Object Oriented System Analysis and Design

Chapter Four: Ensuring Your Requirements are Correct

Ewnetu E. (MSC.)

May 28, 2023

Contents

- Requirement validation techniques
- Testing early and often
- Use case scenario testing

Ensuring your Requirements are Correct

- Requirement documents serve both
  - As contract between us and the customer, detailing what we are to deliver, and
  - As guidelines for the designers, detailing what they build and how they are build.
- Thus, before the requirements can be turned over to the designers and system analysts, we and our customers must be absolutely sure that each knows the other's intent, and that our intents are captured in the requirements documents.
- To establish this certainty, we verify the requirements using different verification techniques such as data cross checking by employing many requirement gathering techniques at different sources .

...Cont'd

- Requirements Checking (Quality Criteria):
  - Consistency:
    - ★ Are there any requirements conflicts?
    - ★ The requirements must be compatible with each other.
  - Completeness: are all functions required by the customer included?
  - Realism: can the requirements be implemented given available budget and technology?
  - Adequacy: the requirements must address the actual needs of the system.
  - Unambiguity: every requirement must be described in a way that precludes different interpretations.

...Cont'd

- Comprehensibility: the requirements must be understandable by the stakeholders and should also include agreements.
- Verifiability: can the requirements be checked?
- Importance: each requirement must indicate how essential it is for the success of the project.
- Viability: all requirements can be implemented with the available technology, human resources and budget.
- Traceability: the context in which a requirement which was created should be easy to retrieved and can be traced back to make changes when needed.

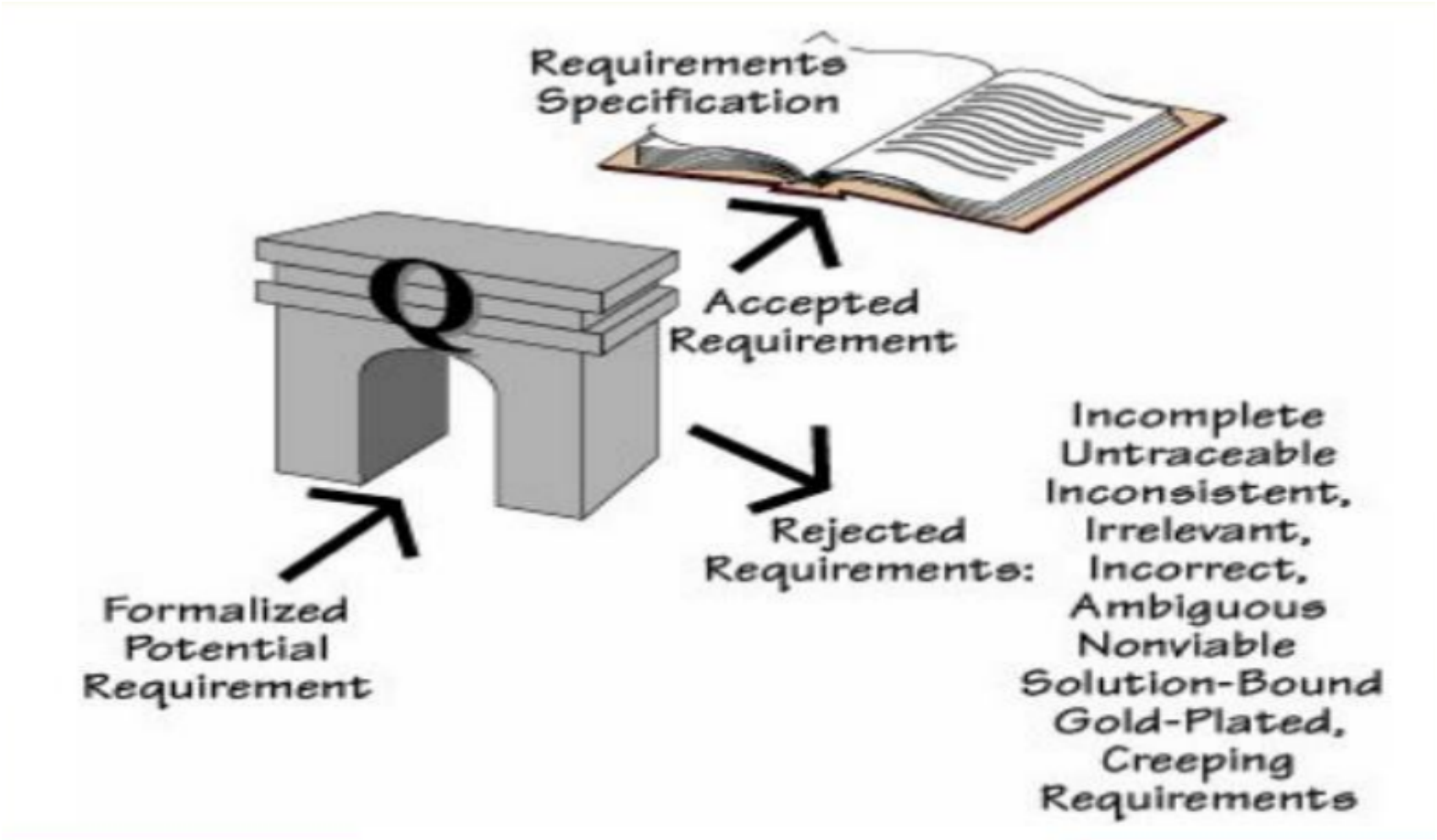
Validation Vs Verification

- Validation $\rightarrow$ is the process of checking whether the specification captures the customer's needs.
- Verification $\rightarrow$ is the process of checking that the software meets the specification.

Requirements Validation

- Validation denotes checking whether inputs, performed activities, and created outputs (requirements artifacts) of the requirements engineering core activities fulfill the defined quality criteria.
- Validation is performed by involving:
  - Relevant stakeholders, other requirement sources (standards, laws, etc.)
  - External reviewers, sometimes.
- Concerned with demonstrating that the requirements define the system that the customer really wants.
- Requirement error costs are high so verification is very important.
- Fixing a requirement error after delivery may cost up to 75 -100 times the cost of fixing an implementation error.

...Cont'd


Requirements Specification
Accepted Requirement
Rejected Requirements:
Incomplete
Untraceable
Inconsistent,
Irrelevant,
Incorrect,
Ambiguous
Nonviable
Solution-Bound
Gold-Plated,
Creeping
Requirements
Formalized Potential Requirement

Principles of Requirements Verification & Validation

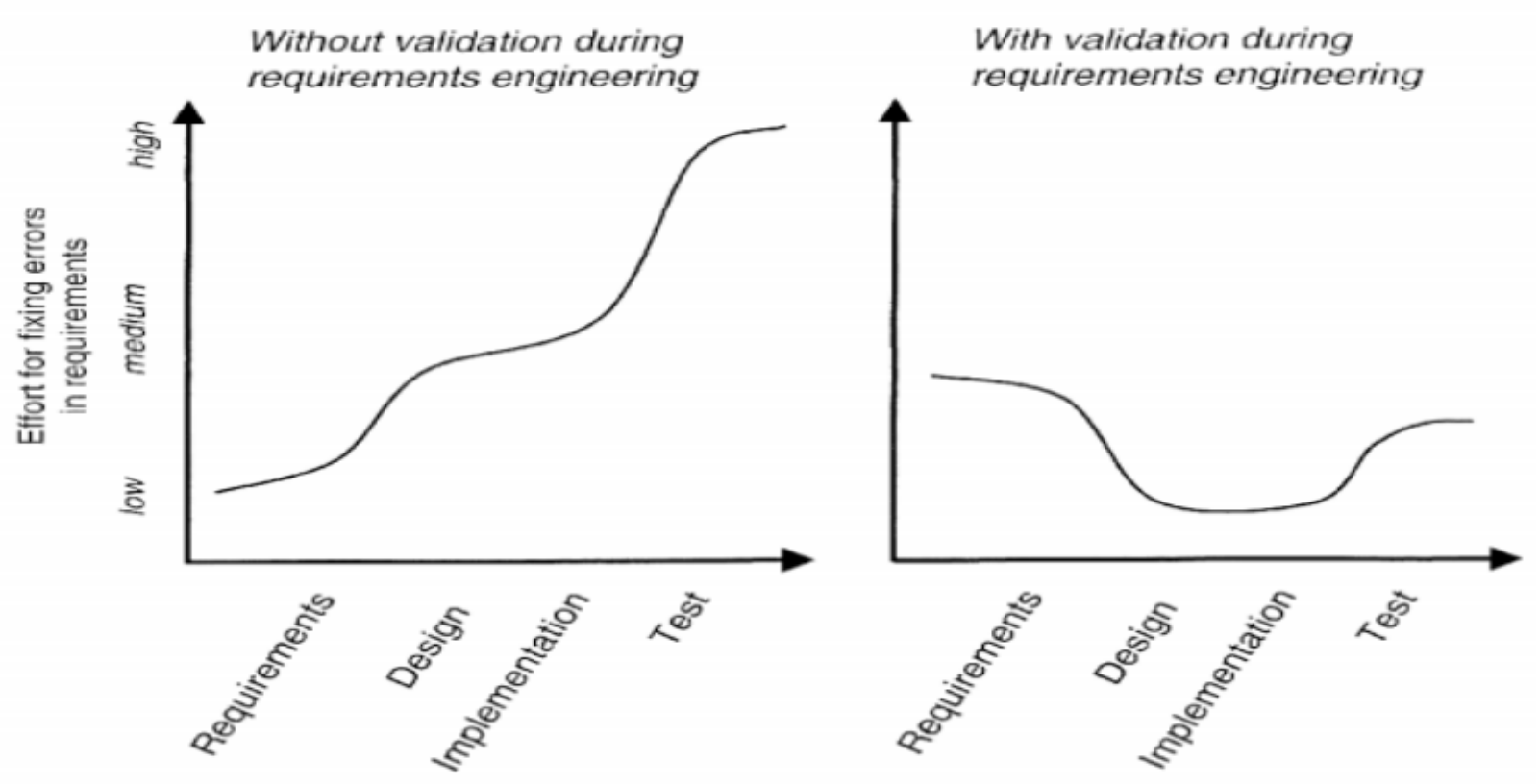
- 1. Involving the Right Stakeholders:
  - Ensure that relevant company & internal as well as relevant external stakeholders participate in validation.
  - Pay attention to the reviewers' independence and appoint external, independent stakeholders, if necessary.
- 2. Defect Detection vs. Defect Correction: separate defect detection from the correction of the detected defects.
- 3. Leveraging (making) Multiple Independent Views: whenever possible, try to obtain independent views that can be integrated during requirements validation in order to detect defects more reliably.

...Cont'd

- 4. Use of Appropriate Documentation Formats
  - Consider using organization standard to format requirement documents.
- 5. Creation of Development Artifacts/samples during Validation
  - If your validation approach generates poor results, try to support defect detection by creating development artifacts such as:
  - Architectural artifacts, test artifacts, user manuals, or goals and scenarios during validation.
- 6. Repeated Validation: establish guidelines that clearly determine when or under what conditions an already released requirements artifact has to be validated again.

Problems with Requirements Validation

- Requirements change quickly during requirements elicitation
- Inconsistencies easily added with each change
- Tool support is needed



Requirements Validation Techniques

- Requirements reviews: systematic manual analysis of the requirements
- Prototyping: using an sample model of the system to check requirements.
  - Allows the stakeholders to try out the requirements for the system and experience them early.
- Test-case generation: developing test cases for requirements to check testability
- Inspection: an organized examination process of the requirements.
- Walkthrough: a walkthrough does not have formally defined procedure and does not require a differentiated role assignment.
  - Checking early whether an idea is feasible or not.
  - Obtaining the opinion and suggestions of other people.
  - Checking the approval of others and reaching agreement.

Testing Early and Often

All tests are made up of at least 4 basic components.

1. Testing Procedure: what are the steps that need to be carried out in order to complete the test
2. Testing Facilities: a list of what are the equipment, the laboratory setting, and the other resources you will need to carry out the testing procedure.
3. Entry Condition: what is the state of the project required in order for the test to begin
4. Exit Condition: what criteria must be met in order for the test to be considered a success or for the test to earn a "passing" score.

Use case Scenario Testing

- A scenario test is a test based on a scenario.
- An ideal scenario test has several considerations:
  - The test should be based on a story about how the program is used: including information about the motivations of the people involved.
  - The story should be motivating: a stakeholder with influence would push to fix a program that failed the test.
  - The story should be credible: it not only could happen in the real world; stakeholders would believe that something like It probably will happen.
  - The story may involve a complex use of the program or a complex environment
  - The test results should be easy to evaluate: this is valuable for all tests, but is especially important for scenarios because they are complex.

Why Usecase Scenario Tests?

- Learn the product
- Expose failures
- Explore expert use of the program
- Bring requirement related issues to the surface, which might involve reopening old requirements discussions (with new data) or surfacing not yet identified requirements.

12 ways to create Good Scenarios

1. Write life histories for objects in the system.
2. List possible users, analyze their interests and objectives.
3. Consider disfavored users: how do they want to abuse your system?
4. List "system events": how does the system handle them?
5. List "special events": what accommodations does the system make for these?
6. List benefits and create end-to-end tasks to check them.
7. Interview users about the visible challenges and failures of the old system.
8. Work alongside users to see how they work and what they do.
9. Read about what systems like this are doing.
10. Study complaints about the predecessor to the system or its competitors.
11. Create a mock business: treat it as real and process its data.
12. Try converting real-life data from a competing or predecessor application.

Thank you!!!