stories and Tasks in a Sprint

[Diagram showing the relationship between releases, sprints, stories, and tasks.]