CIS 422/522

Software Requirements
& a Little Quality Assurance

Requirements Phase Goals

• What does “getting the requirements right” mean in the system’s development context?
• 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
• Sounds easy but hard to do in practice
• Understanding what makes these goals difficult to accomplish helps us understand how to mitigate the risks

A Requirements Process Framework

• Requirements Understanding
  – Requirements Elicitation - establish “what people want”
  – Requirements Negotiation - resolve stakeholder conflicts
• Requirements Specification
  – Concept of Operations - communicate with non-programming audiences
  – Software Requirements Specification - specify precisely what the software must do
• Requirements Validation and Verification
  – Establish that we have the right requirements (feedback)
  – Ensure our specification is good quality

Communicating with 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.
Documentation Approaches

ConOps:

- 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 the development team
  - Purpose is to answer specific technical questions quickly and precisely
  - E.g. “What should the system output for this set of inputs?”
  - Reference, not a narrative, does not “tell a story”
  - Focus on precision and rigor
  - Goal is requirements that are precise, unambiguous, complete, and consistent

SRS:

- More formal, technical requirements for the development team
  - Purpose is to answer specific technical questions quickly and precisely
  - E.g. “What should the system output for this set of inputs?”
  - Reference, not a narrative, does not “tell a story”
  - Focus on precision and rigor
  - Goal is requirements that are precise, unambiguous, complete, and consistent

ConOps: Informal Specification Techniques

- Use natural language and other informal methods
  - 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.

SRS Template

1. Introduction
   1.1 Intended Audience and Purpose
   - Describes the audience and the use of the document for. If some stakeholders are more important than others, describes the priority.

   1.2 How to use the document
   - Describes the document organization. The ConOps should answer the questions, “Where do I find particular information about X?”

2. Concept of Operations
   - SRS Template
     - Provides a high-level overview of the system’s operation and the context in which it will be used.

   2.1 System Context
     - System capabilities
     - Customer and business goals

   3. Behavioral Requirements
     - Specification of the observable system behavior

   3.1 System Inputs and Outputs

   3.2 Detailed Output Behavior

Scenario Analysis Process

Applying scenario analysis in the requirements process

- Requirements Elicitation
  - Identify stakeholders who interact with the system
  - Collect “user stories” - how people would interact with the system to perform specific tasks

- Requirements Specification
  - Use cases with standard format
  - Use templates to standardize, drive elicitation

- Requirements verification and validation
  - Review use-cases for consistency, completeness, user acceptance
  - Verify prototypes against code (e.g., use-case based testing)
Creating Use Cases

• Identify a key actor and purpose
  – The purpose informs the use case title and description
• Identify the main flow (ideal path) from the starting point to the result
  – Preconditions: anything that must be true to initiate the Use Case
  – Trigger: event, if any, initiating the Use Case
  – Basic Flow: sequence of interactions from the trigger event to the result
  – Alternative Flows: identify sequences branching off the Basic Flow

Guidelines for Good Use Cases

• Use Cases should express requirements, not design
  – Focus on import results that provide value to specific actors
    • i.e., if nobody really cares about the outcome, it is not a good use case
  – Focus on what the actor is doing, not the details of how
    • Not: “The user left-clicks on the radio button labeled Balance and presses the Enter button”
    • “The user elects the option to view the balance.”
• Looking for a small number of use cases that capture the most important interactions

Voting System Example

• Who are the actors
• What are the major tasks?
• What are the outcomes?
• What would be an alternative flow?
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

Example state transition diagram

SE, Modeling, Hans van Vliet, ©2008
Formal Specification Example

<table>
<thead>
<tr>
<th>Type Dictionary</th>
<th>Name</th>
<th>Base Type</th>
<th>Units</th>
<th>Legal Values</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speed</td>
<td>Integer</td>
<td></td>
<td></td>
<td>[0, 350]</td>
<td>Speed measured in nautical miles per hour.</td>
</tr>
<tr>
<td>Weight</td>
<td>Integer</td>
<td></td>
<td></td>
<td>[0,100]</td>
<td>Weighting for weighted average</td>
</tr>
<tr>
<td>Time</td>
<td>Integer</td>
<td></td>
<td></td>
<td>time &gt; 0</td>
<td>Time in seconds.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Monitored Variable Dictionary</th>
<th>Name</th>
<th>Type</th>
<th>Initial Value</th>
<th>Accuracy</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>LowResWS1 Speed</td>
<td>Speed</td>
<td>0</td>
<td>1</td>
<td>Wind speed reported by first low resolution sensor</td>
<td></td>
</tr>
<tr>
<td>LowResWS2 Speed</td>
<td>Speed</td>
<td>0</td>
<td>1</td>
<td>Wind speed reported by second low resolution sensor</td>
<td></td>
</tr>
<tr>
<td>HighResWS1 Speed</td>
<td>Speed</td>
<td>0</td>
<td>2.5</td>
<td>Wind speed reported by first high resolution sensor</td>
<td></td>
</tr>
<tr>
<td>HighResWS2 Speed</td>
<td>Speed</td>
<td>0</td>
<td>2.5</td>
<td>Wind speed reported by second high resolution sensor</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Controlled Variable Dictionary</th>
<th>Name</th>
<th>Type</th>
<th>Initial Value</th>
<th>Accuracy</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>TransWindSpeed</td>
<td>MagType</td>
<td></td>
<td>N/A</td>
<td>Transmission value of wind speed</td>
<td></td>
</tr>
</tbody>
</table>

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

Terminology

- Avoid “functional” and non-functional” classification
- Behavioral Requirements – any information necessary to determine if the run-time behavior of a given implementation constitutes an acceptable system
  - All quantitative constraints on the system's run-time behavior
  - Other objective measures (safety, performance, fault-tolerance)
  - In theory all can be validated by observing the running system and measuring the results
- Developmental Quality Attributes - any constraints on the system's static construction
  - Maintainability, reusability, ease of change (mutability)
  - Measures of these qualities are necessarily relativistic (i.e., in comparison to something else)

Quality Requirements

Behavioral vs. Developmental

<table>
<thead>
<tr>
<th>Behavioral (observable)</th>
<th>Developmental Qualities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performance</td>
<td>Modifiability(ease of change)</td>
</tr>
<tr>
<td>Security</td>
<td>Portability</td>
</tr>
<tr>
<td>Availability</td>
<td>Reusability</td>
</tr>
<tr>
<td>Reliability</td>
<td>Ease of integration</td>
</tr>
<tr>
<td>Usability</td>
<td>Understandability</td>
</tr>
<tr>
<td></td>
<td>Support concurrent development</td>
</tr>
</tbody>
</table>

Properties resulting from the properties of components, connectors and interfaces that exist at run time.

Properties resulting from the properties components, connectors and interfaces that exist at design time whether or not they have any distinct run-time manifestation.
Specifying Quality Requirements

• Is it important to specify the quality requirements explicitly? Unambiguously?
  – Hint: what role would quality requirements play in customer acceptance?
• Are these kinds of specifications adequate?
  – “The system interface shall be easy to use.”
  – “The system shall maximize the number of user transactions”

Example Timing Requirements

5.2. TIMING REQUIREMENTS FOR DEMAND FUNCTIONS

For all the demand functions, the rate of demand is so low that it will not constitute a significant CPU load.

For the stroke entries, the desired maximum delay is not known; the entry is the maximum delay in the current CPU which we will use as an approximation. In one case, both the current and desired values are given. The current value would be good enough to satisfy requirements, but the desired rate would be preferred.

<table>
<thead>
<tr>
<th>Function name</th>
<th>Minimum delay to completion</th>
</tr>
</thead>
<tbody>
<tr>
<td>FALSE</td>
<td>*200 ms</td>
</tr>
<tr>
<td>Switch AUTOCA, light on/off</td>
<td>*200 ms</td>
</tr>
<tr>
<td>Issue computer failure</td>
<td>no significant</td>
</tr>
<tr>
<td>Change slide size</td>
<td>*200 ms</td>
</tr>
<tr>
<td>Switch X, light on/off</td>
<td>*200 ms</td>
</tr>
<tr>
<td>Switch Y, light on/off</td>
<td>*200 ms</td>
</tr>
<tr>
<td>Switch Z, light on/off</td>
<td>*200 ms</td>
</tr>
<tr>
<td>Change latitude greater than 90 degrees</td>
<td>*200 ms</td>
</tr>
<tr>
<td>Switch N, light on/off</td>
<td>*200 ms</td>
</tr>
<tr>
<td>FILE</td>
<td>200 ms</td>
</tr>
<tr>
<td>剑</td>
<td>40 ms</td>
</tr>
</tbody>
</table>

Requirements Validation and Verification

• Feedback-control for requirements
• Should answer two distinct questions:
  – Validation: “Are we building to the right requirements?”
  – 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
• Validation requires going back to the stakeholders: can use many techniques
  – Review of specifications
  – Prototyping
  – Storyboarding
  – Use case walkthroughs
  – Review software iterations
• Verification requires checking work products against specifications
  – Review
  – Testing
  – Formal modeling and analysis
Questions?

Assignments

• Set up instructor meetings this week
• Finish incomplete drafts of deliverables