Requirements Engineering: Pitfalls

CECS 590

Requirements Engineering – Outline

- WHY do we need Requirements Engineering and what is it?
- Principles: Definitions, process, roles, problem/solution view, artifact orientation
- System Models: Decomposition and abstraction, system views
- Frameworks: What reference structures can I use for requirements?
- Business Case Analysis: Why are we building this system?
- Stakeholders: Who are the people to talk to about requirements?
- Goals and Constraints: What are the major objectives for the system?
- System Vision: What exactly do we want to achieve?
- Domain Models: What are the surrounding systems ours interacts with?
- Usage Models: How will the system interact with the user?
- Software quality models: How to determine the quality characteristics?
- Quality requirements: How to specify which qualities need to be met?
- Typical pitfalls in Requirements Engineering
- Towards a system specification: How to hand over to design?
- Quality assurance: How to ensure that RE is done in a good way?
- Change management: How to evolve requirements?

Pitfalls in Requirements Engineering

1. Poor Requirements Quality
2. Over Emphasis on Simplistic Use Case Modeling
3. Inappropriate Constraints
4. Requirements Not Traced
5. Missing Requirements
6. Excessive Requirements Volatility including Unmanaged Scope Creep
7. Inadequate Verification of Requirements Quality
8. Inadequate Requirements Validation
9. Inadequate Requirements Management
10. Inadequate Requirements Process
11. Inadequate Tool Support
12. Unprepared Requirements Engineers

[Firesmith 2007, JOT, Vol. 6, No. 1, see Beachboard]

Poor Requirements Quality

- Ambiguous, not cohesive, incomplete, inconsistent, incorrect, out-of-date,
- Specified using technical jargon rather than the terminology of the user or business/application domain,
- Not restricted to externally-visible behavior or properties of the system,
- Infeasible to implement or manufacture,
- Not actually mandatory (i.e., merely nice-to-haves on someone's wish list),
- Irrelevant to the system being built,
- Lacking in necessary metadata such as priority and status,
- Untraced,
- Unverifiable and unvalidatable

→ Poor-quality requirements greatly increase development and sustainment costs and often cause major schedule overruns.

[Firesmith 2007, JOT, Vol. 6, No. 1, see Beachboard]

Poor Requirements Quality

List of “forbidden” words:

About, Applicable, Approximate, Bad, Best, Best Practice, Between, Clearly, Could, E.G., Easily, Easy, Effective, Efficient, Enable, Etc., Excellent, Fast, Finest, Good, High, His / Her, Ideal, Include But Shall Not Be Limited To, Least, Like, Low, Maximize, May, Minimum, Mostly, Relevant, Same, Should, Similar, So As, State Of The Art, Sufficient, Support, Support, Target, User Friendly, Worse, Would

[Firesmith 2007, JOT, Vol. 6, No. 1, see Beachboard]

Are These Good Requirements?

Requirement #1

- MTTR (Mean Time To Repair) will be less than 0.5 hours including diagnosis and without travel time

Requirement #2

- The system shall ensure the host processor is in Big Real Mode before any communication with the I/O Processor because the I2O Architecture states that the IOP message queues and message frames may be located anywhere in the system memory space, including above the 1MB Real Mode boundary. If the host processor is not in Big Real Mode, the system shall switch into Big Real Mode. The host processor shall save the previous host processor state and restore it upon exit.

Are These Good Requirements?

Requirement #3

- The third key requirement is power consumption. Generally, the power consumption requirements are driven by noise requirements, or CE compatibility. The customers expressed the need for lower active power consumption so that passive cooling can be used. However, this is one possible implementation, and other implementations need to be addressed by engineering. Standby power consumption should meet the levels obtained by CE devices; 5-10W, and be achievable with the fan off. Cost is a factor. 10W standby is acceptable if the implementation cost is less than that of 5W standby. These requirements were articulated by Customer1, Customer2, Customer3, Customer4, and Customer5.

A Good Requirement is:

- Complete
- Correct
- Concise
- Feasible
- Necessary
- Prioritized
- Unambiguous
- Verifiable
- Consistent
- Traceable

Ambiguous

- A Slightly Ambiguous Requirement
- From an actual memorandum, as quoted in Pilot Magazine in December 1996:
- The Landing Pilot is the Non-Handling Pilot until the decision altitude call, when the Handling Non-Landing Pilot hands the handling to the Non-Handling Landing Pilot, unless the latter calls “go around," in which case the Handling Non-Landing Pilot continues handling and the Non-Handling Landing Pilot continues non-handling until the next call of "land" or "go around" as appropriate. In view of recent confusions over these rules, it was deemed necessary to restate them clearly.

Unverifiable

Requirements are often unverifiable because they are ambiguous, can't be decided, or not worth the cost to verify.

- "The system shall not fail during the first 5 years of normal operation"
- "The product must be easy to learn"
- "Use of the device shall reduce fatalities by 30% in the event of a catastrophic failure"

Are These Good Requirements?

When creating a data element, the system should pre-populate data wherever possible.

Users shall not be prevented from deleting data they have entered, because they own it. Users must be able to view or maintain another user's data, but must not be able to delete it without the appropriate permissions.

3.1.5: The XYZ product must be fast and easy to use.

Good Requirements Checklist

Complete:

✓ ▪Formal review by domain experts and stakeholders indicates that all necessary material is included.

✓ ▪Exceptional behavior is specified (the "else" of the requirement).

✓ ▪No "TBDs" remain.

✓ ▪The content is detailed enough to drive the current phase of the development process.

✓ ▪The content is not arbitrarily or prematurely detailed.

✓ ▪It is economically safe to proceed.

Correct:

✓ Stakeholder/SME review locates no errors.

✓ The requirements are consistent with all source materials.

✓ The requirements have been reviewed and approved by all appropriate parties.

Also see other attributes for related items.

Good Requirements Checklist

Concise

✓ Each requirement addresses a single issue.

✓ Rationale, examples, and other supporting data are separated from the requirement.

✓ The requirement is expressed using the simplest grammar and as few words as possible.

Feasible

✓ All Requirements are known to be feasible through use in prior products, through analysis, or through prototyping.

Necessary

✓ Each requirement can be traced to at least one of the following:

- Market Segment Analysis or lateral benchmarking of similar products
- A need expressed by the customer or end user
- Planned implementation of a new usage model
- Business strategy, roadmaps, or sustainability needs

✓ All stakeholders agree that each product requirement is necessary.

Good Requirements Checklist

Prioritized:

✓ ·Tradeoffs between requirements are clear.

✓ ·Multiple dimensions have been considered, such as cost, customer value, and development risk.

✓ ·All product stakeholders have provided input to the prioritization process.

✓ ·The requirements are realistically distributed among the priority levels.

Unambiguous:

✓ ·Each requirement is clear to the intended audience, possessing a single interpretation.

✓ ·Terms are defined where necessary and used consistently.

✓ ·The requirements are devoid of weak words (easy, fast, etc.) and unbounded lists (such as, including, ...).

✓ ·Diagrams, algorithms, use cases, tables, or other devices are used to reduce ambiguity where appropriate.

Good Requirements Checklist

Verifiable:

✓ ·Each requirement is unambiguous.

✓ ·The implementation of each requirement can be clearly and effectively established via demonstration, inspection, or testing.

✓ ·Quality and performance requirements are quantified using an appropriate scale of measure.

Consistent:

✓ ·Each requirement is represented only once in a specification and referenced where

needed.

✓ ·Each requirement is internally consistent with other product requirements at its level.

✓ ·Each requirement is externally consistent with requirements at other levels (product, business, market, etc.).

Traceable:

✓ Each requirement is uniquely and persistently identified.

✓ Each requirement is written as concisely and simply as possible. ✓ Each requirement expresses only one function or idea.

Buzzword Bingo

large	often	etcetera	minimize	usually
improved	sexy	robust	best-of-breed	optionally
and/or	user-friendly	FREE	maximize	easy
probably	support	optimize	preferably	state-of-the-art
average	efficient	fast	world class	usual

Five buzzwords in any row, column, or diagonal is "BINGO!"
Only mark off buzzwords uttered by others.
Baiting your coworker is allowed.

Pitfalls in Requirements Engineering

1. Poor Requirements Quality
2. Over Emphasis on Simplistic Use Case Modeling
3. Inappropriate Constraints
4. Requirements Not Traced
5. Missing Requirements
6. Excessive Requirements Volatility including Unmanaged Scope Creep
7. Inadequate Verification of Requirements Quality
8. Inadequate Requirements Validation
9. Inadequate Requirements Management
10. Inadequate Requirements Process
11. Inadequate Tool Support
12. Unprepared Requirements Engineers

[Firesmith 2007, JOT, Vol. 6, No. 1, see Beachboard]