Lecture 13

➔ (Week 13 - March 1, 2024)

Definitions:

- **D (Domain Properties)**: Things in the Application Domain (AD) that are true anyway.
- **R (Requirements)**: Things in the Application Domain (AD) that we wish to be made true.
- **S (Specification)**: Description of the behaviors the program must have in order to meet the requirements.

Verification Criteria:

1. **Program Correctness**: Ensures that the program running on a specific computer satisfies the specification.
2. **Specification Validity**: Ensures that the specification, given the domain properties, satisfies the requirements.

Validation Criteria:

1. **Requirement Discovery**: Ensures that all important requirements are discovered and understood.
2. **Domain Property Discovery**: Ensures that all relevant domain properties are discovered and understood.

Techniques for Validation and Verification:

- **Prototyping**: Creating working models of the system to validate requirements.
- **Informal Walkthrough**: Reviewing the requirements informally to identify errors, inconsistencies, or missing elements.
- **Formal Inspection/Walkthrough**: A structured review process to identify errors and ensure completeness.
- **Animation**: Using animated representations to validate system behaviors.
- **Traceability**: Ensuring that requirements are linked to specific system components and vice versa.
- **Testing**: Executing the system with test cases to validate its behavior.
- **Model-based Verification and Validation (V&V)**: Using formal models to verify system behaviors.

Early vs. Late Validation:

- **Early Validation**: Using scenarios, mock-ups, and prototypes to validate requirements before formalization.
- **Late Validation**: Formal reviews and inspections to identify errors, inconsistencies, and incompleteness in the specifications.

Strategies for Validation:

- **Scenarios and Interviews**: Using scenarios and semi-structured interviews with users to validate requirements.
- **Storyboards**: Creating visual representations of system behaviors to elicit user feedback.
- **Active, Passive, and Interactive Storyboards**: Different approaches to presenting system behaviors for validation.

Purpose of Validation Techniques:

- **Gain Early Reaction**: Obtain feedback from users and stakeholders early in the development process.
- **Identify Issues**: Identify errors, inconsistencies, and missing requirements.
- **Elicit User Feedback**: Encourage user input and engagement in the validation process.

Requirements Validation Issues:

Requirements Clarification:

- Poorly expressed or accidentally omitted information during elicitation.
- Need to gather missing information.

Missing Information:

- Some information is not present in the requirements document.
- Need to identify and collect the missing details.

Requirements Conflict:

- Significant conflicts exist between requirements.
- Stakeholders must negotiate to resolve conflicts.

Unrealistic Requirement:

- Some requirements seem implausible given the available technology or system constraints.
- Stakeholders need to be consulted to make the requirement more realistic.

Validation Techniques and Processes:

Management Reviews:

- E.g., preliminary design review (PDR), critical design review (CDR).
- Aimed at providing confidence in the design's soundness.
- Often attended by management and sponsors.

Walkthroughs:

- Informal technique used by development teams to improve product quality.
- Focus on finding defects.

Fagan Inspections:

- Formal process management tool.
- Used to improve the quality of the development process.
- Involves collecting defect data and analyzing process quality.

Formal Inspection vs. Walkthrough:

Formal Inspection:

- Initiated by the project team.
- Planned, with a defined agenda and specific format.
- Author is not the presenter.
- Frequently uses checklists.

Walkthrough:

- Initiated by the author.
- Semi-formal, often poorly planned.
- Involves leading the development team through a segment of the artifact, with questions and comments.

Benefits of Formal Inspection:

- Works well for programming and application development.
- More effective than testing, leading to fewer errors.
- Significant productivity improvement.
- Reduction in costs for verification and validation.
- Organizational benefits such as increased morale and better estimation.

Key Aspects of Inspection:

- **Size**: 3 to 7 reviewers, ensuring all relevant expertise is available.
- **Duration**: Never more than 2 hours to maintain concentration.
- **Outputs**: All reviewers must agree on results, with detailed documentation.
- **Scope**: Focus on a small part of the document/design.
- **Timing**: Review once the author has finished the product but not too soon or too late.
- **Roles**: Review leader, recorder, reader, author, and other reviewers.

Participant Selection:

- **Include**: Specialists, team members, invited experts, interested parties.
- **Exclude**: Line managers, individuals with personality clashes or conflicts of interest.

Review Techniques:

- **Checklist**: Structured review using predefined questions/issues.
- **Walkthrough**: Step-by-step presentation of the document.
- **Round Robin**: Each reviewer raises an issue in turn.
- **Speed Review**: Rapid assessment of document comprehensibility.

Model Checking:

- **Used for**:
    - Behavioral workflow (process) and state machine models.
    - Utilizes reachability analysis approach.
- **Typical Properties to Verify**:
- **General Properties**:
        - Absence of deadlocks in systems with concurrency.
    - All states can be reached and all transitions can be traversed.
- **Specific Properties**:
        - Logic assertions or invariants.
    - Temporal logic (predicate logic extension with operators like 'always' and 'eventually').

Basic Checks for Model Verification:

- **Syntactic/Semantic Check**:
    - E.g., UML Class diagrams.
    - Ensure correct representation of classes, associations, and specializations.
- **Cross Check Amongst Diagrams**:
    - Ensure consistency among diagrams, especially regarding class/object relationships.
- **Basic Checks**:
    - Validate consistency between various diagrams such as Class, Use Case, and System Sequence Diagrams (SSDs).
    - Ensure all relevant components are represented in the appropriate diagrams.

Documentation Requirements:

- **Includes**:
    - Date, numbered pages, list of topics, change and version control.
    - Process representation with a numbered circle (or rounded rectangle).
- **Characteristics**:
    - Identifier should begin with a verb.
    - Limited number of processes (ideally 7 +/- 2).
    - Avoid black holes or miracles.
    - Ensure model balance.

Specific Checks for Diagrams:

- **Use Case Diagrams**:
    - Validate presence of users for each use case.
    - Ensure documentation for each use case.
- **Class Diagrams**:
    - Confirm capture of all mentioned classes/entities.
    - Check existence of methods to get/set attributes for each class.
- **Sequence Diagrams**:
    - Validate presence of associated classes and methods.
    - Confirm association between sender and receiver classes.
- **StateChart Diagrams**:

    - Verify consistency with class diagram.
    - Ensure clear trigger events and transitions.

Traceability:

- **IEEE-STD-830**:
    - Backwards traceability: Origin of each requirement.
    - Forward traceability: Linking requirements to design/implementation artifacts.
- **DOD-STD-2167A**:
    - Specifications are traceable if they implement all stipulations from predecessor documents and maintain consistency.
- **Requirements Management**:
    - Utilize traceability techniques to ensure coverage of elicitation notes and traceability between different levels of requirements.

Types of Traceability:

- **Vertical Traceability**:
    - Capture relationships across different types of artifacts.
- **Horizontal Traceability**:
    - Relationships between artifacts of the same type.

Pre and Post Traceability:

- **Pre-Traceability**:
    - Understanding the origin and rationale before adding to the requirements document.
- **Post-Traceability**:
    - Understanding how it's implemented and linking it back to the requirements.

Explicit Traceability:

- **Definition**:
    - Links elements not intrinsically linked.
    - Relationships are developed based on external considerations given by team members.
    - Implemented specifically through links or indicators.
- **Examples**:
    - Explicitly linking scenarios with class diagrams.
    - Indicating how requirements relate to Java code.

Implicit Traceability:

- **Definition**:
    - Inherent in the nature of the artifacts.
    - Examples include processes linked within a Data Flow Diagram (DFD).

Model Cross-Consistency Rules:

- **Definition**:
    - Rules ensuring consistency among different models.
    - Not explicitly represented but maintained implicitly.

Internal vs. External Traceability:

- **Internal**:
    - Relationships between artifacts of the same type.
    - E.g., scenarios within a use case.
- **External**:
    - Relationships between different artifacts.
    - E.g., scenarios and class diagrams.

Managing Changes:

- **Natural Occurrence**:
    - Changes in requirements during development process.
    - Reasons include missing requirements, business changes, fixes, and new needs.
- **Impact on Artifacts**:
    - Requirements changes may impact design, implementation, and testing.
    - Top-down propagation and bottom-up analysis of changes.

Importance of Traceability:

- **For Verification**:
    - Assisting in preparing test suite and assessing conformance to requirements.
    - Ensuring completeness, consistency, and impact analysis.
- **For Maintenance**:
    - Assessing and implementing change requests.
    - Tracing design rationale and ensuring document access.
- **For Management**:
    - Facilitating change and risk management.
    - Enhancing project and development control.

Benefits of Traceability:

- **Prevents Knowledge Loss**.
- **Supports Verification Process**.
- **Facilitates Impact Analysis**.
- **Enhances Software Quality**.
- **Aids in Reengineering and Reuse**.
- **Reduces Risks**.

Challenges of Traceability:

- **Expensive**:
    - Requires tool support and constant maintenance.
- **Size and Diversity**:
    - Diverse elements to be traced, necessitating continuous effort.

Independent Verification and Validation (V&V):

- **Definition**:
    - Performed by a separate contractor to ensure independent technical opinion.
- **Types of Independence**:
    - Managerial, Financial, and Technical.

Validation vs. Verification:

- **Validation**:
    - Ensures the right problem is being solved.
    - Utilizes techniques like prototyping, inspection, and formal analysis.
- **Verification**:
    - Ensures engineering steps are sound.
    - Involves consistency checking, traceability, and appropriate V&V methods.