CIS 422/522

Software Requirements

Project Status Reports

What is a "software requirement?"

- A description of something the software must do or property it must have
- The set of system requirements denote the problem to be solved and any constraints on the solution
  - Ideally, requirements specify precisely what the software must do without describing how to do it
  - Any system that meets requirements should be an acceptable implementation
- Anything the customer wants(?)
  - Common in industry
  - What might be the issues?

Requirements Review

- Important to get the requirements right
- Only three goals
  1. Understand precisely what is required of the software
  2. Communicate that understanding to all of the parties involved in the development (stakeholders)
  3. Control production to ensure the final system satisfies the requirements
- All of the goals can be difficult to accomplish in practice

What makes requirements difficult?

- Comprehension (understanding)
  - People don't (really) know what they want (…until they see it)
  - Superficial grasp is insufficient to build correct software
- Communication
  - People work best with regular structures, conceptual coherence, and visualization
  - Software's conceptual structures are complex, arbitrary, and difficult to visualize
- Control (predictability, manageability)
  - Difficult to predict which requirements will be hard to meet
  - Requirements change all the time
  - Together can make planning unreliable, cost and schedule unpredictable
- Inseparable Concerns
  - Many requirements issues cannot be cleanly separated (I.e., decisions about one necessarily impact another)
  - Difficult to apply "divide and conquer"
  - Must make tradeoffs where requirements conflict

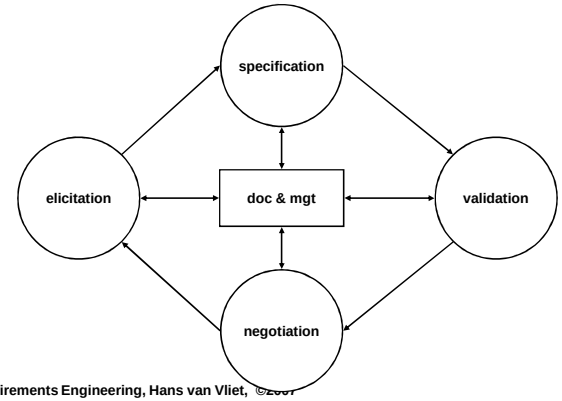
Requirements Process

A Requirements Process Framework

- Managing requirements difficulties requires having a good process

1. Requirements Understanding
   1. Requirements Elicitation
      - How do we establish "what people want?"
   2. Requirements Negotiation
      - How do we resolve stakeholder conflicts?
2. Requirements Specification
   1. Concept of Operations
      - How do we communicate with non-technical audiences?
   2. Software Requirements Specification
      - How do we specify precisely what the software must do?
3. Requirements Validation (and Verification)
   - How do we establish that we have the right requirements (feedback)
   - How do we ensure our specification is good quality?

RE Process from Text: Same Idea



SE, Requirements Engineering, Hans van Vliet, ©2007

Requirements Elicitation

- Addresses the general question: "What do the stakeholders want?"
- Inherently open-ended, ambiguous question
- Addressed by a number of elicitation methods
  - Interview – traditional standard
  - Domain analysis and modeling
  - Focus groups
  - Prototyping
  - Scenario analysis (current favorite), etc.
- All have differing costs, strengths, and weaknesses. None is a complete solution.
- Implications
  - Use more than one approach
  - Check the results *early and often*

Example: Elicitation by Interview

- Interview techniques: basically ask the user what he/she wants or expects of the software
- What are the limitations and potential issues?
  - Does an individual user know what he/she wants?
  - Is a person's understanding of the problem likely to be complete?
  - Is a person's understanding and desire likely to stay constant over time?
  - Will different people have different understandings?
  - Will everyone be equally able to express their views?
  - Will the interviewer have the same understanding of the domain of discourse as the interviewee?
  - Will the needs of all of the stakeholders have equal importance?
  - Will we always know who all the stakeholders are?
- What does this say about the limitations of interview techniques?

Requirements Negotiation

- or "Why the customer is not always right."
- Stakeholders' requirements often conflict
  - Needs of different customers/users may conflict
    - E.g., Salesmen want convenience and speed, management wants security and accountability
  - Developer's needs may conflict with customer's
    - E.g., development cost vs. customer wants
  - May conflict within the development organization
- Choosing which requirements should be addressed and their relative importance requires *negotiation and tradeoffs* among stakeholders

Requirements Specification

A Requirements Process Framework

- Managing requirements difficulties is addressed by having a good process

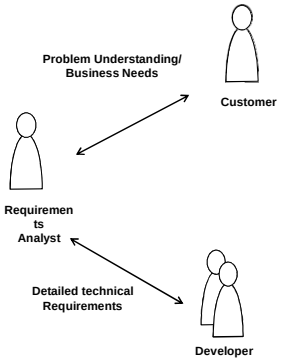
1. Requirements Understanding
   1. Requirements Elicitation
      - How do we establish "what people want?"
   2. Requirements Negotiation
      - How do we resolve stakeholder conflicts?
2. **Requirements Specification**
   1. **Concept of Operations**
      - **How do we communicate with non-technical audiences?**
   2. **Software Requirements Specification**
      - **How do we specify precisely what the software must do?**
3. Requirements Validation (and Verification)
   - How do we establish that we have the right requirements (feedback)
   - How do we ensure our specification is good quality?

Purposes and Stakeholders

- Many potential stakeholders using requirements for different purposes
  - Customers: the requirements typically document what should be delivered and may provide the contractual basis for the development
  - Managers: provides a basis for scheduling and a yardstick for measuring progress
  - Software Designers: provides the "design-to" specification
  - Coders: defines the range of acceptable implementations and is the final authority on the outputs that must be produced
  - Quality Assurance: basis for validation, test planning, and verification
  - Also: potentially Marketing, regulatory agencies, etc.

Needs of Different Audiences

- Customer/User
  - Focus on problem understanding
  - Use language of problem domain
  - Technical if problem space is technical
- Development organization
  - Focus on system/software solutions
  - Use language of solution space (software)
  - Precise and detailed enough to write code, test cases, etc.



Two Kinds of Software Requirements

- Communicate with stakeholders who understand the problem domain but not necessarily programming (solution domain): e.g. customers, users, marketing
  - Do not understand computer languages but may understand technical domain-specific languages
  - Must develop understanding in common languages
- Communicate with developers: sufficiently precise and detailed to code-to, test-to, etc.
  - Stated in the developer's terminology
  - Addresses properties like completeness, consistency, precision, lack of ambiguity
- For businesses, these may be two separate documents

Documentation Approaches

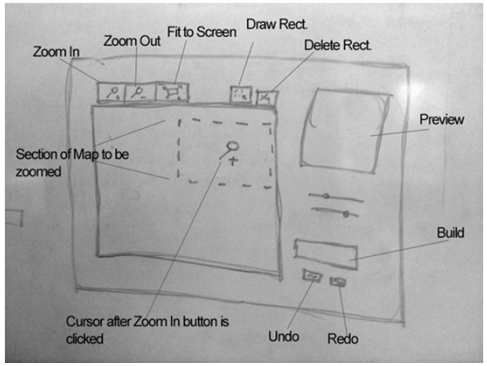
- Informal requirements to describe the system's capabilities from the customer/user point of view
  - Purpose is to answer the questions, "What is the system for?" and "How will the user use it?"
  - Tells a story: "What does this system do for me?"
  - Focus on communication over rigor
- More formal, technical requirements for development team (architect, coders, testers, etc.)
  - Purpose is to answer specific technical questions about the requirements quickly and precisely
    - "What should the system output for this set of inputs?"
    - Reference, not a narrative, does not "tell a story"
  - Goal is to develop requirements that are precise, unambiguous, complete, and consistent
  - Focus on precision and rigor

Informal Specification Techniques

- Most requirements specification methods are informal
  - Natural language specification
  - Use cases
  - Mock-ups (pictures)
  - Story boards
- Benefits
  - Requires little technical expertise to read/write
  - Useful for communicating with a broad audience
  - Useful for capturing intent (e.g., how does the planned system address customer needs, business goals?)
- Drawbacks
  - Inherently ambiguous, imprecise
  - Cannot effectively establish completeness, consistency
- However, can add rigor with standards, templates, etc.

Mock-up Example

Can use simple technology



Use Cases

1 Use Case: Manage Reports

1.1 Description
This Use Case describes operation for Creating, Saving, Deleting, Printing, Exiting and Displaying reports.

1.2 Actors
User
Project database

1.3 Triggers
Program Manager selects operations from menu.

1.4 Flow of events

1.4.1 Basic Flow
1. User chooses desired report by selecting "Report" -> "Open" from the menu bar
2. System displays report to screen
3. User selects desired report layout using Use Case Specify Report
4. Steps 2 and 3 are repeated until user is satisfied
5. User can Save or Print report using use case Save Report or Print Report
6. User Exits report by selecting "Exit" from the "File" menu

1.4.2 Alternative Flows

1.4.2.1 Create New Report
1. User selects "Create New Report" from file menu
2. ...

1.4.2.2 Delete Report
...

1.4.3 Preconditions
etc

A systematic approach to use cases

- **Uses a standard template**
- **Easier to check, read**
- **Still informal**

Technical Specification

The SRS

The role of rigorous specification

Requirements Documentation

- Is a detailed requirements specification necessary?
- How do we know what "correct" means?
  - How do we decide exactly what capabilities the modules should provide?
  - How do we know which test cases to write and how to interpret the results?
  - How do we know when we are done implementing?
  - How do we know if we've built what the customer asked for (may be distinct from "want" or "need")?
  - Etc…
- Correctness is a *relation* between a spec and an implementation (M. Young)
- Implication: until you have a spec, you have no standard for "correctness"

Technical Requirements

- Focus on developing a technical specification
  - Should be straight-forward to determine acceptable inputs and outputs
  - Preferably, can systematically check completeness consistency
- A little rigor in the right places can help a lot
  - Adding formality is not an all-or-none decision
  - Use it where it matters most to start (critical parts, potentially ambiguous parts)
  - Often easier, less time consuming than trying to say the same thing in prose
- E.g. in describing conditions or cases
  - Use predicates (i.e., basic Boolean expressions)
  - Use mathematical expressions
  - Use tables where possible

Formal Specification Example

Type Dictionary

Name	Base Type	Units	Legal Values	Comment
Speed	Integer	Knots	[0, 250]	Speed measured in nautical miles per hour.
Weight	Integer	percent	[0,100]	Weighting for weighted average
time	Integer	seconds	time > 0	Time in seconds.

Monitored Variable Dictionary

Name	Type	Initial Value	Accuracy	Comment
LowResWS1	Speed	0	1	Wind speed reported by first low resolution sensor
LowResWS2	Speed	0	1	Wind speed reported by second low resolution sensor
HighResWS1	Speed	0	2.5	Wind speed reported by first high resolution sensor
HighResWS2	Speed	0	2.5	Wind speed reported by second high resolution sensor

Controlled Variable Dictionary

Name	Type	Initial Value	Accuracy	Comment
TransmWindSpeed	MsgType	ShortMsg	N/A	Transmitted value of wind speed

- SCR formal model
  - Define explicit types
  - Variables monitored or controlled

Formal Specification Example

TransmWindSpeed Event Function

The transmitted wind speed is a moving, weighted average over the length of the history of sensor readings defined as follows:

LW= LowResWeight, HW=HighResWeight, H = History

For *i* be the current count of all sensor readings and v[*i*] the i^{th} sequential value of variable *v* (hence LWS1[i] is the most recent value of LWS1).

TransmWindSpeed Event Function

	Events
	@(TransmitPeriod)
TransmWindSpeed =	(LW* (LWS1[i] + LWS21[i] +… + LWS[i-H]+ LWS[i-H]) + HW *(HWS1[i] + HWS21[i] +… + HWS[i-H]+ HWS[i-H]) / H * (LW + HW)

Requirements Validation

- Feedback-control for requirements
  - Should answer: “Are we building to the right requirements?”
  - Distinct from verification: “Are we building what we specified?”
  - The book is confused on the distinction
    - Checking internal consistency (agreement with itself) is verification
    - Checking external consistency (agreement with the world) is validation
- Requires going back to the stakeholders
  - Earlier is better
- Can use many techniques
  - Review of specifications
  - Prototyping
  - Story-boarding
  - Use case walkthroughs
  - Review software iterations
- Focus on getting to an acceptable product for the key stakeholders

Summary

- Requirements characterize “correct” system behavior
- Being in control of development requires:
  - Getting the right requirements
  - Communicating them to the stakeholders
  - Using them to guide development
- Requirements activities must be incorporated in the project plan
  - Requirements baseline
  - Requirements change management

Assignments for next Thur.

- Respond to reviews
- Read Chap. 11 (Architecture)

Walkthrough

- Consider: What kinds of questions should your documents answer?
  - Assume a manager unfamiliar with the project is reviewing your status
  - Would your documents answer key questions about the project goals and current status?
- Team page: Who is on the team?
- Project plan
  - Who is responsible for which tasks?
  - What are the anticipated risks and what are you doing about them?
  - What is your development process and how does it help address the risks?
  - What is the project schedule of tasks and deliverables?
  - What is the current status relative to schedule?
- ConOps: What capabilities will the software provide the user or customer?
- SRS: What are the detailed technical requirements?
- Architecture: What are the software structures?