CS189A - Capstone

Chandra Krintz
Department of Computer Science
UC Santa Barbara
http://www.cs.ucsb.edu/~ckrintz/
Requirements engineering

• Process of establishing the services that the customer requires from a system and the constraints under which it operates and is developed.
  – May range from a high-level abstract statement of a service or of a system constraint to a detailed mathematical functional specification.
  – Precisely stated, unambiguous

• User requirements
  – Statements in natural language plus diagrams of the services the system provides and its operational constraints. Written for customers.

• System requirements
  – A structured document setting out detailed descriptions of the system’s functions, services and operational constraints. Defines what should be implemented so may be part of a contract between client and contractor.
User and system requirements

User requirement definition

1. The MHC-PMS shall generate monthly management reports showing the cost of drugs prescribed by each clinic during that month.

System requirements specification

1.1 On the last working day of each month, a summary of the drugs prescribed, their cost and the prescribing clinics shall be generated.
1.2 The system shall automatically generate the report for printing after 17.30 on the last working day of the month.
1.3 A report shall be created for each clinic and shall list the individual drug names, the total number of prescriptions, the number of doses prescribed and the total cost of the prescribed drugs.
1.4 If drugs are available in different dose units (e.g. 10mg, 20 mg, etc.) separate reports shall be created for each dose unit.
1.5 Access to all cost reports shall be restricted to authorized users listed on a management access control list.
Be Careful About Ambiguities in Informal Specifications

- “The input can be typed or selected from the menu“
  - The input can be typed or selected from the menu or both
  - The input can be typed or selected from the menu but not both

- “The number of songs selected should be less than 10”
  - Is it strictly less than?
  - Or, is it less than or equal?

- “The user has to select the options A and B or C”
  - Is it “(A and B) or C”
  - Or, is it “A and (B or C)”
Functional and Non-functional Requirements

- **Functional requirements (user + system requirements)**
  - Statements of services the system should provide, how the system should react to particular inputs and how the system should behave in particular situations.
  - May also state what the system should not do.

- **Domain requirements**
  - Constraints on the system from the domain of operation
    - Operating environment (e.g. underwater, temp range, environmental conditions to be tolerated)

- **Non-functional requirements**
  - Constraints on services or functions offered by the system such as timing constraints, constraints on the development process, standards, etc.
  - Often apply to the system as a whole rather than individual features or services.
Types of Non-functional Requirements

- Non-functional requirements
  - Product requirements
    - Efficiency requirements
      - Usability requirements
        - Performance requirements
      - Space requirements
    - Dependability requirements
    - Security requirements
  - Organizational requirements
    - Regulatory requirements
  - External requirements
    - Ethical requirements
    - Legislative requirements
    - Accounting requirements
    - Safety/security requirements
  - Development requirements
  - Environmental requirements
  - Operational requirements

# Metrics for Specifying Non-functional Requirements

<table>
<thead>
<tr>
<th>Property</th>
<th>Measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speed</td>
<td>Processed transactions/second</td>
</tr>
<tr>
<td></td>
<td>User/event response time</td>
</tr>
<tr>
<td></td>
<td>Screen refresh time</td>
</tr>
<tr>
<td>Size</td>
<td>Mbytes</td>
</tr>
<tr>
<td></td>
<td>Number of ROM chips</td>
</tr>
<tr>
<td>Ease of use</td>
<td>Training time</td>
</tr>
<tr>
<td></td>
<td>Number of help frames</td>
</tr>
<tr>
<td>Reliability</td>
<td>Mean time to failure</td>
</tr>
<tr>
<td></td>
<td>Probability of unavailability</td>
</tr>
<tr>
<td></td>
<td>Rate of failure occurrence</td>
</tr>
<tr>
<td></td>
<td>Availability</td>
</tr>
<tr>
<td>Robustness</td>
<td>Time to restart after failure</td>
</tr>
<tr>
<td></td>
<td>Percentage of events causing failure</td>
</tr>
<tr>
<td></td>
<td>Probability of data corruption on failure</td>
</tr>
<tr>
<td>Portability</td>
<td>Percentage of target dependent statements</td>
</tr>
<tr>
<td></td>
<td>Number of target systems</td>
</tr>
</tbody>
</table>
The software requirements document

- The official statement of what is required of the system developers
- Includes a specification of both user and system requirements
- Defines **WHAT** the system should do not **HOW** it should do it
  - Design comes later

- Agile and extreme SWE processes express requirements as
  - **Use cases** – how a system will act
  - Or as scenarios called **user stories** (describe result/benefit of it)
Agile Requirements Specification

1. Define project specifics
2. Team goals and objectives
3. Background and strategic fit
4. Assumptions
5. User Stories or Use Cases
6. User Interaction and Design
7. Questions
8. What we’re NOT Doing

• Evolve the document over time, concurrently with development

Required reading: https://www.atlassian.com/agile/requirements
Requirements validation techniques

• Requirements reviews
  – Systematic manual analysis of the requirements.
  – Review/commit changes to repository as part of workflow
    • Multiple team members OK it before committing
    • All team members get notification when its updated

• Prototyping
  – Using an executable model of the system to check requirements.

• Test-case generation
  – Developing tests for requirements to check testability.
Your Living Requirements Document: A Shared Google Doc

• Authors, Team, Project Title
• Intro – including problem, innovation, science, core technical advance (2-3 pages)
  – Define project specifics, team goals/objectives, background, and assumptions
• System architecture overview
  – High level diagram (1 page)
  – User interaction and design (1+ page)
• Requirements (functional and non-functional)
  – User stories or use cases (links)
  – Prototyping code, tests, metrics: github commits/issues
• System models: contexts, sequences, behavioral/UML, state
• Appendices
  – Technologies employed
PRDv1: Your Living Requirements Document: A Shared Google Doc (due in ~2 weeks)

- Authors, Team, Project Title
- Intro – including problem, innovation, science, core technical advance (2-3 pages)
  - Define project specifics, team goals/objectives, background, and assumptions
- System architecture overview
  - High level diagram (1 page)
  - User interaction and design (1+ page)
- Requirements (functional and non-functional)
  - User stories or use cases (links) → 10 for PRDv1 prioritized
  - Prototyping code, tests, metrics (5+ user stories): github commits/issues
- System models: contexts, sequences, behavioral/UML, state
- Appendices
  - Technologies employed
PRDv2: Your Living Requirements Document: A Shared Google Doc (due 1 month after PRDv1)

- Authors, Team, Project Title
- Intro: problem, innovation, science, core technical advance
  - Define project specifics, team goals/objectives, background, and assumptions
- System architecture overview
  - High level diagram (1 page)
  - User interaction and design (1 page)
- Requirements (functional and non-functional)
  - User stories or use cases (links) → 20+ for PRDv2 prioritized
  - Prototyping code, tests, metrics (10+ user stories): github commits/issues
- System models (1+ pages)
  - Contexts, interactions, structural, behavioral (UML)
  - Use cases, sequencing, event response, system state, classes/objects
- Appendices - Technologies employed