Chapter 4. Requirements Defects, Causes, and Mitigations

"We have met the enemy and he is us." Pogo (Walt Kelly)

4.1 Requirements defects

The following kinds of major defects may occur in various types of requirements e.g., a quality goal or domain function may be missing. This scheme does not include minor defects such as misspellings that do not imperil understanding.

4.1.1 Types of major defects in a requirement

- Unclear, ambiguous, poorly-stated, or unnecessary content
    - vague, ambiguous, or undefined terminology
    - confusing organization
    - unclear references, e.g., several
    - weak imperatives, e.g., should, could
    - unclear scope of negation
    - "or", which could be exclusive or inclusive
- Inadequate content
- Invalid content
    - incorrect
    - too restrictive e.g., unnecessary design details
    - infeasible
    - unverifiable
- Unnecessary e.g., outside of scope
- Unprioritized
- Misplaced or compound e.g., using "and"

4.1.2 Types of major defects in a set of requirements specs

- Missing specs
    - rarely used functions
    - tacit (subconscious) activities
    - undiscovered requirements
    - incomplete decomposition
- Conflicting sets
- Infeasible sets
- Redundant sets

4.2 Defect causes and mitigations

In the following, defect causes and their mitigations are organized by stakeholder.

The diversity of requirements defects and mitigations makes clear that **there is no easy fix for the problem of defective requirements**.

4.2.1 Customers, users, support staff, and subject matter experts

4.2.1.1 Inadequate understanding of the nature of requirements or requirements development (RD) responsibilities and tasks

Caused by simplistic or inadequate guidance e.g., just two types: functional and non-functional

Mitigate by providing some of the following:

- Requirements and RD overview video for domain and application sources
- Case studies of effective RD
- A set of meta-requirements, with examples, for domain and application sources
- Functional and quality attribute requirements templates and analyzers
- Examples of application models and specs
- Requirements review description and checklists

4.2.1.2 Ineffective communication of application requirements and their changes

May entail inclusion of defective information. May be caused by incomplete or inaccurate documentation or improper translation.

Mitigate by developing a glossary of domain and application terminology and by modeling the requirements e.g., prototyping or test designs.

4.2.1.3 Inadequate understanding of domain ontology

E.g., overlooked stakeholders or activities (e.g., rarely performed)

Mitigate by modeling domain ontology

4.2.1.3.1 Limited domain experience

Mitigate using Subject Matter Experts (SMEs), domain activity observation, or rapid prototyping

4.2.1.3.2 Inadequate imagining of use

Mitigate by stimulating imagination with identification workshops for user stories or use cases or using prototypes.

4.2.1.4 Unclear descriptions of application requirements and their changes

Vagueness, ambiguity, and insufficiency are the rule. The devil is in the many, many details.

Mitigate by providing

- Familiar examples of clear and unclear specs
- Unclear language detectors
- User manual or operator manual early
- Requirements-based test designs
- Requirements review checklists
- Examples of application models
- Project glossary
- Templates for composite requirements

4.2.1.5 Excessively restrictive descriptions of application requirements

E.g., unnecessary design details

Mitigate by providing familiar examples of unnecessary details and their removal

4.2.1.6 Tacit knowledge

Mitigate by probing identification and decision-making processes and by noting tacit knowledge markers**.** For example, assume a potential user mentions a process you know nothing about or a decision you can't imagine making based on what you know. When you ask how to perform the process or make the decision, you may get an explicit set of steps or a "Well, ...". If it is the latter, you have just found some tacit knowledge. The unknown process or decision names are the markers. Other markers are the well-known process box labeled "Then magic happens". If it needs magic to get from input to output, something is missing.

4.2.1.7 Inaccurate assumptions about developer understanding

Customers and users may assume that "others" know more about the domain or application than they do.

Mitigate by documenting scope of developer understanding and experience as well as discussing unstated restrictions.

4.2.1.8 Inadequate participation

This may include disruptive behavior.

4.2.1.8.1 Inadequate sharing of information

This includes failure to identify other stakeholders who should take part.

Mitigate by stressing the critical nature of participation and clearly identify the results of inadequate participation.

4.2.1.8.2 Ineffective participation in reviews

Mitigate by using effective requirements reviews (e.g., High-Impact inspections [Ref 4.1]) and stressing the importance of skeptical attitudes

4.2.1.9 Infeasible goals

Goals may be technically or financially (i.e., not within budget) infeasible. Technical infeasibility may result from something that can't be done at all or two requirements that conflict.

When goals must be scaled back or modified, mitigate using conflict resolution techniques e.g., each party should state position and assumptions, differences should be identified and a mutually unsatisfactory resolution strategy defined.

4.2.1.10 Inappropriate prioritization of requirement tasks

Mitigate by documenting customer values and priorities.

4.2.1.11 Inadequate trust, motivation, or focus

If caused by a toxic culture, mitigate by seeking employment elsewhere.

Mitigate by identifying experiences and behaviors impacting trust or

motivation.

Set achievable near-term goals that might improve trust or motivation.

Trust can be established or reestablished by taking time to share stories, respecting experiences, understanding needs, setting goals together, and thoughtful listening.

4.2.2 Technical specialists and developers

Specialists include:

- safety specialists
- quality specialists
- architects
- V, V, & T specialists
- Project leads

4.2.2.1 Inadequate understanding of the nature of requirements or RD responsibilities and tasks

Mitigate by providing

- Requirements and RD overview video for technical specialists and developers
- Case studies of effective technical RD
- Comprehensive set of requirements templates and analyzers
- Examples of models and specs
- Project glossary
- Quality attributes model [Ref 4.2]
- Requirements review description and checklists

4.2.2.2 Inadequate understanding of technical issues

4.2.2.2.1 Inadequate understanding of domain ontology or application

4.2.2.2.1.1 Limited domain experience

Mitigate by involving SMEs and users early and by using domain activity observation or rapid prototyping

4.2.2.2.1.2 Inadequate understanding of domain ontology

(See 4.2.1.3)

4.2.2.2.2 Inadequate imagining of application hazards

Mitigate by using a safety specialist, by learning hazard analysis techniques, or by stimulating imagination with hazard identification workshops

4.2.2.2.3 Inexperience with technical requirements

Mitigate with guidance from requirements and technical specialists

4.2.2.3 Unclear descriptions of technical issues and their changes

(See 4.2.1.4)

4.2.2.4 Inadequate participation

(See 4.2.1.8)

4.2.2.5 Inappropriate prioritization of requirement tasks

(See 4.2.1.10)

4.2.2.6 Inadequate trust, motivation, or focus

(See 4.2.1.11)

4.2.3 Requirement specialists

4.2.3.1 Inadequate understanding of the nature of requirements or RD responsibilities and tasks

(See 4.2.2.1)

4.2.3.2 Inadequate understanding of domain ontology, application, or cost constraints

4.2.3.2.1 Limited domain experience

(See 4.2.2.2.1.1)

4.2.3.2.2 Inadequate understanding of domain ontology

(See 4.2.1.3)

4.2.3.2.3 Overlooked information sources

E.g., stakeholders, users, documents, or related systems

Mitigate by involving SMEs and users early

4.2.3.2.4 Inadequate imagining of usage or hazards

Mitigate by promoting incremental development and by taking part in user story, use case, or hazard identification workshops

4.2.3.3 Inadequate understanding of stakeholder relationships or attitudes

Mitigate by developing a stakeholder map based on stakeholder and middle management interviews.

4.2.3.4 Inadequate skills

Skill areas include:

- Requirements risk management
- Expectation management
- Elicitation or collaboration
- Specification and modeling
- Communication
- Requirements validation and verification

Mitigate by using training resources and identifying skill-area mentors.

4.2.3.5 Inadequate management of requirements suite

Mitigate by developing a suite management strategy.

4.2.3.6 Inadequate resources

Resources include:

- Access to information
- Stakeholder support
- Support tools
- Time

Mitigate by documenting the value of additional resources and the effects of not getting them.

4.2.3.7 Inadequate trust, motivation, or focus

(See 4.2.1.11)

4.3 Requirements risk management

In addition to defect-specific mitigations, consider the following risk management activities.

Study past requirements defects and their human causes

Monitor

- Developer and customer understanding
- Customer expectations
- Unresolved and looming conflicts
- Requirements quality
- Requirements changes
- Imprecise requirements
- Current requirements defects and their human causes

Bottom line

There are many types of defects, many human causes, and many mitigations. The challenge is find those requirements defects that are causing your significant failures and then implementing mitigations that will improve your software. Read about failure analysis in the next chapter.

Supplementary readings (References)

4.1 Gelperin, David (2005) "How Does High-Impact Work?" Available at understandingrequirements.com/chapter-2-references.php (2.13)

4.2 Gelperin, David (2017) Quality model at www.quality-aware.com/daves-q-a-stuff.php