User Stories

An Introduction

Big Design Up Front

- Traditional requirements analysis:
  - Talk to stakeholders (customer, end users, etc.)
  - Think about the planned system; develop UML models
  - Maybe prototype a bit
  - Spend months developing a big document covering every requirement of the system up front
  - Give said document to developers
  - Receive word from developers that the project will actually take 24 months instead of the desired 6 months

Big Design Up Front: Problems

- Very difficult to envision every possible feature needed up front
- Process of documenting every requirement is tedious and error-prone
- Customers often change their minds or come up with new ideas as they see the software being developed
- Requirements documents are long and boring
  - Greater chance they will simply be skimmed or entire sections will be skipped
  - Hard to grasp the big picture behind a 300-page requirements specification

Big Design Up Front: Problems

- Time wasted writing 3/4 of the requirements that the team won't be able to complete in the allotted 6 months
- More time wasted as the development team decides which requirements it can implement in time
- Levels of indirection between customers and developers
  - Lack of direct communication leads to misinterpretation of the intended functionality
  - Remember the telephone game?

Big Design Up Front: Problems

- Focusing on a checklist of requirements rather than on the user's goals does not necessarily lead to a good overall understanding of a product:

3.4) The product shall have a gasoline-powered engine
3.5) The product shall have a four wheels
    3.5.1) The product shall have a rubber tire mounted to each wheel
3.6) The product shall have a spinning blade mounted on its underside
3.7) The product shall have a foam seat

vs.

- The product makes it fast and easy for me to mow my lawn
- I am comfortable while using the product

Big Design Up Front: Problems

- Software is intangible
  - Very hard to estimate reliably
  - Pipe dream: glorious PERT charts enumerating every task that must be completed, the duration of each task, and the order in which the tasks must be completed

Big Design Up Front: Problems

- As software is built, customers often change their minds about existing features and think up new features
- Requirements changes are called a change of scope
  - Implies that the scope of the project was previously fully defined
  - Implies that a project is complete when it fulfills its list of requirements rather than its intended users' goals

Software Requirements = Communication

- Those who want the software must communicate with those who will build it
- A project relies on information from those who view the software from a business perspective, and those who view it from a development perspective
  - If the business side dominates too heavily, it can mandate functionality and deadlines typically with little concern that:
    - The development team can deliver the requested functionality on schedule
    - The development team understands exactly what is being requested

Software Requirements = Communication

- On the other hand, if the development side dominates, it
  - Replaces the language of business (the language of the domain) with technical jargon
  - Loses the opportunity to learn what is needed by listening
- Is this any better? Likely not …

A Solution?

- Given that:
  - Requirements often change throughout a project
  - Reliable estimation is extremely difficult
  - It is difficult to come up with all requirements up front
- What do we do?
  - Make decisions based on the information we have
  - Do it often
- Spread decision-making across the duration of the project: we need a process that gets us information as early and often as possible

User Stories

- A user story describes functionality valuable to a user/customer
- Three main components:
  - **Card**: A written description of the story used to plan and serve as a reminder
  - **Conversation**: Conversations about the story to flesh out its details
  - **Confirmation**: Tests that convey and document details; confirm to us when the story is complete
- Often written on the front of an index card, with confirmation details written on the back

User Stories: Customer Team

- Stories are usually written by a customer team:
  - Ensure the software will meet the needs of its users
  - Includes developers, testers, product managers, actual users, customers, etc.
- Stories must be written in the language of the user – not in technical jargon
  - Allows the customer team to be able to prioritize stories

User Stories: Examples

- Good examples:

> A user can post his/her resume to the web site
> A user can search for jobs
> A company can post new job listings

- Bad examples:

> The software will be written in C++
> The program will connect to the database through a connection pool

- User's don't care about the programming language used or technical details such as how the application connects to a database

User Stories: Common Templates

- "As a *<role>*, I want *<goal/desire>*."
- "In order to *<receive benefit>* as a *<role>*, I want *<goal/desire>*."
- "As *<who>* *<when>* *<where>*, I *<what>* because *<why>*."
- "As a *<role>*, I can *<action with system>* so that *<external benefit>*."
- "As *<persona>*, I want *<what?>* so that *<why?>*."

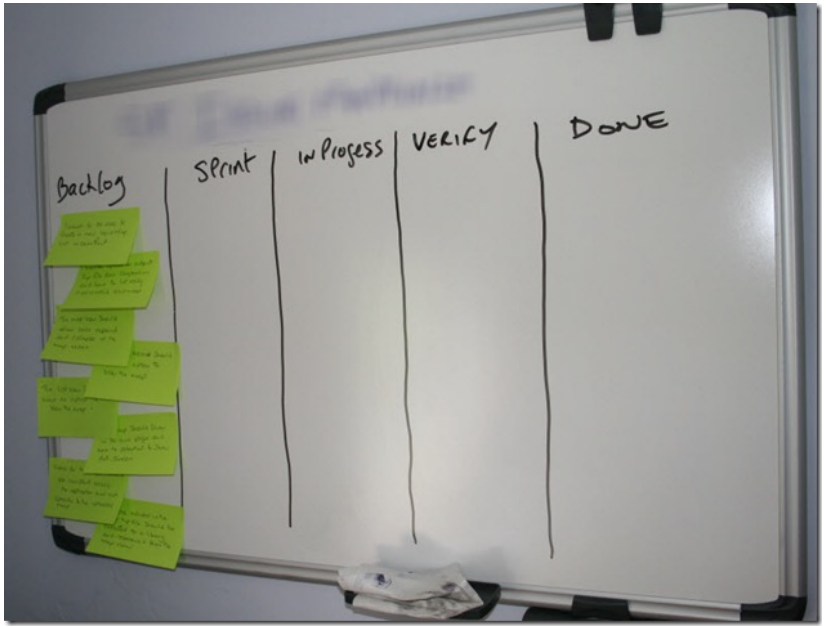
who's using things, what do they want? what they're using thing to do?

User Stories: Conversation

- Notice that many details are missing
  - What fields can the user search for jobs on?
  - Does the user have to be logged in?
- A user story is a reminder to have a conversation
  - We will discuss these details with the customer at the time they become important (just-in-time requirements analysis)
  - The conversation is the important part – not the story itself
- Cards represent customer requirements rather than document them
  - Card contains the story; details worked out in the conversation and recorded in the confirmation

User Stories: People Really Use Them

User Stories: People Really Use Them



User Stories: People Really Use Them