Requirements Verification and Validation

Table of Contents

- Introduction to Requirements Verification and Validation
- Requirements Verification and Validation Techniques
  - Simple checks
  - Prototyping
  - Functional test design
  - User manual development
  - Reviews and inspections
  - Model-based (formal) Verification and Validation

- The software is done. We are just trying to get it to work...\(^1\)

\(^1\) Anonymous
Requirements Verification and Validation

- **Requirements Validation**
  - Check that the *right product is being built*
  - Ensures that the software being developed (or changed) will satisfy its stakeholders
  - Checks the software requirements specification against stakeholders goals and requirements

- **Requirements Verification**
  - Check that *product is being built right*
  - Ensures that each step followed in the process of building the software yields the right products
  - Checks consistency of the software requirement specification artefacts and other software development products (design, implementation, ...) against the specification

Requirements Verification and Validation (2)

- Help ensure delivery of what the client wants
- Need to be performed at every stage during the (requirements) process
  - **Elicitation**
    - Checking back with the elicitation sources
    - “So, are you saying that . . . . ?”
  - **Analysis**
    - Checking that the domain description and requirements are correct
  - **Specification**
    - Checking that the defined system requirement will meet the user requirements under the assumptions of the domain/environment
    - Checking conformity to well-formedness rules, standards...
V&V vs. Analysis

- Both have several activities in common
  - Reading requirements, problem analysis, meetings and discussions...

- Analysis works with raw, incomplete requirements as elicited from the system stakeholders
  - Develop a software requirements specification document
  - Emphasis on “we have the right requirements”

- Requirements V&V works with a software requirements specification and with negotiated and agreed (and presumably complete) domain requirements
  - Check that these specifications are accurate
  - Emphasis on “we have the right requirements well done”

Requirements V&V Techniques
Various Requirements V&V Techniques

- Simple checks
  - Traceability, well-written requirements
- Prototyping
- Functional test design
- User manual development
- Reviews and inspections
  - Walkthroughs
  - Formal inspections
  - Checklists
- Model-Based V&V
  - First-order logic
  - Behavioral models

Simple Checks

- Various checks can be done using traceability techniques
  - Given the requirements document, verify that all elicitation notes are covered
  - Tracing between different levels of requirements
  - Checking goals against tasks, features, requirements...
- Involves developing a traceability matrix
  - Ensures that requirements have been taken into consideration (if not there should be a reason)
  - Ensures that everything in the specification is justified
- Verify that the requirements are well written (according to the criteria already discussed)
Prototyping (1)

- Excellent for validation by users and customers
  - More accessible than specification
  - Demonstrate the requirements and help stakeholders discover problems

- Come in all different shapes and sizes
  - From paper prototype of a computerized system to formal executable models/specifications
  - Horizontal, vertical
  - Evolutive, throwaway

Prototyping (2)

- Important to choose scenarios or use cases for elicitation session

- Prototyping-based validation steps
  - Choose prototype testers
  - Develop test scenarios
    - Careful planning is required to draw up a set of test scenarios which provide broad coverage of the requirements
    - Users should not just play around with the system as this may never exercise critical system features
  - Execute test scenarios
  - Document problems using a problem reporting tool
Comment on next two techniques

- The two V&V techniques, namely *Functional Test Design* and *User Manual Development*, are not really V&V techniques.
- They are activities that must be performed anyway, and they are based on the specification document.
- Through these activities, as for any other activities based on the specification document, errors and other problems with this document may be detected.

Functional Test Design

- Functional tests at the system level must be developed
  - Derived from the requirements specification
  - Each (functional) requirement should have an associated test
  - Non-functional requirements are harder to validate with testing
  - Each requirements test case must be traced to its requirements
  - Inventing requirements tests is an effective validation technique
- Designing these tests may reveal errors in the specification (even before designing and building the system)!
  - Missing or ambiguous information in the requirements description may make it difficult to formulate tests
- Some software development processes (e.g., agile methods) begin with tests before programming ➔ Test-Driven Development (TDD)
User Manual Development

- Same reasoning as for functional test design
  - Has to be done at some point
  - Reveals problems earlier
- Forces a detailed look at requirements
- Particularly useful if the application is rich in user interfaces / for usability requirements

- Typical information in a user manual
  - Description of the functionality
  - How to get out of trouble
  - How to install and get started with the system

Reviews and Inspections (1)

- A group of people read and analyze requirements, look for potential problems, meet to discuss the problems, and agree on a list of action items needed to address these problems

- A widely used requirements validation technique

- Can be expensive
  - Careful planning and preparation
  - Pre-review checking
  - Need appropriate checklists (must be developed if necessary and maintained)
Reviews and Inspections (2)

- Different types of reviews with varying degrees of formality exist (similar to JAD vs. brainstorming sessions)
  - Reading the document
    - A person other than the author of the document
  - Reading and approval (sign-off)
    - Encourages the reader to be more careful (and responsible)
  - Walkthroughs
    - Informal, often high-level overview
    - Can be led by author/expert to educate others on his/her work
  - Formal inspections
    - Very structured and detailed review, defined roles for participants, preparation is needed, exit conditions are defined
    - E.g., Fagan Inspection

Reviews and Inspections (3)

- Different types of reviews (cont’d)
  - Focused inspections
    - Reviewers have roles, each reviewer looks only for specific types of errors
Typical Review / Inspection Steps (1)

- 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

Typical Review / Inspection Steps (2)

- Prepare for review
  - Individual reviewers read the requirements to find conflicts, omissions, inconsistencies, deviations from standards, and other problems
- Hold review meeting
  - Individual comments and problems are discussed and a set of action items to address the problems is established
Typical Review / Inspection Steps (3)

- Follow-up actions
  - The chair of the review checks that the agreed action items have been carried out
- Revise document
  - Requirements document is revised to reflect the agreed action items
  - At this stage, it may be accepted or it may be re-reviewed

Review Team

- 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 a user
Review – Problem Categorization

- 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
- 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

Fagan Inspection (1)

- Formal and structured inspection process
**Fagan Inspection (2)**

- Characterized by rules on who should participate, how many reviewers should participate, and what roles they should play
  - Not more than 2 hours at a time, to keep participants focused
  - 3 to 5 reviewers
  - Author serves as the presenter of the document
  - Metrics are collected
    - Important: the author’s supervisor does not participate in the inspection and does not have access to data
    - This is not an employee evaluation
  - Moderator is responsible for initiating the inspection, leading the meeting, and ensuring issues found are fixed
  - All reviewers need to prepare themselves using checklists
  - Issues are recorded in special forms

**Fagan Inspection (3)**

- The inspection meeting is like a brainstorming session to identify (potential) problems
- Re-inspection if > 5% of the document change
  - Some variants are less tolerant... too easy to introduce new errors when correcting the previous ones!
Requirements Review Checklists (1)

- Essential tool for an effective review process
  - List common problem areas and guide reviewers
  - May include questions on several quality aspects of the document: comprehensibility, redundancy, completeness, ambiguity, consistency, organization, standards compliance, traceability ...
- There are general checklists and checklists for particular modeling and specification languages
- Checklists are supposed to be developed and maintained

- See examples on google

Requirements Review Checklists (2)

- Sample of elements in a requirements review checklist
  - Comprehensibility – 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?
  - Consistency – do the descriptions of different requirements include contradictions? Are there contradictions between individual requirements and overall system requirements?
Requirements Review Checklists (3)

- Sample of elements (cont’d)
  - **Organisation** – is the document structured in a sensible way? Are the descriptions of requirements organised 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? Do they include links to related requirements and to the reasons why these requirements have been included?

Comments on Reviews and Inspections

- **Advantages**
  - Effective (even after considering cost)
  - Allow finding sources of errors (not only symptoms)
  - Requirements authors are more attentive when they know their work will be closely reviewed
  - Encourage them to conform to standards
  - Familiarize large groups with the requirements (buy-in)
  - Diffusion of knowledge

- **Risks**
  - Need to be limited in time
  - Time consuming and expensive (but usually cheaper than the alternative)
Model-based (formal) Verification and Validation

Modeling paradigms

- **Entity-Relationship modeling** – e.g. UML Class diagrams
- **Workflow modeling notations** – there are many different “dialects”, such as UML Activity diagrams, UCM, BPML, Petri nets (a very simple formal model), Colored Petri nets
- **State machines** – e.g. Finite State Machines (FSM – a very simple formal model), extended FSMs, such as UML State diagrams
- **First-order logic** – notations such as Z, VDM, UML-OCL, etc.
  - Can be used as an add-on with the other paradigms above, by providing information about data objects and relationships (possibly in the form of “assertions” or “invariants” that hold at certain points during the dynamic execution of the model)
  - Can be used alone, expressing structural models and behavioral models (there are many examples of using Z for such purpose)
Formal V&V techniques and tools

- Available V&V techniques will vary from one modeling paradigms to another and will also depend on the available tools (that usually only apply to a particular modeling paradigm)
- The following functions may be provided through tools
  - Completeness checking: only according to certain syntax rules, templates
  - Consistency checking: given model M, show that M does not imply a contradiction and does not have any other undesirable general property (e.g. deadlock possibility)
  - Model checking: given a model M and some properties P, show that any system implementation satisfying M will have the properties P
  - Generation of system designs or prototype implementations (from workflow or state machine models)
  - Generation of test cases

Model checking

- Verifies that the model satisfies temporal logic properties, for example:
  - If A occurs, B will occur in the future (eventually)
  - If C occurs, D will be true always in the future
- Traverse systematically all possible behaviors (execution paths) of the machine (reachability analysis)
- Verification of properties done after reachability analysis or on the fly
- Model checker verifies $M \Rightarrow P$ (if no trace of states and transitions leading to the violation of $P$ is found; otherwise a counter example trace is provided)
- Major obstacle is state space explosion

Example tools:

- SPIN (see [http://spinroot.com/spin/whatispin.html](http://spinroot.com/spin/whatispin.html)) - for distributed systems with message passing
- Alloy (see [http://alloy.mit.edu/community/](http://alloy.mit.edu/community/)) – for OO Class diagrams with assertions