Lecture 5: Requirements I

Lecture goals
1. Document verifiable requirements
2. Elicit requirements from stakeholders

Purpose of requirements
- What should a product do?
- What should a product not do?
- How is a product constrained?
- Take client's perspective
  - Meeting requirements should provide meaningful visibility
  - Not about design – "what", not "how"
- How should a product be tested?
- Risks of insufficient requirements documentation
  - Client dissatisfaction
  - Late discovery/rework
  - Poor design tradeoffs
- Code is not a specification

Top reasons for project failure
- See *The Chaos Report (1994)* from The Standish Group
- 70% of projects failed because developers built the wrong system (reasons related to requirements and client interactions)

Requirements engineering steps
1. Analysis: Establish functionality in consultation with stakeholders
2. Modeling: Organize requirements systematically
3. Definition: Record and communicate precise requirements

Heavyweight vs. Lightweight
- Heavyweight
  - Gather most requirements upfront
o Document requirements formally

- Lightweight
  o Start with system-level requirements
  o Expand and refine requirements iteratively (e.g. for each sprint)
  o Continual client interaction
  o Requirement still exist and should still be documented

**Types of requirements**

- Functional
  o What a product should do
  o What a product should not do
  o Can be verified locally

- Non-functional
  o Aka "quality requirements"
  o Property of system as a whole

- Constraints
  o Limits how the system can be built

**Validation & verification**

- Validation
  o "Are you building the right thing?"
    ▪ Would a system satisfying all of the requirements (and nothing else) meet the business need?
    ▪ Are assumptions in models consistent with reality?
  o Involve client
    ▪ User testing
    ▪ Acceptance testing

- Verification
  o "Did you build it right?"
    ▪ Implementations should be verified against requirements
    ▪ Design can be verified by analysis
    ▪ Process can be verified by audits

- Testing
  ▪ Can define pass/fail criteria based on previous step
Requirements definition

- Audience: Client AND developers
- CS 5150: Use future report/presentation to validate requirements with client
  - "Our understanding of your requirements is that ..."

Writing good requirements

- Must be verifiable
  - Can it be measured?
    - Use proxy measurements if needed
  - Are tolerances specified?
  - Can you design a test?
    - Include pass/fail criteria
  - Is it feasible? (to implement AND to verify)
- Must relate to client-relevant behavior
- Use consistent wording
  - "Shall"
  - "Should" if there are exceptions
  - Consistent names for actors, interactions, events
- Use appropriate format
  - Flow chart, decision table, ...
- Provide rationale
  - Link to requirements being derived from or depended on

Tracking and tracing

- Objective: facilitate verification, validation, revision
- Complete list
- Unique identifier
- Organized, cross-linked
- Linked to verification activities
  - Separate document (e.g. verification matrix)
- Change review procedure
Analysis

- Check for completeness, consistency

Stakeholder and viewpoint analysis

- Identify who is affected by the system
  - Client
  - Customers
  - Users (many categories)
  - Administrators
  - Maintainers
- Effort often not proportional to utilization
  - E.g. administrative capabilities are often much larger than user capabilities

Eliciting requirements: interviews

- Difficult, but essential
- Tips:
  - Allow plenty of time
  - Prepare before meeting client
  - Keep full notes
  - Clarify what you do not understand
    - Define domain-specific terminology
  - Repeat what you hear
- Consider all stakeholders
- Ask questions
  - "Why do you do things this way?"
  - "Is this essential?"
    - Be wary – impact may not be obvious
  - "What are the alternatives?"

Negotiation and prioritization

- Conflicts, and difficulties affecting cost and schedule, must be resolved with client
  - Help client understand the tradeoffs
o Be open to suggestions
  • Incremental delivery (e.g. Agile sprints) encourages regular prioritization

To be continued...