EC3290 – Software Requirements Engineering

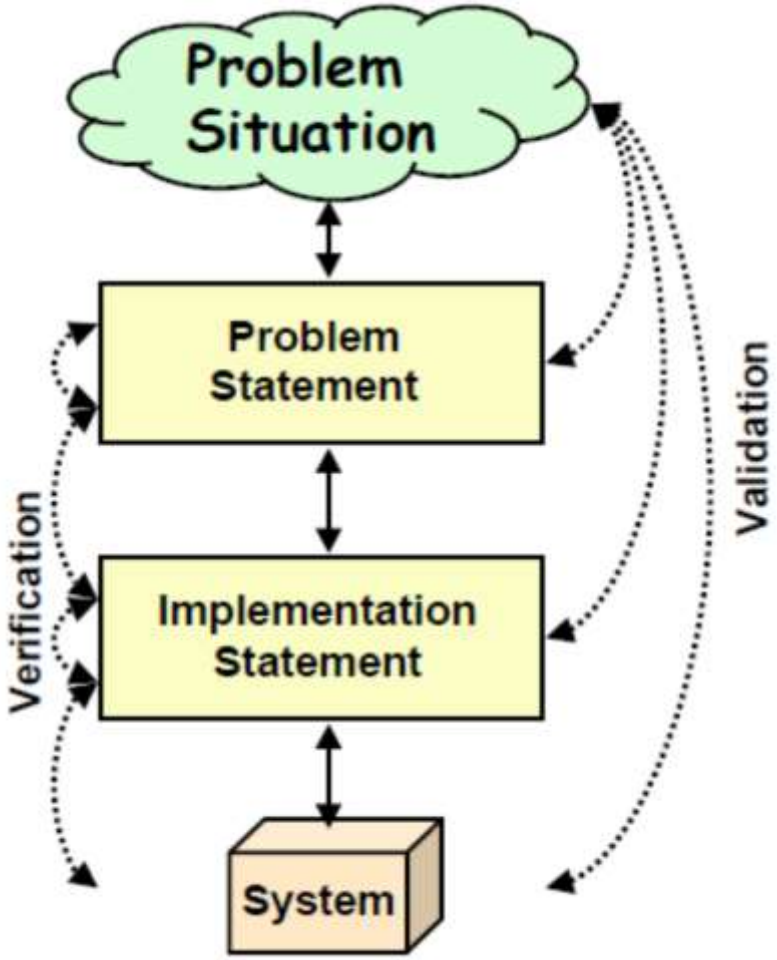
Topic 4 – Software Verification and Validation

Learning Objectives

- Understand the difference between verification and validation
- Explore techniques used in requirement V&V
- Learn how to identify conflicts, inconsistencies, and incompleteness in requirements

What is Requirement

- **Verification**
  - Check if the requirement document is correct and logically structured.
  - Are we building the product right?
- **Validation**
  - Confirm if the documented requirements truly reflect the needs of stakeholders.
  - Are we building the right product?



Importance of V&V

- Reduces costly rework in later stages
- Ensures customer satisfaction
- Improves software quality

- **Example:** NASA lost a $125 million spacecraft because of a requirements error. That's one expensive typo.

Example 1

YAHOO! MAIL

- Incident: Yahoo! mail doesn't let me log in
- Failure: The user account cannot be accessed in the user database.
- Fault: The user database can not be reached.
- Error: There was no backup user database in the system.

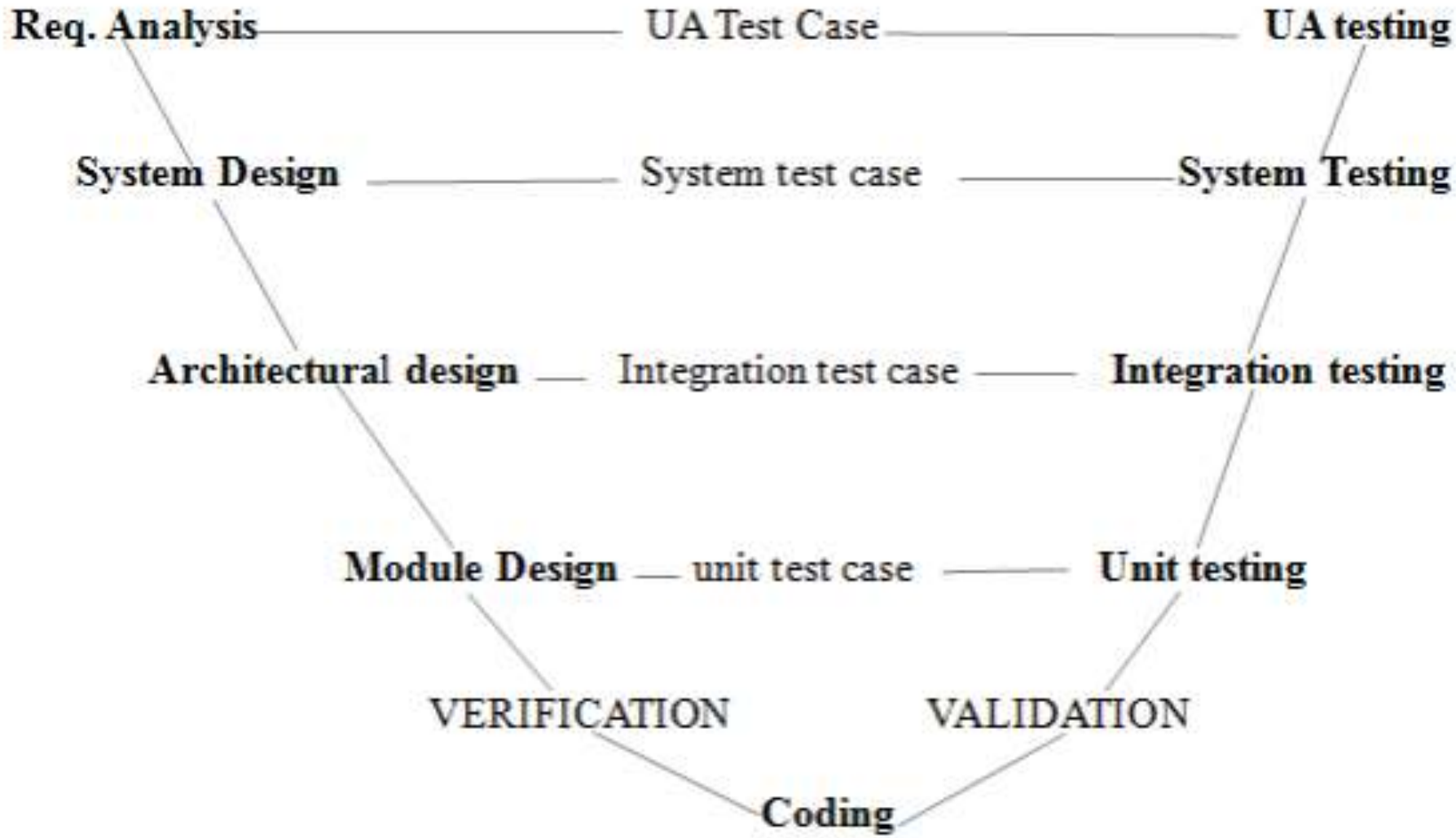
Example 2

- Fatal Therac-25 X-ray Radiation
- In 1986, a man in Texas received between 16,500-25,000 radiations in less than 10 sec, over an area of about 1 cm.
- He passed away 5 months later.
- The root cause of the incident was a SW failure ☹

- Incident: A patient passed away
- Failure: The device applied higher frequency of radiations than what was safe. Safety range: [1…10,000 Hz].
- Fault: The software controller of the device did not have a conditional block (if …. else statements) to perform range checking on the frequency of the radiation to be applied.
- (2) Errors:
  1. The SW developer of the device controller system had forgotten to include a range checking conditional block on the frequency of the radiation to be applied.
  2. The device operator was NOT supposed to enter anything outside [1…10,000 Hz] range.

When Does V&V Happen?

- Throughout the software development life cycle (SDLC)
- Especially during and after requirements elicitation and specification
- Before design and implementation begin


Req. Analysis
UA Test Case
UA testing
System Design
System test case
System Testing
Architectural design
Integration test case
Integration testing
Module Design
unit test case
Unit testing
VERIFICATION
VALIDATION
Coding

Verification: In-Depth

- Ensures the software meets the **specified** requirements
- Focuses on correctness and consistency
- Techniques: Reviews, Walkthroughs, Inspections

Validation: In-Depth

- Ensures the software fulfills **user needs and expectations**
- Involves stakeholders
- Techniques: Prototyping, User Feedback, Acceptance Testing

The V&V Relationship

- **Verification = Internal check**
- **Validation = External check**
- Both are essential; one without the other can lead to disaster

Verification

- **Definition**
  Verification refers to the set of activities that ensure software correctly implements the specific function.
- **Focus**
  It includes checking documents, designs, codes, and programs.

VS

Validation

- **Definition**
  Validation refers to the set of activities that ensure that the software that has been built is traceable to customer requirements.
- **Focus**
  It includes testing and validating the actual product.

V&V Activities in Requirements Phase

- Requirements review
- Consistency checks
- Prototyping for validation
- Traceability analysis

Inspection Techniques

- Formal, peer-based examination of documents
- Checklist-driven
- Focus on:
  - Ambiguities
  - Conflicts
  - Incompleteness

Walkthroughs

- Informal group discussion
- Presenter leads; others provide feedback
- Goal: Understand and evaluate the requirements

Reviews

- Technical review meetings
- Used to detect errors early
- May be formal or informal

- **Example:** Weekly review sessions with client or internal QA team.

Prototyping for Validation

- Build a mock-up of the system or interface
- Gather feedback from users
- Helps discover hidden needs

- Because "I'll know it when I see it" is not a requirement.

Acceptance Criteria and Testing

- Define what it means for a requirement to be "done"
- Involve end-users in writing test cases
- Must be measurable and testable

Conflict Detection in Requirements

- **Conflict:** Two or more requirements contradict each other
- **Example:** One requirement says “System must run in 2 seconds,” another says “Must perform deep analysis.”
- **Solution:** Use traceability and conflict matrices.

Inconsistency Detection

- Different parts of the requirement contradict or don’t align
- **Example:** “All users must log in” vs. “Guest users can access without login”
- **Technique:** Cross-checking related requirements.

Completeness Check

- Ensures all required functionalities are described
- No missing constraints or conditions

- **Checklist:** Are all stakeholders' needs addressed? Are all interfaces defined?

Ambiguity Detection

- Ambiguous: “The system should be fast”
- Clear: “System response time shall not exceed 2 seconds under load”

Traceability Matrix

Maps requirements to:

- Stakeholder needs
- Design components
- Test cases
  Benefit: Helps detect missing or extra functionality
  Example table: Req1 → Use Case 1 → Module A → Test Case 5

Requirements Traceability Matrix

Project Name	<Project Name here>	Created On	3-Oct-11	Reviewed On	4-Oct-11
Release No	<Project Release>	Created By	<Creater's Name>	Reviewed By	<Reviewer's Name>
Version	<Doc version>				

ID	Requirement ID	Requirement Description	Status	Design Document	Code Module	TestCase ID	Test Case Name	User Manual	Tested On/ Verification
001	UC 1.0	Testing Requirement Description here. It should not be more than 2-3 lines	Status	DM-001	CM-001	TC-001	ProjName_UCID_TestCase Name	Section 4.5	Pending
002	UC 1.1	Testing Requirement Description here. It should not be more than 2-3 lines	Approved	DM-002	CM-002	TC-002 TC-003	N.A	Section 4.6	Verified
003	UC 1.2	Testing Requirement Description here. It should not be more than 2-3 lines	Status	DM-003	CM-003	TC-004 TC-006 TC-007 TC-008		Section 5.7	Verified
004	UC 1.3	Testing Requirement Description here. It should not be more than 2-3 lines	Approved	DM-004	CM-004			Section 6.8	In-progress
005	UC 1.4	Testing Requirement Description here. It should not be more than 2-3 lines	Approved	DM-005	CM-005			Section 7.9	Not Verified
006	UC 1.5	Testing Requirement Description here. It should not be more than 2-3 lines	TBD	DM-006	CM-006			Section 4.10	Verified
007	UC 1.6	Testing Requirement Description here. It should not be more than 2-3 lines	Approved					Section 4.11	Not Verified
008	UC 1.7	Testing Requirement Description here. It should not be more than 2-3 lines	TBD					Section 4.12	Pending
009	UC 1.8	Testing Requirement Description here. It should not be more than 2-3 lines	TBD					Section 4.13	Not Verified
010	UC 1.9	Testing Requirement Description here. It should not be more than 2-3 lines	Approved					Section 4.14	Pending

Tool Support for V&V

Tools like:

- IBM DOORS
- ReqView
- JIRA (for issue tracking)

- Help automate conflict detection and versioning

V&V in Agile Context

- Continuous validation via sprints and reviews
- Definition of Done (DoD) ensures verification
- User stories and acceptance criteria help validate

Common Mistakes in V&V

- Skipping reviews
- Relying too much on tools
- Not involving users in validation

Summary

- V&V ensure correct and complete requirements
- Verification = “Did we build it right?”
- Validation = “Did we build the right thing?”
- Use a variety of techniques: inspection, prototyping, testing