CIS 422/522

Technical Requirements (SRS)
Quality Requirements

1

Requirements Documentation

- Is a detailed requirements specification necessary?
- How do we know what "correct" means?
  - How do we decide exactly what capabilities code 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

- Focuses on developing a rigorous specification
  - Should be straight-forward to determine acceptable inputs and outputs
  - Preferably, can systematically check completeness consistency
- Use cases are not sufficient
- Generally accomplished by *modeling* required behavior
  - Formal model: models based on formal languages
  - Partial and semi-formal models

Formal Models

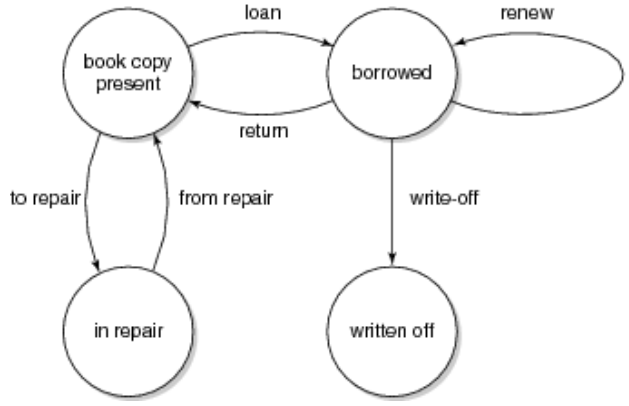
- Requirements modeling methods based on formal languages, e.g.
  - SCR: finite state machines
  - Z: formal logic
  - Statecharts: concurrent automata
- Advantages: allows users to
  - Derive the set of acceptable outputs for given inputs
  - Prove properties like consistency, completeness, safety, liveness
- Disadvantages
  - Requires rare skills
  - Expensive to produce and change
- Used seldom except where mission/safety critical (e.g., Intel fab after $475M FDIV error)

Semi-formal Modeling

- Many semi-formal methods used
  - Structured but non-mathematical models
  - Formal but partial models
- E.g. UML models add some rigor to Use Cases
  - Activity diagrams
  - Sequence diagrams
  - Disadvantage: tends to model design and implementation
- Modeling critical parts of the requirements
  - Use predicates (i.e., basic Boolean expressions)
  - Use mathematical expressions
  - Use tables
- 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
  - Often easier, less time consuming than trying to say the same thing in prose

Example state transition diagram

Does the Address Book have stateful behavior?
What are the states? Transitions?



SE, Modeling, Hans van Vliet, ©2008

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

For Your Projects

- Inputs and outputs
  - Be explicit about value types and ranges for each input variable (e.g. Name, Zip, phone)
    - How many digits? Other characters?
  - Be explicit about acceptable outputs
    - Export values and formats
    - Values output to printer (i.e., how is the output a function of the stored values?)
  - Easiest to define the inputs and outputs as abstract variables
- Detailed behavioral requirements
  - Specify acceptable results for a sort
  - Specify acceptable search results
  - Specify state changes (if applicable)

Quality Requirements

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 Requirements- any constraints on the system's static construction
  - Maintainability, reusability, ease of change (mutability)
  - Measures of these qualities are necessarily relative (I.e., in comparison to something else)

Behavioral vs. Developmental

Behavioral (observable)	Developmental Qualities
• Performance • Security • Availability • Reliability • Usability	• Modifiability(ease of change) • Portability • Reusability • Ease of integration • Understandability • Support concurrent development
Properties resulting from the behavior of components, connectors and interfaces that exist at run time.	Properties resulting from the structure of 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 support the maximum possible number of simultaneous users"

Specifying Quality Requirements

- When using natural language, write objectively verifiable requirements when possible
  - Load handling: "The system will support 15 or more concurrent users while staying within required performance bounds."
  - Maintainability: "The following kinds of requirements changes will require changes in no more than one module of the system…"
  - Performance:
    - "System output X has a deadline of 5 ms from the triggering input event."
    - "System output Y must be updated at a frequency of no less than 20 ms."
- Provides unambiguous requirement even if it is not practical to test for compliance

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 starred entries, the desired maximum delay is not known; the entry is the maximum delay in the current OFP, 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.

Function name	Maximum delay to completion
IMS:	
Switch AUTOCAL light on/off	*200 ms
Switch computer control on/off	*200 ms
Issue computer failure	not significant
Change scale factor	*200 ms
Switch X slewing on/off	*200 ms
Switch Y slewing on/off	*200 ms
Switch Z slewing on/off	*200 ms
Change latitude-greater-than-70-degrees	*200 ms
Switch INA light on/off	*200 ms
FLR:	
Enable radar cursor	200 ms
Slave or release slave	40 ms

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
  - Using them to check the quality of the implemented system

Questions?