Requirement Engineering

Natural Language Specification

Indranil Saha

Department of Computer Science and Engineering
Indian Institute of Technology Kanpur

Natural Language Specifications

- (Pros) Expressive, intuitive, and universal
- (Cons) Potentially vague, ambiguous, and its meaning depends on the background of the reader

Best Practices

- Use language consistently to distinguish between mandatory (use “shall”) and desirable (use “should”) requirements
- Use text highlighting (bold, italic, or color) to pick out key parts of the requirement
- Avoid using software engineering or domain-specific jargon, abbreviations, and acronyms
- Try to associate a rationale with each user requirement wherever possible

Examples of Natural Language Specifications

- (Specification) The system shall measure the blood sugar and deliver insulin, if required, every 10 minutes.

  (Rationale) Changes in blood sugar are relatively slow so more frequent measurement is unnecessary; less frequent measurement could lead to unnecessarily high sugar levels.

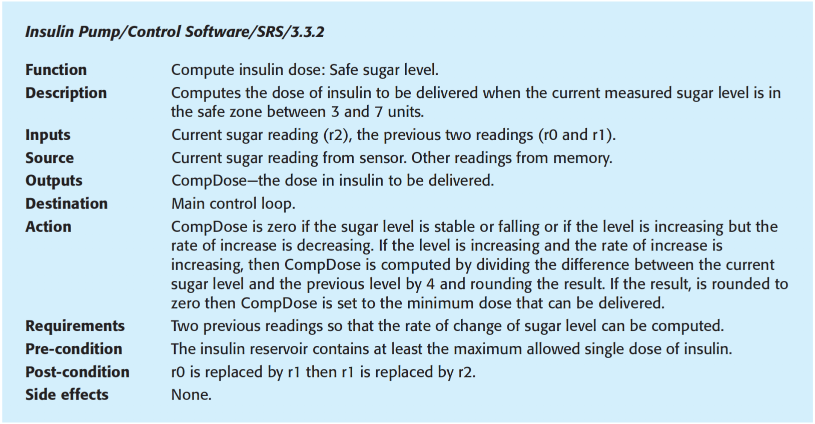
- (Specification) The system shall run a self-test routine every minute with the conditions to be tested and the associated actions defined in Table 1

  (Rationale) A self-test routine can discover hardware and software problems and alert the user to the fact the normal operation may be impossible

Structured Specifications

- Structured natural language is a way of writing system requirements where the freedom of the requirements writer is limited and all requirements are written in a standard way
- Maintains most of the expressiveness and understandability of natural language but ensures that some uniformity is imposed on the specification
- Structured language notations use templates to specify system requirements
- The specification may use programming language constructs to show alternatives and iteration, and may highlight key elements using shading or different fonts

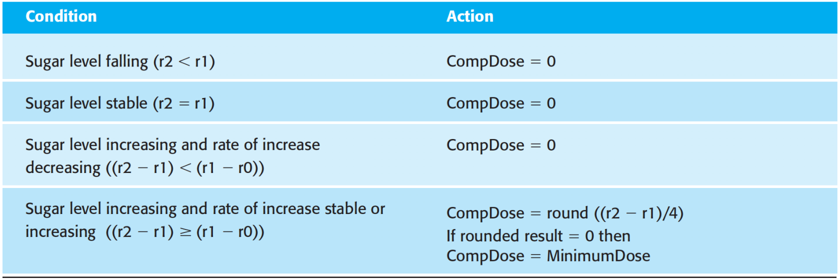
Example of Structured Specifications



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.
Requirements	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.

Example of Tabular Specifications



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

Requirement Engineering

Natural Language Specification

Indranil Saha

Department of Computer Science and Engineering
Indian Institute of Technology Kanpur