Validation Techniques

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

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

Goals

... illustrate satisfaction ...

Scenarios

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

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

- Goals
- Scenarios

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

Requirement
(natural language)

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.

Data model
Behavioural model
Transition diagram

Entity–relationship diagram

Data flow diagram

Function: inform security company

State: alarm state  Event: 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

 avisuse 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
• Prototyping
Prototyping lifecycle

• **Prototyping is used for:**
  • understanding the requirements for the user interface
  • examining feasibility of a proposed design approach
  • exploring system performance issues

• **Problems:**
  • users treat the prototype as the solution
  • a prototype is only a partial specification
Prototyping

“A software prototype is a partial implementation constructed primarily to enable customers, users, or developers to learn more about a problem or its solution.” [Davis 1990]

“Prototyping is the process of building a working model of the system” [Agresti 1986]

• Approaches to prototyping
  ➢ Presentation Prototypes
    • explain, demonstrate and inform – then throw away
    • e.g. used for proof of concept; explaining design features; etc.
  ➢ Exploratory Prototypes
    • used to determine problems, elicit needs, clarify goals, compare design options
    • informal, unstructured and thrown away.
  ➢ Breadboards or Experimental Prototypes
    • explore technical feasibility; test suitability of a technology
    • Typically no user/customer involvement
  ➢ Evolutionary (e.g. “operational prototypes”; “pilot systems”):
    • development seen as continuous process of adapting the system
    • “prototype” is an early deliverable, to be continually improved.
• Throwaway Prototyping
  ➢ Purpose:
  • to learn more about the problem or its solution…
  • discard after desired knowledge is gained.
  ➢ Use:
  • early or late
  ➢ Approach:
  • horizontal - build only one layer (e.g. UI)
  • “quick and dirty”
  ➢ Advantages:
  • Learning medium for better convergence
  • Early delivery → early testing → less cost
  • Successful even if it fails!
  ➢ Disadvantages:
  • Wasted effort if reqts change rapidly
  • Often replaces proper documentation of the requirements
  • May set customers’ expectations too high
  • Can get developed into final product

• Evolutionary Prototyping
  ➢ Purpose
  • to learn more about the problem or its solution…
  • …and reduce risk by building parts early
  ➢ Use:
  • incremental; evolutionary
  ➢ Approach:
  • vertical - partial impl. of all layers;
  • designed to be extended/adapted
  ➢ Advantages:
  • Requirements not frozen
  • Return to last increment if error is found
  • Flexible(?)
  ➢ Disadvantages:
  • Can end up with complex, unstructured system which is hard to maintain
  • early architectural choice may be poor
  • Optimal solutions not guaranteed
  • Lacks control and direction
Validation Techniques

• What are goals of verification and validation?
• Checking quality
• Model analysis
• Prototyping
Build a prototype

Goal:
- Illustrate the major functionality of the system
- Check the feasibility and validity of the requirements

The prototype should support (a part of) the requirements specified in the requirements specification. Revise and complement the specification, if prototyping shows discrepancies

- Prototyping using
  - Pen, paper, post-its, markers, etc
  - Develop the mockups, show scenario what the specified system should do

- Prototyping using
  - proto.io
  - 15 days trial
Examples

• Paper mockups
  https://youtu.be/E1dNYHdA-bs
  https://youtu.be/ea069CQe-I4
  https://youtu.be/pUliqNL4t_Y
  https://youtu.be/KqvY6B1HJEM

• Using prototype tools
  https://app.moqups.com/VxOuSi2HQ2/view/page/ad64222d5?ui=0
  https://youtu.be/Mz0dZ4s067U