Validation Techniques

- What are goals of verification and validation?
- Checking quality
- Model analysis
Verification and Validation

• Validation:
  - "Are we building the right system?"
  - Does our problem statement accurately capture the real problem?
  - Did we account for the needs of all the stakeholders?

• Verification:
  - "Are we building the system right?"
  - Does our design meet the spec?
  - Does our implementation meet the spec?
  - Does the delivered system do what we said it would do?
  - Are our requirements models consistent with one another?
Refresher: V&V Criteria

- **Some distinctions:**
  - Domain Properties: things in the application domain that are true anyway
  - Requirements: things in the application domain that we wish to be made true
  - Specification: a description of the behaviours the program must have in order to meet the requirements

- **Two verification criteria:**
  - The Program running on a particular Computer satisfies the Specification
  - The Specification, given the Domain properties, satisfies the Requirements

- **Two validation criteria:**
  - Did we discover (and understand) all the important Requirements?
  - Did we discover (and understand) all the relevant Domain properties?
Validation Techniques

• What are goals of verification and validation?
• **Checking quality**
• Model analysis
Quality of Requirements Specification

J. Krogstie, Semiotic Quality Framework
SEQUAL

- $L$ - language extension
- $D$ - the domain
- $M$ - externalised model
- $K_s$ - relevant explicit knowledge of the stakeholders
- $K_m$ - relevant explicit knowledge of modellers
- $I$ - social actor interpretation
- $T$ - technical actor interpretation
- $G$ - organisationally-motivated goals of the modelling task
SEQUAL – model quality

- Physical quality
- Empirical quality
- Syntactic quality
- Semantic quality
- Pragmatic quality
- Perceived semantic quality
- Social quality
- Organisational quality
Physical quality

- Although information systems specifications and models are not of the physical kind, any model can be represent physically - e.g. on disk or paper

- Specification should be
  - Electronically stored
Empirical quality

Empirical quality deals with the variety of elements distinguished, error frequencies when being written or read, coding and ergonomics for computer-human interaction, for documentation and tools.

- **Specification should be:**
  - **Understandable**
    - all classes readers can easily comprehend the meaning of all requirements with a minimum of explanation.
  - **Concise**
    - it is as short as possible without affecting any other quality of the requirements specification.
Pragmatic quality

• **Pragmatic** quality is the correspondence between the model and the audience’s interpretation of it.

• Specification should be:
  • **Executable/Interpretable/Prototypable**
    • there exists a software tool capable of inputting the requirements specification and providing a dynamic behavioural model.
  • **Organised**
    • its contents are arranged so that readers can easily locate information and logical relationships among adjacent sections are apparent.
  • **Cross-referenced**
    • Cross-references are used to relate sections containing requirements with other relative requirements
Social quality

• The goal for **social** quality is agreement.

• Tool support
  • models created based on the different internal reality of the participants that are to agree
Syntactic quality

→ **Syntactic** quality includes correctness of lexicon, syntax and structural quality.

• Syntactic errors:
  • Syntactic invalidity
  • Syntactic incompleteness

• Automated tool support:
  • Error prevention
  • Error detection
  • Error correction
Semantic quality

→ **Semantic** quality is the correspondence between the model and the modelling domain

→ **Feasibility:**
  
  → Attempts at reaching a state of total validity and completeness will lead to unlimited spending of time and money on the modelling activity.

  → The time to terminate a modelling activity is thus not when the model is “**perfect**” *(which will never happen)*, but when it has reached a state where further modelling is regarded to be less beneficial than applying the model in its current state
Semantic quality

• **Complete**
  • everything that the software is supposed to do is included;
  • responses of the software to all realisable classes of input data in all recognisable classes of situations are included;
  • all pages are numbered, all figures and tables are numbered, named, and referenced; all terms are defined; all units of measure are provided; and all referenced material are present;
  • no sections are marked "To be determined".

• **Correct**
  • every requirement represent something required of the system to be built.

| User needs that the system must respond button press within 5 seconds. | Requirement: the system must respond button press within 10 seconds. |
Semantic quality

- **Internally consistent**
  - no subset of individual requirements stated therein conflict

  a) The light shall be lit when the button is pressed.
  b) When the button is released, the light shall become lit.

- **Externally consistent**
  - no requirement stated therein conflict with any already base-lined project documentation

  a) The system should *prompt* the message
  b) The system should *cue* the message
Semantic quality

- **Annotated by**
  - relative importance, relevant stability, version

- **Precise**
  - (1) numeric quantities are used whenever possible;
  - (2) the appropriate levels of precision are used for all numeric quantities.

- **Traced**
  - the origin of each of its requirements is clear

  The system shall respond to any occurrence of request X within 20 seconds.

- **Traceable**
  - it is written in a manner that facilitates the referencing of each individual statement

- **Verifiable**
  - there exist finite, cost effective techniques that can be used to verify that every requirement stated therein is satisfied by the system to be built.
Semantic quality

- **Achievable**
  - there could exist at least one system design and implementation that correctly implements all the requirements stated in the requirements specification

- **Design-independent**
  - there exist more than one system design and implementation that correctly implements all requirements stated in the requirements specification

- **At the right level of detail**
  - specific enough so that any system built that satisfies the requirements in the specification satisfies all user needs
  - abstract enough so that all systems that satisfy all user needs also satisfy all requirements
Semantic quality

- **Modifiable**
  - structure and style are such that any changes can be made easily, completely and consistently

- **Unambiguous**
  - every requirement stated therein has only one possible interpretation

<table>
<thead>
<tr>
<th>Up to 12 aircraft, the small display format shall be used. Otherwise the large display format shall be used.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aircraft that are non-friendly and have an unknown mission or the potential to enter restricted airspace within 5 minutes shall rise an alert.</td>
</tr>
<tr>
<td>Aircraft that are <em>either</em> non-friendly and have an unknown mission <em>or</em> have the potential to enter restricted airspace within 5 minutes shall rise an alert.</td>
</tr>
<tr>
<td>Aircraft that are non-friendly <em>and either</em> have an unknown mission <em>or</em> the potential to enter restricted airspace within 5 minutes shall rise an alert.</td>
</tr>
</tbody>
</table>
Orthogonal aspects

- **Reusable**
  - its sentences, paragraphs, and sections can be easily adopted and adapted for use in subsequent requirements specification.

- **Physical quality**
  - a persistent form that is available to those who potentially will want to reuse it

- **Syntactic quality**
  - syntactically correct

- **Semantic quality**
  - Similar domains
  - white-box reuse - modifiable, comprehensible and comprehended (techniques of pragmatic quality), annotated, and other

- **Social quality**
  - model integration and conflict resolution can be useful to investigate to what extent the solutions based on the model to be reused, should be reused.
THERE IS **NO** SUCH THING AS

**A PERFECT REQUIREMENTS SPECIFICATION!**

- **incomplete**
- **not understandable**
- **ambiguous**
- **redundant**
- **inconsistent**

- expand
- condense
- expand
- formalise
- resolve
- reduce
- add explanations
- expand

21
Validation Techniques

• What are goals of verification and validation?
• Checking quality
• Model analysis
Model Checking

• Has revolutionized formal verification:
  ➢ emphasis on partial verification of partial models
    • E.g. as a debugging tool for state machine models
  ➢ fully automated

• What it does:
  ➢ Mathematically – computes the “satisfies” relation:
    • Given a temporal logic theory, checks whether a given finite state machine is a model for that theory.
  ➢ Engineering view – checks whether properties hold:
    • Given a model (e.g. a FSM), checks whether it obeys various safety and liveness properties

• How to apply it in RE:
  ➢ The model is an (operational) Specification
    • Check whether particular requirements hold of the spec
  ➢ The model is (an abstracted portion of) the Requirements
    • Carry out basic validity tests as the model is developed
  ➢ The model is a conjunction of the Requirements and the Domain
    • Formalise assumptions and test whether the model respects them
Model Analysis

• Verification
  ➢ “Is the model well-formed?”
  ➢ Are the parts of the model consistent with one another?

• Validation
  ➢ Animation of the model on small examples
  ➢ Formal challenges:
    • “if the model is correct then the following property should hold…”
  ➢ ‘What if’ questions:
    • reasoning about the consequences of particular requirements;
    • reasoning about the effect of possible changes
    • “will the system ever do the following…”
  ➢ State exploration
    • E.g. use a model checking to find traces that satisfy some property
Requirements Specification

1 Introduction
   Purpose
   Scope
   Definitions, acronyms, abbreviations
   Reference documents
   Overview

2 Overall Description
   Product perspective
   Product functions
   User characteristics
   Constraints
   Assumptions and Dependencies

3 Specific Requirements

Appendices
Index
Goal-Scenario coupling

... initiate and influence the definition of ...

... classify ...

Goals

Scenarios

... illustrate satisfaction...

... lead to the identification of new ...

... lead to revision of ...
Key Relationships

Goals

Scenarios

Solution oriented requirements

Elicitation, refinement and validation of solution-oriented requirements

Refinement of existing and elicitation of new goals and scenarios
Documenting Solution-Oriented Requirements

Entity: entrance door  Entity: glass break detector

Data model  Behavioural model

Requirement
(natural language)

Transition diagram

State: alarm state  Event: inform security company

Event: entrance door damaged

If a glass break detector attached to the entrance door detects that the entrance door has been damaged, the system shall enter the alarm state and inform the security company.

Functional model

Data flow diagram

Function: inform security company
We’ve looked at the following non-UML diagrams

- **Goal Models**
  - Capture strategic goals of stakeholders
  - Good for exploring ‘how’ and ‘why’ questions with stakeholders
  - Good for analysing trade-offs, especially over design choices

- **Strategic Dependency Models (i*)**
  - Capture relationships between actors in an organisational setting
  - Helps to relate goal models to organisational setting
  - Good for understanding how the organisation will be changed
Use cases

Use Cases
- capture the view of the system from the view of its users
- good starting point for specification of functionality
- good visual overview of the main functional requirements

Cross-checks:
- Does each use case have a user?
  - Does each user have at least one use case?
- Is each use case documented?
  - Using sequence diagrams or use case template
Class diagrams

➢ **Class Diagrams**
  • capture the structure of the information used by the system
  • good for analysing the relationships between data items used by the system
  • good for helping you identify a modular structure for the system

➢ **Cross checks**
  • Does the class diagram capture all the classes mentioned in
    • - other diagrams?
    • - specification glossary?
  • Does every class have methods to get/set its attributes?
Statecharts

• capture all possible responses of an object to all use cases in which it is involved
• good for modeling the dynamic behavior of a class of objects
• good for analyzing event ordering, reachability, deadlock, etc.

Cross-checks:

➢ Does each statechart diagram capture (the states of) a single class?
  ➢ Is that class in the class diagram?
➢ Does each transition have a trigger event?
  ➢ Is it clear which object initiates each event?
  ➢ Is each event listed as an operation for that object’s class in the class diagram?
➢ Does each state represent a distinct combination of attribute values?
  ➢ Is it clear which combination of attribute values?
  ➢ Are all those attributes shown on the class diagram?
➢ Are there method calls in the class diagram for each transition?
  ➢ …a method call that will update attribute values for the new state?
  ➢ …method calls that will test any conditions on the transition?
  ➢ …method calls that will carry out any actions on the transition?
**Sequence Diagrams**
- capture an individual scenario (one path through a use case)
- good for modeling dialog structure for a user interface or a business process
- good for identifying which objects (classes) participate in each use case
- helps you check that you identified all the necessary classes and operations

**Cross-checks:**
- Is each class in the class diagram?
- Can each message be sent?
  - Is there an association connecting sender and receiver classes on the class diagram?
  - Is there a method call in the sending class for each sent message?
  - Is there a method call in the receiving class for each received message?
Validation Techniques

• What are goals of verification and validation?
• Checking quality
• Model analysis