

- Requirements
- Chapters 6 and 7
- see SRS template for next submission

## Requirements

- A *requirement* is a property of the system being constructed
  - high-level business, functional- or non-functional, interface, ... properties
- It is generated by the customer in conjunction with the system engineer or analyst
- Requirements should focus on *what* the proposed system is intended to do, not *how* it will do it
  - may also include what the system should never do
  - should also include how errors are handled

## Properties

- Requirements should be simple not compound
- Requirements should be testable
  - Else they are *objectives*
- Requirements should be organized
  - Related requirements grouped
  - Abstract requirements containing detailed requirements
  - Priorities indicated
- Requirements should be numbered
  - So that they can be *traced*
- Requirements should be descriptive (*what*) not prescriptive (*how*)

## Requirements Process

- Elicitation / gathering
- Refinement
- Analysis and modeling
- Verification
  - Consistency, completeness, etc.
- Validation
  - Rapid prototyping, feedback from customers
- Specification
- Management
  - Traceability, change management

## Who/what is involved?

- Customers or people representing them
  - Well sure, they are using it
- Marketing
  - They are probably promoting its development
  - They have to figure out how to sell it
- Documentation
  - They will use requirements to produce documentation
- Testers
  - They will test against it

- Developers
  - Don't want something required they can't build
  - Needs to be complete and usable
- Technical Support
  - Since they will have to support it
- Customer Training
  - Need to develop any courses if needed
- Lawyers, Regulators
- Requests for Proposals
- Standards
- Earlier Versions, Competitors Products
- Business Plans
  
- Lots of people
  - A great deal of conflict

## Issues

- Customers do not know what they want. It is the analyst's job to *elicit* requirements and validate them with the customer
- There is a natural tendency for requirements to change over time
  - (hopefully not too much)
- The process of designing and building a system inherently raises new requirements

## Elicitation Techniques

- Interview / Meeting
- Survey / Questionnaire
- Observation
- Domain analysis
- Prototypes
- Scenarios / user stories
- Brainstorming

## Functional and non-functional requirements

- Functional
  - what the system should do
  - “A user shall be able to search either all of the initial set of databases or select a subset from it”

- Non-functional:
  - Product
    - efficiency (performance, transaction rate, response time), space; reliability, portability, usability, availability, manageability...
    - “.. UI shall be implemented as simple HTML without frames of Java applets”
  - Organizational
    - standards, implementation requirements (programming language, tools), delivery requirements (format)
    - “... documents shall conform to definition XYZ...”
  - External
    - interoperability with other systems; ethical; privacy issues, safety (from legal stand point too)
    - “... shall not disclose personal information apart from name and lib. ref. number to library staff...”

- Also domain requirements
  - functional and non-functional
  - formulas, standards
  - ...

## User vs. System Requirements

- User
  - natural language, no technical detail, high-level
  - purpose – user/client mgt should verify that the system will do what they intended
- System
  - detailed functional and non functional requirements
  - clearly specify (unique id)
  - visualize
- note: requirements document is a contract!

#### User requirement definition

1. The software must provide a means of representing and accessing external files created by other tools.

#### System requirements specification

- 1.1 The user should be provided with facilities to define the type of external files.
- 1.2 Each external file type may have an associated tool which may be applied to the file.
- 1.3 Each external file type may be represented as a specific icon on the user's display.
- 1.4 Facilities should be provided for the icon representing an external file type to be defined by the user.
- 1.5 When a user selects an icon representing an external file, the effect of that selection is to apply the tool associated with the type of the external file to the file represented by the selected icon.

- Requirements are What not How, *\*should\** be separate from design and implementation
  - not always true
  - especially when you start specifying more detailed system requirements
    - system design may start emerging
    - some design issues may be a requirement in their own
    - understanding how system components interact will expose design alternatives, procedures, data formats...
    - may need to adopt some design choices in order to proceed with validation of requirements specification

## Requirements Verification

- Correctness
  - Accurately reflects needs
- Completeness
  - No missing pieces
- Consistency
  - Absence of conflicts
  - hard to avoid – e.g., optimize for performance, size, reliability...
- Clarity
  - Absence of ambiguity
- Coherence
  - Singleness of purpose
- Feasibility
  - Capable of being accomplished
- Testability
  - if you cannot think off possibilities how to test the requirement, it will probably be hard to ensure it's implemented in the first place
- Traceability
  - Throughout the life cycle *What not how*
- Modularity / organized
- Needed by the customer

- Spacecraft system
  - To minimise weight, the number of separate chips in the system should be minimised.
  - To minimise power consumption, lower power chips should be used.
  - However, using low power chips may mean that more chips have to be used. Which is the most critical requirement?

- to ease verification/validation efficiently – need to be able to represent and analyze requirements specification efficiently/clearly
- Representation options
  - Natural Language
  - Form based
  - Tabular
  - Graphs and Sequences diagrams
- select a format, stick to it, use combination of fonts/highlights/...
- may supplement structured language with tables/graphs/sequence diagrams
- should not be technical jargon – remember customer needs to understand and agree with SRS

## Analysis

- Goal hierarchies
- Use cases/scenarios
- Context diagrams
- Formal modeling
- Simulation
- Object-oriented analysis

## Form-based node specification

*Insulin Pump/Control Software/SRS/3.3.2*

**Function** Compute insulin dose: Safe sugar level

**Description** Computes the dose of insulin to be delivered when the current measured sugar level is in the safe zone between 3 and 7 units.

**Inputs** Current sugar reading (r2), the previous two readings (r0 and r1)

**Source** Current sugar reading from sensor. Other readings from memory.

**Outputs** CompDose \$ the dose in insulin to be delivered

**Destination** Main control loop

**Action:** CompDose is zero if the sugar level is stable or falling or if the level is increasing but the rate of increase is decreasing. If the level is increasing and the rate of increase is increasing, then CompDose is computed by dividing the difference between the current sugar level and the previous level by 4 and rounding the result. If the result, is rounded to zero then CompDose is set to the minimum dose that can be delivered.

**Requires** Two previous readings so that the rate of change of sugar level can be computed.

**Pre-condition** The insulin reservoir contains at least the maximum allowed single dose of insulin..

**Post-condition** r0 is replaced by r1 then r1 is replaced by r2

**Side-effects** None

## Tabular specification

Condition	Action
Sugar level falling ( $r2 < r1$ )	CompDose = 0
Sugar level stable ( $r2 = r1$ )	CompDose = 0
Sugar level increasing and rate of increase decreasing ( $(r2-r1) < (r1-r0)$ )	CompDose = 0
Sugar level increasing and rate of increase stable or increasing. $((r2-r1) \geq (r1-r0))$	CompDose = round $((r2-r1)/4)$ If rounded result = 0 then CompDose = MinimumDose



- Beware of vague verbs (*handled, rejected, processed, skipped, eliminated*)
- Beware of ambiguous pronouns (“*The I/O module communicates with the data validation module and its control flag is set.*”)
- Look for statements that imply certainty (always, every, all, none, never) and ask for proof.
- When a term is explicitly defined in one place, try substituting the definition for other occurrences.

## Samples

- As Written
  - *Software will not be loaded from unknown sources onto the system without first having the software tested and approved*
- Better
  - *3.2.5.2 Software shall be loaded onto the operational system only after it has been tested and approved*
- Better
  - *3.2.5.2 Software shall be loaded onto the operational system only after it has been tested to be in accordance with MIL-SPEC 3425 and approved by the Change Control Board (CCB)*

- 3.4.6.3
  - The system shall  
Prevent the processing of duplicate electronic files by checking a new SDATE record. An email message shall be sent upon occurrence
- 3.4.6.3
  - The system shall
    - a. Prevent processing of duplicate electronic files by checking the date and time of submission
    - b. Send the following email message:
      - Request updated submission of date and time, if necessary,  
or
      - That the processing was successful when successful

- srs.doc

