Requirements Engineering

Chapter 4 Requirements Validation

Learning Objective

...to emphasize that validation is concerned with checking the documentation for consistency, completeness and accuracy. This chapter enumerates complementary approaches including prototyping, automated model validation where automatic checks may be applied to a structured or formal requirements model of the system.

Frederick T Sheldon
Assistant Professor of Computer Science
University of Colorado at Colorado Springs

Validation objectives

- ⊗ Certifies that the requirements document is an acceptable description of the system to be implemented
- ⊗ Checks a requirements document for
  - Completeness and consistency
  - Conformance to standards
  - Requirements conflicts
  - Technical errors
  - Ambiguous requirements

Analysis and validation

- Analysis works with raw requirements as elicited from the system stakeholders
  - "Have we got the right requirements" is the key question to be answered at this stage
- Validation works with a final draft of the requirements document i.e. with negotiated and agreed requirements
  - "Have we got the requirements right" is the key question to be answered at this stage

Validation inputs and outputs

Validation inputs

- Requirements document
  - Should be a complete version of the document, not an unfinished draft. Formatted and organized according to organizational standards
- Organizational knowledge
  - Knowledge, often implicit, of the organization which may be used to judge the realism of the requirements
- Organizational standards
  - Local standards e.g. for the organization of the requirements document

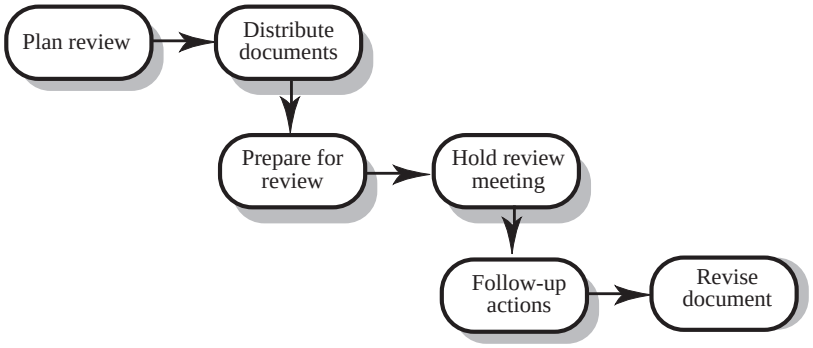
Validation outputs

- Problem list
  - List of discovered problems in the requirements document
- Agreed actions
  - List of agreed actions in response to requirements problems. Some problems may have several corrective actions; some problems may have no associated actions

Requirements reviews

- A group of people read and analyze the requirements, look for problems, meet and discuss the problems and agree on actions to address these problems

Requirements review process



Review activities

- Plan review
  - The review team is selected and a time and place for the review meeting is chosen.
- Distribute documents
  - The requirements document is distributed to the review team members
- Prepare for review
  - Individual reviewers read the requirements to find conflicts, omissions, inconsistencies, deviations from standards and other problems.

Review activities

- Hold review meeting
  - Individual comments and problems are discussed and a set of actions to address the problems is agreed.
- Follow-up actions
  - The chair of the review checks that the agreed actions have been carried out.
- Revise document
  - The requirements document is revised to reflect the agreed actions. At this stage, it may be accepted or it may be re-reviewed

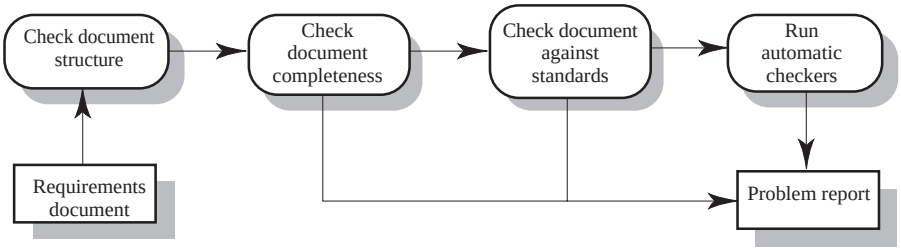
Problem actions

- ⊗ Requirements clarification
  - The requirement may be badly expressed or may have accidentally omitted information which has been collected during requirements elicitation.
- ⊗ Missing information
  - Some information is missing from the requirements document. It is the responsibility of the requirements engineers who are revising the document to discover this information from system stakeholders.
- ⊗ Requirements conflict
  - There is a significant conflict between requirements. The stakeholders involved must negotiate to resolve the conflict.
- ⊗ Unrealistic requirement
  - The requirement does not appear to be implementable with the technology available or given other constraints on the system. Stakeholders must be consulted to decide how to make the requirement more realistic.

Pre-review checking

- ⊗ Reviews are expensive because they involve a number of people spending time reading and checking the requirements document
- ⊗ This expense can be reduced by using pre-review checking where one person checks the document and looks for straightforward problems such as missing requirements, lack of conformance to standards, typographical errors, etc.
- ⊗ Document may be returned for correction or the list of problems distributed to other reviewers

Pre-review checking



Review team membership

- ⊗ Reviews should involve a number of stakeholders drawn from different backgrounds
  - People from different backgrounds bring different skills and knowledge to the review
  - Stakeholders feel involved in the RE process and develop an understanding of the needs of other stakeholders
- ⊗ Review team should always involve at least a domain expert and an end-user

Review checklists

- Understandability
  - Can readers of the document understand what the requirements mean?
- Redundancy
  - Is information unnecessarily repeated in the requirements document?
- Completeness
  - Does the checker know of any missing requirements or is there any information missing from individual requirement descriptions?
- Ambiguity
  - Are the requirements expressed using terms which are clearly defined? Could readers from different backgrounds make different interpretations of the requirements?

Review checklists

- Consistency
  - Do the descriptions of different requirements include contradictions? Are there contradictions between individual requirements and overall system requirements?
- Organization
  - Is the document structured in a sensible way? Are the descriptions of requirements organized so that related requirements are grouped?
- Conformance to standards
  - Does the requirements document and individual requirements conform to defined standards? Are departures from the standards, justified?
- Traceability
  - Are requirements unambiguously identified, include links to related requirements and to the reasons why these requirements have been included?

Checklist questions

- Is each requirement uniquely identified?
- Are specialized terms defined in the glossary
- Does a requirement stand on its own or do you have to examine other requirements to understand what it means?
- Do individual requirements use the terms consistently
- Is the same service requested in different requirements? Are there any contradictions in these requests?
- If a requirement makes reference to some other facilities, are these described elsewhere in the document?
- Are related requirements grouped together? If not, do they refer to each other?

Requirements problem example

- "4. EDDIS will be configurable so that it will comply with the requirements of all UK and (where relevant) international copyright legislation. Minimally, this means that EDDIS must provide a form for the user to sign the Copyright Declaration statement. It also means that EDDIS must keep track of Copyright Declaration statements which have been signed/not-signed. Under no circumstances must an order be sent to the supplier if the copyright statement has not been signed."

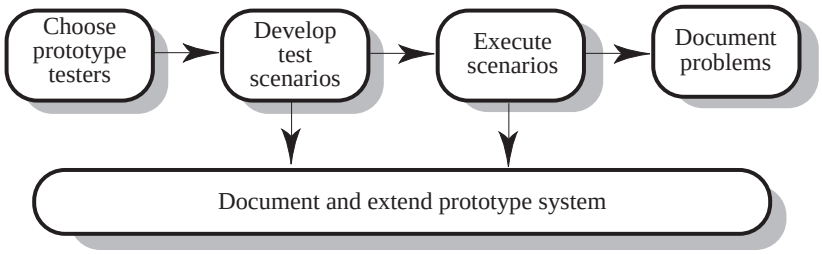
Problems

- ⊗ Incompleteness
  - What international copyright legislation is relevant?
  - What happens if the copyright declaration is not signed?
  - If a signature is a digital signature, how is it assigned?
- ⊗ Ambiguity
  - What does signing an electronic form mean? Is this a physical signature or a digital signature?
- ⊗ Standards
  - More than 1 requirement. Maintenance of copyright is one requirement; issue of documents is another

Prototyping

- ⊗ Prototypes for requirements validation demonstrate the requirements and help stakeholders discover problems
- ⊗ Validation prototypes should be complete, reasonably efficient and robust. It should be possible to use them in the same way as the required system
- ⊗ User documentation and training should be provided

Prototyping for validation



Prototyping activities

- ⊗ Choose prototype testers
  - The best testers are users who are fairly experienced and who are open-minded about the use of new systems. End-users who do different jobs should be involved so that different areas of system functionality will be covered.
- ⊗ Develop test scenarios
  - Careful planning is required to draw up a set of test scenarios which provide broad coverage of the requirements. End-users shouldn't just play around with the system as this may never exercise critical system features.
- ⊗ Execute scenarios
  - The users of the system work, usually on their own, to try the system by executing the planned scenarios.
- ⊗ Document problems
  - Its usually best to define some kind of electronic or paper problem report form which users fill in when they encounter a problem.

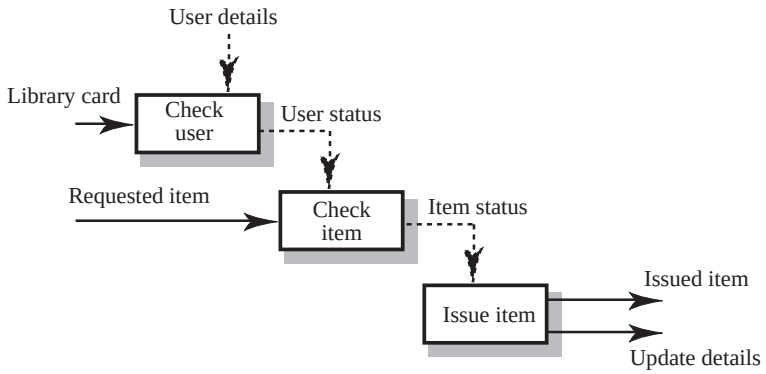
User manual development

- Writing a user manual from the requirements forces a detailed requirements analysis and thus can reveal problems with the document
- Information in the user manual
  - Description of the functionality and how it is implemented
  - Which parts of the system have not been implemented
  - How to get out of trouble
  - How to install and get started with the system

Model validation

- Validation of system models is an essential part of the validation process
- Objectives of model validation
  - To demonstrate that each model is self-consistent
  - If there are several models of the system, to demonstrate that these are internally and externally consistent
  - To demonstrate that the models accurately reflect the real requirements of system stakeholders
- Some checking is possible with automated tools
- Paraphrasing the model is an effective checking technique

Data-flow diagram for Issue



Paraphrased description

Check user	
Inputs and sources	User's library card from end-user
Transformation function	Checks that the user is a valid library user
Transformation outputs	The user's status
Control information	User details from the database
Check item	
Inputs and sources	The user's status from Check user
Transformation function	Checks if an item is available for issue
Transformation outputs	The item's status
Control information	The availability of the item
Issue item	
Inputs and sources	*None*
Transformation function	Issues an item to the library user. Items are stamped with a return date.
Transformation outputs	The item issued to the end user Database update details
Control information	Item status - items only issued if available

Requirements testing

- Each requirement should be testable i.e. it should be possible to define tests to check whether or not that requirement has been met.
- Inventing requirements tests is an effective validation technique as missing or ambiguous information in the requirements description may make it difficult to formulate tests
- Each functional requirement should have an associated test

Test case definition

- What usage scenario might be used to check the requirement?
- Does the requirement, on its own, include enough information to allow a test to be defined?
- Is it possible to test the requirement using a single test or are multiple test cases required?
- Could the requirement be re-stated to make the test cases more obvious?

Test record form

- The requirement's identifier
  - There should be at least one for each requirement.
- Related requirements
  - These should be referenced as the test may also be relevant to these requirements.
- Test description
  - A brief description of the test and why this is an objective requirements test. This should include system inputs and corresponding outputs.
- Requirements problems
  - A description of problems which made test definition difficult or impossible.
- Comments and recommendations
  - These are advice on how to solve requirements problems which have been discovered.

Requirements test form

Requirements tested: 10.(iv)

Related requirements: 10.(i), 10.(ii), 10.(iii), 10.(vi), 10. (vii)

Test applied: For each class of user, prepare a login script and identify the services expected for that class of user.

The results of the login should be a web page with a menu of available services.

Requirements problems: We don't know the different classes of EDDIS user and the services which are available to each user class. Apart from the administrator, are all other EDDIS users in the same class?

Recommendations: Explicitly list all user classes and the services which they can access.

Hard-to-test requirements

- ⊗ System requirements
  - Requirements which apply to the system as a whole. In general, these are the most difficult requirements to validate irrespective of the method used as they may be influenced by any of the functional requirements. Tests, which are not executed, cannot test for non-functional system-wide characteristics such as usability.
- ⊗ Exclusive requirements
  - These are requirements which exclude specific behavior. For example, a requirement may state that system failures must never corrupt the system database. It is not possible to test such a requirement exhaustively.
- ⊗ Some non-functional requirements
  - Some non-functional requirements, such as reliability requirements, can only be tested with a large test set. Designing this test set does not help with requirements validation.

Key points

- ⊗ Requirements validation should focus on checking the final draft of the requirements document for conflicts, omissions and deviations from standards.
- ⊗ Inputs to the validation process are the requirements document, organizational standards and implicit organizational knowledge. The outputs are a list of requirements problems and agreed actions to address these problems.
- ⊗ Reviews involve a group of people making a detailed analysis of the requirements.
- ⊗ Review costs can be reduced by checking the requirements before the review for deviations from organizational standards. These may result from more serious requirements problems.

Key points

- Checklists of what to look for may be used to drive a requirements review process.
- Prototyping is effective for requirements validation if a prototype has been developed during the requirements elicitation stage.
- Systems models may be validated by paraphrasing them. This means that they are systematically translated into a natural language description.
- Designing tests for requirements can reveal problems with the requirements. If the requirement is unclear, it may be impossible to define a test for it.