Today

• Requirements (cont.)

Homework

• Proj 5 out
• HW 3 out
Qualities of “good” requirement statements

Q: What do you think are qualities of “good” requirements?

**Complete**
- fully describes feature (function) or characteristic
- all info needed to start designing and/or implementing

**Correct**
- accurately describe the feature or characteristic ★ important!

**Feasible**
- possible to implement (with known technologies)

**Necessary**
- the customer really needs it (not just “nice to have”)
- good to trace back to user ★ important!

**Prioritized**
- relative ranking of needed features/characteristics
- more later ★ important!
Unambiguous

- all readers should agree — a single, consistent interpretation

Verifiable

- can determine if software satisfies requirement
- often realized as “acceptance” tests

Example of acceptance tests & priorities in user stories

Title: See your transaction history

Description: As a ...

Importance: 10

How to demo:
- login and select "transactions"
- make a deposit
- go back to transactions
- check that new deposit added

- 10 means very important
- acceptance tests often on back of user story
- we’ll discuss these more later

Focused

- about one feature
- not overly general ... specifies a concrete requirement
Qualities of “good” requirement specifications

Q: What do you think are qualities of good specifications?

Complete

- no missing requirements
- car example ...

Consistent ★ important!

- no conflicting requirements

Modifiable (bec. they will change!)

- give them unique names/labels
- best to pick meaningful names
- some use hierarchical numbering (R1.1.1, R2.2.1 ... why?)
- or a mix (ED-1.1, ED-2.1, where ED = “Editor Functions”)

Traceable

- each requirement linked back to the source (user)

In general, specifications will not have all these qualities

Similarly, with the qualities for individual statements
Some “Best Practices”

Create a **project glossary**
- define the specialized terms from the domain (with synonyms)
- try to reduce ambiguity and confusion
- can help you “say less” in requirement statements

Identify **user roles**
- develop requirements to specific users (avoids feature creep)
- can help with prioritization (scope issues)
- e.g., supporting some roles may be less important than others

“**Timebox**” discussions
- limit length of meetings to 30 min (max 1 hour)
- better to collect requirements in short bursts
- ... than over long drawn-out meetings

Q: Why?
- gives people a chance to think things over (when not meeting)
- focus on important stuff because of time cutoff
- engages people more (few like long meetings)

- similarly, limit the number of “cooks in the kitchen”
- instead, have more meetings with fewer and different people
Draw diagrams

- e.g., context diagram, flow diagram, use-case diagrams, UI mock-ups/storyboard
- diagrams can help communicate system needs
- diagrams can help form a shared understanding with users
- diagrams can help make ambiguous requirements more concrete