Lecture 8: Verification and Validation

• What are goals of V&V

• Validation Techniques
  ➢ Inspection
  ➢ Model Checking
  ➢ Prototyping

• Verification Techniques
  ➢ Consistency Checking
  ➢ Making Specifications Traceable

• Independent V&V
Lecture 8:
Verification and Validation

• What are goals of V&V

• V&V Techniques
  - Inspection
  - Model Checking
  - Prototyping
  - Making Specifications Traceable
  - Consistency Checking

• Independent V&V
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?
V&V Example

• Example:
  - Requirement R:
    - “Reverse thrust shall only be enabled when the aircraft is moving on the runway”
  - Domain Properties D:
    - Wheel pulses on if and only if wheels turning
    - Wheels turning if and only if moving on runway
  - Specification S:
    - Reverse thrust enabled if and only if wheel pulses on

• Verification
  - Does the flight software, P, running on the aircraft flight computer, C, correctly implement S?
  - Does S, in the context of assumptions D, satisfy R?

• Validation
  - Are our assumptions, D, about the domain correct? Did we miss any?
  - Are the requirements, R, what is really needed? Did we miss any?
Inquiry Cycle

Prior Knowledge
(e.g. customer feedback)

Observe
(what is wrong with the current system?)

Look for anomalies - what can’t the current theory explain?

Model
(describe/explain the observed problems)

Create/refine a better theory

Intervene
(replace the old system)

Design
(invent a better system)

Design experiments to test the new theory

Initial hypotheses

Carry out the experiments (manipulate the variables)
Shortcuts in the inquiry cycle

Prior Knowledge (e.g. customer feedback)

Observe (what is wrong with the current system?)

Model (describe/explain the observed problems)

Design (invent a better system)

Intervene (replace the old system)

Get users to try it

Check properties of the model

Analyze the model

Build a Prototype

Analyze (what is wrong with the model?)

Check properties of the model

Analyze the model

Build a Prototype

Analyze (what is wrong with the prototype?)

Get users to try it

Check properties of the model

Analyze the model

Build a Prototype

Analyze (what is wrong with the model?)

Check properties of the model

Analyze the model

Build a Prototype

Analyze (what is wrong with the prototype?)

Get users to try it

Check properties of the model

Analyze the model

Build a Prototype

Analyze (what is wrong with the model?)

Check properties of the model

Analyze the model

Build a Prototype

Analyze (what is wrong with the prototype?)

Get users to try it

Check properties of the model

Analyze the model

Build a Prototype

Analyze (what is wrong with the model?)

Check properties of the model

Analyze the model

Build a Prototype

Analyze (what is wrong with the prototype?)

Get users to try it

Check properties of the model

Analyze the model

Build a Prototype

Analyze (what is wrong with the model?)

Check properties of the model

Analyze the model

Build a Prototype

Analyze (what is wrong with the prototype?)

Get users to try it

Check properties of the model

Analyze the model

Build a Prototype

Analyze (what is wrong with the model?)

Check properties of the model

Analyze the model

Build a Prototype

Analyze (what is wrong with the prototype?)

Get users to try it

Check properties of the model

Analyze the model

Build a Prototype

Analyze (what is wrong with the model?)

Check properties of the model

Analyze the model

Build a Prototype

Analyze (what is wrong with the prototype?)

Get users to try it

Check properties of the model

Analyze the model

Build a Prototype

Analyze (what is wrong with the model?)

Check properties of the model

Analyze the model

Build a Prototype

Analyze (what is wrong with the prototype?)

Get users to try it

Check properties of the model

Analyze the model

Build a Prototype

Analyze (what is wrong with the model?)

Check properties of the model

Analyze the model

Build a Prototype

Analyze (what is wrong with the prototype?)

Get users to try it

Check properties of the model

Analyze the model

Build a Prototype

Analyze (what is wrong with the model?)

Check properties of the model

Analyze the model

Build a Prototype

Analyze (what is wrong with the prototype?)

Get users to try it

Check properties of the model

Analyze the model

Build a Prototype

Analyze (what is wrong with the model?)

Check properties of the model

Analyze the model

Build a Prototype

Analyze (what is wrong with the prototype?)

Get users to try it

Check properties of the model

Analyze the model

Build a Prototype

Analyze (what is wrong with the model?)

Check properties of the model

Analyze the model

Build a Prototype

Analyze (what is wrong with the prototype?)

Get users to try it

Check properties of the model

Analyze the model

Build a Prototype

Analyze (what is wrong with the model?)

Check properties of the model

Analyze the model

Build a Prototype

Analyze (what is wrong with the prototype?)

Get users to try it

Check properties of the model

Analyze the model

Build a Prototype

Analyze (what is wrong with the model?)

Check properties of the model

Analyze the model

Build a Prototype

Analyze (what is wrong with the prototype?)

Get users to try it

Check properties of the model

Analyze the model

Build a Prototype

Analyze (what is wrong with the model?)

Check properties of the model

Analyze the model

Build a Prototype

Analyze (what is wrong with the prototype?)

Get users to try it

Check properties of the model

Analyze the model

Build a Prototype

Analyze (what is wrong with the model?)

Check properties of the model

Analyze the model

Build a Prototype

Analyze (what is wrong with the prototype?)

Get users to try it

Check properties of the model

Analyze the model

Build a Prototype

Analyze (what is wrong with the model?)

Check properties of the model

Analyze the model

Build a Prototype

Analyze (what is wrong with the prototype?)

Get users to try it

Check properties of the model

Analyze the model

Build a Prototype

Analyze (what is wrong with the model?)

Check properties of the model

Analyze the model

Build a Prototype

Analyze (what is wrong with the prototype?)

Get users to try it

Check properties of the model

Analyze the model

Build a Prototype

Analyze (what is wrong with the model?)

Check properties of the model

Analyze the model

Build a Prototype

Analyze (what is wrong with the prototype?)

Get users to try it

Check properties of the model

Analyze the model

Build a Prototype

Analyze (what is wrong with the model?)

Check properties of the model

Analyze the model

Build a Prototype

Analyze (what is wrong with the prototype?)

Get users to try it

Check properties of the model

Analyze the model

Build a Prototype

Analyze (what is wrong with the model?)

Check properties of the model

Analyze the model

Build a Prototype

Analyze (what is wrong with the prototype?)

Get users to try it

Check properties of the model

Analyze the model

Build a Prototype

Analyze (what is wrong with the model?)

Check properties of the model

Analyze the model

Build a Prototype

Analyze (what is wrong with the prototype?)

Get users to try it

Check properties of the model

Analyze the model

Build a Prototype

Analyze (what is wrong with the model?)

Check properties of the model

Analyze the model

Build a Prototype

Analyze (what is wrong with the prototype?)

Get users to try it

Check properties of the model

Analyze the model

Build a Prototype

Analyze (what is wrong with the model?)

Check properties of the model

Analyze the model

Build a Prototype

Analyze (what is wrong with the prototype?)

Get users to try it

Check properties of the model

Analyze the model

Build a Prototype

Analyze (what is wrong with the model?)

Check properties of the model

Analyze the model

Build a Prototype

Analyze (what is wrong with the prototype?)

Get users to try it

Check properties of the model

Analyze the model

Build a Prototype

Analyze (what is wrong with the model?)

Check properties of the model

Analyze the model

Build a Prototype

Analyze (what is wrong with the prototype?)

Get users to try it

Check properties of the model

Analyze the model

Build a Prototype

Analyze (what is wrong with the model?)

Check properties of the model

Analyze the model

Build a Prototype

Analyze (what is wrong with the prototype?)

Get users to try it

Check properties of the model

Analyze the model

Build a Prototype

Analyze (what is wrong with the model?)

Check properties of the model

Analyze the model

Build a Prototype

Analyze (what is wrong with the prototype?)

Get users to try it

Check properties of the model

Analyze the model

Build a Prototype

Analyze (what is wrong with the model?)

Check properties of the model

Analyze the model

Build a Prototype

Analyze (what is wrong with the prototype?)

Get users to try it

Check properties of the model

Analyze the model

Build a Prototype

Analyze (what is wrong with the model?)

Check properties of the model

Analyze the model

Build a Prototype

Analyze (what is wrong with the prototype?)

Get users to try it

Check properties of the model

Analyze the model

Build a Prototype

Analyze (what is wrong with the model?)

Check properties of the model

Analyze the model

Build a Prototype

Analyze (what is wrong with the prototype?)

Get users to try it

Check properties of the model

Analyze the model

Build a Prototype

Analyze (what is wrong with the model?)

Check properties of the model

Analyze the model

Build a Prototype

Analyze (what is wrong with the prototype?)

Get users to try it

Check properties of the model

Analyze the model

Build a Prototype

Analyze (what is wrong with the model?)

Check properties of the model

Analyze the model

Build a Prototype

Analyze (what is wrong with the prototype?)

Get users to try it

Check properties of the model

Analyze the model

Build a Prototype

Analyze (what is wrong with the model?)

Check properties of the model

Analyze the model

Build a Prototype

Analyze (what is wrong with the prototype?)

Get users to try it

Check properties of the model

Analyze the model

Build a Prototype

Analyze (what is wrong with the model?)

Check properties of the model

Analyze the model

Build a Prototype

Analyze (what is wrong with the prototype?)

Get users to try it

Check properties of the model

Analyze the model

Build a Prototype

Analyze (what is wrong with the model?)

Check properties of the model

Analyze the model

Build a Prototype

Analyze (what is wrong with the prototype?)

Get users to try it

Check properties of the model

Analyze the model

Build a Prototype

Analyze (what is wrong with the model?)

Check properties of the model

Analyze the model

Build a Prototype

Analyze (what is wrong with the prototype?)

Get users to try it

Check properties of the model

Analyze the model

Build a Prototype

Analyze (what is wrong with the model?)

Check properties of the model

Analyze the model

Build a Prototype

Analyze (what is wrong with the prototype?)

Get users to try it

Check properties of the model

Analyze the model

Build a Prototype

Analyze (what is wrong with the model?)

Check properties of the model

Analyze the model

Build a Prototype

Analyze (what is wrong with the prototype?)

Get users to try it

Check properties of the model

Analyze the model

Build a Prototype

Analyze (what is wrong with the model?)

Check properties of the model

Analyze the model

Build a Prototype

Analyze (what is wrong with the prototype?)

Get users to try it

Check properties of the model

Analyze the model

Build a Prototype
Lecture 8:
Verification and Validation

• What are goals of V&V

• V&V Techniques
  ➢ Inspection
  ➢ Model Checking
  ➢ Prototyping
  ➢ Making Specifications Traceable
  ➢ Consistency Checking

• Independent V&V
Reviews, Walkthroughs, Inspections…

• **“Management reviews”**
  - E.g. preliminary design review (PDR), critical design review (CDR), …
  - Used to provide confidence that the design is sound
  - Attended by management and sponsors (customers)
  - Often just a “dog-and-pony show”

• **“Walkthroughs”**
  - developer technique (usually informal)
  - used by development teams to improve quality of product
  - focus is on finding defects

• ***These definitions are not widely agreed!***
  - Other terms used:
    - Formal Technical Reviews (FTRs)
    - Formal Inspections
  - **“Formality” can vary:**
    - **informal:**
      - meetings over coffee,
      - regular team meetings,
      - etc.
    - **formal:**
      - scheduled meetings,
      - prepared participants,
      - defined agenda,
      - specific format,
      - documented output
Benefits of formal inspection

• Formal inspection works well for programming:
  - **For applications programming:**
    - more effective than testing
    - most reviewed programs run correctly first time
  - **Data from large projects**
    - error reduction by a factor of 5; (10 in some reported cases)
    - improvement in productivity: 14% to 25%
    - percentage of errors found by inspection: 58% to 82%
    - cost reduction of 50%-80% for V&V (even including cost of inspection)
  - **Effects on staff competence:**
    - increased morale, reduced turnover
    - better estimation and scheduling (more knowledge about defect profiles)
    - better management recognition of staff ability

• These benefits also apply to requirements inspections
  - Many empirical studies investigated variant inspection processes
  - Mixed results on the relative benefits of different processes
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
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 or Evolve?

- **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
Lecture 8: Verification and Validation

• What are goals of V&V

• V&V Techniques
  ➢ Inspection
  ➢ Model Checking
  ➢ Prototyping
  ➢ Making Specifications Traceable
  ➢ Consistency Checking

• Independent V&V
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
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?
- Does every class have methods to get/set its attributes?
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
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

**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?
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
Lecture 8: Verification and Validation

• What are goals of V&V

• V&V Techniques
  ➢ Inspection
  ➢ Model Checking
  ➢ Prototyping
  ➢ Making Specifications Traceable
  ➢ Consistency Checking

• Independent V&V
Independent V&V

• **V&V performed by a separate contractor**
  - Independent V&V fulfills the need for an independent technical opinion.
  - Cost between 5% and 15% of development costs
  - Studies show up to fivefold return on investment:
    - Errors found earlier, cheaper to fix, cheaper to re-test
    - Clearer specifications
    - Developer more likely to use best practices

• **Three types of independence:**
  - Managerial Independence:
    - separate responsibility from that of developing the software
    - can decide when and where to focus the V&V effort
  - Financial Independence:
    - Costed and funded separately
    - No risk of diverting resources when the going gets tough
  - Technical Independence:
    - Different personnel, to avoid analyst bias
    - Use of different tools and techniques
Lecture 8: Verification and Validation

- What are goals of V&V
- Validation Techniques
  - Inspection
  - Model Checking
  - Prototyping
- Verification Techniques
  - Consistency Checking
  - Making Specifications Traceable
- Independent V&V