CSE 70: Requirements & Pair Programming

Ingolf Krueger
Learning Goals for Today
Learning Goals

• Understand what requirements are.

• Be able to categorize requirements into functional and non-functional ones.

• Be able to write requirements succinctly yourself.

• Be able to provide initial estimates for given requirements.

• Understand what pair programming means.

• Ability to apply pair programming principles on your project.
What’s a Requirement anyway?
Requirements

- A singular, documented need of what a particular product or service should be or do

- Identifies a necessary attribute, capability, characteristic, or quality of a system in order for it to have value and utility to a user *at a given point in time*

- **Requirements change!**

1http://en.wikipedia.org/wiki/Requirement
How to find Requirements?

• Communicate with your customer/users!
• Know your stakeholders!
  – Customer
  – Users
  – Architects
  – Developers
  – QA
  – Maintainers
  – ...
• Agile:
  – user stories on index cards!
• Plan Driven:
  – Requirements documents (e.g., IEEE Std 830-1998)
Ideal Requirements – rarely found this way

- Necessary
- Correct
- Unambiguous
- Complete
- Testable: is each requirement satisfied in the final system?
- Consistent: requirements do not contradict each other
- Modifiable: can update requirements easily.
- Traceable:
  - know why each requirement exists;
  - go from source to requirements;
  - go from requirement to implementation;
  - back from implementation to requirement.
Reality: Iterative Process!

1. Ask Stakeholder
2. Observe/Shadow Stakeholder
3. Brainstorm "Bluesky"

- Write down Requirements
- Estimate/Prioritize
- Plan Iteration & Implement
Functional vs. Non-Functional Requirements

- **Functional**
  - What the system is supposed to do

- **Non-functional**
  - Constraints at the implementation of the functional requirements
    - **Product**
      - Usability
      - Efficiency (Space, Time)
      - Reliability
      - Portability
    - **Organization**
      - Delivery
      - Implementation
      - Standards
    - **External**
      - Interoperability
      - Ethics
      - Legislation (Privacy, Safety)
Example: **User Stories**
(Unit of Customer-Visible Functionality)
Title: Set up conference

Description: User marks a set of other users in their buddy list, then issues the "setup conference" command, then enters the name of the conference and optionally a description/purpose.
Properties of User Stories

• Written from the user’s perspective

• Describe one thing the user can do with/in the system

• Written in Language the user can understand (avoid computer jargon)

• Avoid mentioning how it is implemented

• Be brief!

• Cover a few hours to a few days worth of work.

• Tests should jump out at you immediately.
Your turn!
Team up for 5 minutes!

Write up as many user stories about the chat system as you have index cards available in your team!
Estimation
Estimation

- Happens among the Software Engineers (customer could interfere with/pressure team)
- Write estimate on top of user story index card
- Add assumptions you are making(!)
- Have every team member give their estimate
  - Jointly remove assumptions
  - Jointly remove outliers
- Include:
  - Learning/research that needs to be done prior to implementing
  - Infrastructure setup
- Large estimate (more than 15 days)
  - Split story
Example: Estimation
Title: Set up conference

Description: User marks a set of other users in their buddy list, then issues the "setup conference" command, then enters the name of the conference and optionally a description/purpose.
Split
Split User Story

**Title:** Mark buddies

**Description:** User marks a set of other users in their buddy list.

**Title:** Issue setup command

**Description:** User issues “setup conference” command
Title: Spec conf parameters

Description: User enters the name of the conference and optionally a description/purpose.

Title: Set up conference (combo)

Description: User marks a set of other users in their buddy list, then issues the “setup conference” command, then enters the name of the conference and optionally a description/purpose.
Your turn!
Team up for 5 minutes!

Estimate your stories among your team members!
From Requirements to Testing
User Stories/Tasks turn into Acceptance/Unit Tests!

- Write a test for every user story.

- Break down user stories into tasks (“mini user stories”, ~hours to a day’s worth of work)

- Write tests for every task.

- Can’t find a test for a user story/task?
  - Go back to analyzing the problem
  - Split user story
  - Let the customer give you the test
From Requirements to Pair Programming
Laurie Williams (NCSU) on Pair Programming

"Pair programming is a simple, straightforward concept. Two programmers work side-by-side at one computer, continuously collaborating on the same design, algorithm, code, and test. It allows two people to produce a higher quality of code than that produced by the summation of their solitary efforts."

• Roles:
  – Driver: types or writes
  – Navigator: observer (looking for tactical & strategic defects)

• Periodically switch roles of driver and navigator
• Pair coding, design, debugging, testing, etc.

©L. Williams
Watch the movie
What have you learned today?
Learning Goals

• Understand what requirements are.

• Be able to categorize requirements into functional and non-functional ones.

• Be able to write requirements succinctly yourself.

• Be able to provide initial estimates for given requirements.

• Understand what pair programming means.

• Ability to apply pair programming principles on your project.