Requirements Validation Review

- **What is** requirements validation**, and why is it important in software engineering?**

  Requirements validation is the process of checking gathered and documented requirements to ensure correctness. It is important because errors in the requirements document lead to costly rework when discovered during development or after deployment.

Requirements Validation Review

- **Why can incorrect requirements cause major problems later in a project?**

  Because discovering requirement errors after development starts or after system deployment requires redesigning, re-coding, re-testing, and may affect the schedule and budget.

Requirements Validation Review

- **Which type of check ensures that no contradictory or conflicting requirements exist?**

  Consistency checks.

- **A user requests a feature, but deeper analysis shows additional hidden functions are needed. Which validation check detects this?**

  Validity checks.

Requirements Validation Review

- **What does a completeness check ensure in the requirements document?**

  It ensures all required functions and constraints are correctly defined and included.

- **What does** realism checking **involve, and what two real-world constraints must be considered?**

  It checks if requirements can actually be implemented using existing technology. It must consider **budget** and **schedule** constraints.

Requirements Validation Review

- **NOTE:** Verifiability means requirements should be testable and described in a way that allows writing tests.
- **Why must system requirements always be verifiable?**
  To avoid disputes between customer and contractor and to ensure that every requirement can be tested after implementation.
- **How does requirements validation relate to requirements modeling?**
  Validation checks if requirements are correct; modeling represents these validated requirements using different diagrams or models.

Software Validation

- **1. Software Testing & V&V**
- Software testing is the process of running a program with the goal of finding errors. It is part of the broader domain of **Verification and Validation (V&V)**.
- **Verification**: Ensures the software correctly implements specific functions.
  "Are we building the product right?"
- **Validation**: Ensures the software meets customer needs and requirements.
  "Are we building the right product?"

Software Validation

- **2. Attributes of a “Good” Test**

- A high-quality test should have:

1. **High probability of finding errors**.
2. **Non-redundancy:** avoid repeating the same test unnecessarily.
3. **Best of breed:** from similar tests, choose the one most likely to uncover errors.
4. **Balanced complexity:** not too simple and not overly complex; executed independently.

Software Validation

- **3. Software Quality**

  Software quality is evaluated using three main groups of factors:

- **A- Product Operation Factors**

1- Correctness: How well the program satisfies requirements and mission goals.

2- Reliability: The ability of the program to perform its intended function accurately over time.

3- Usability: Effort required to learn, operate, and interpret the system.

4- Integrity: Protection against unauthorized access or misuse of data.

5- Efficiency: How well the program uses computing resources and code to carry out its function.

Software Validation

- **B. Product Revision Factors**

These describe how easy it is to modify or improve the software:

6. **Maintainability**: Effort required to find and fix errors.
7. **Flexibility**: Ease of modifying the software for new conditions.
8. **Testability**: Effort required to test the system and verify behavior.

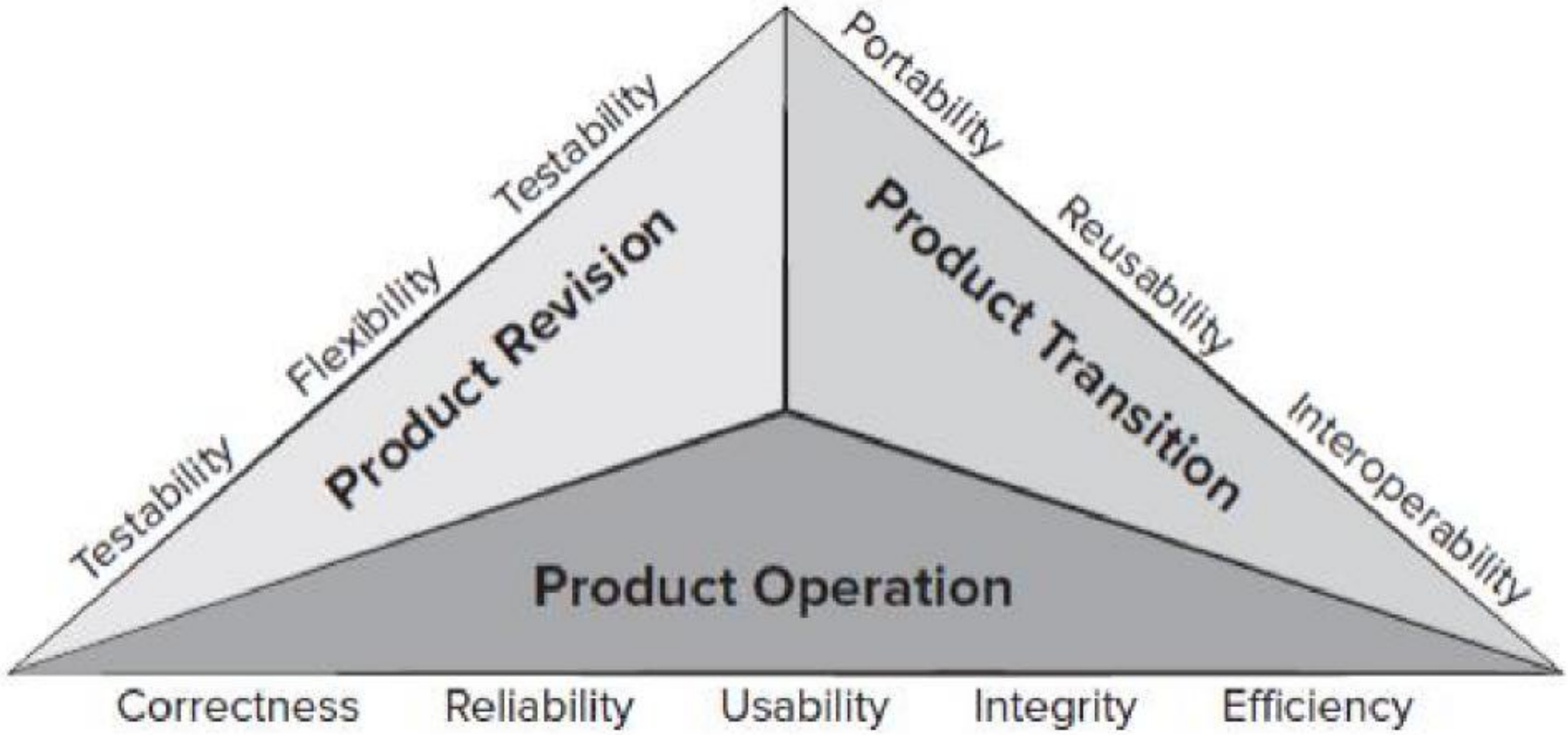
Software Validation

- **C. Product Transition Factors**

These describe how easily the software adapts to new environments:

9. **Portability**: Ease of moving the software to another platform.
10. **Reusability**: Ability to reuse parts of the system in other systems.
11. **Interoperability**: Ability of the software to work with other systems.

Quality Factors



Software Validation

- **2. Formal Technical Review (FTR)**
- A **Formal Technical Review** is a structured quality control activity performed by software engineers.
- **Objectives of FTR:**

1. **Identify errors** in logic, function, or implementation.
2. **Verify requirement compliance:** ensure the software meets documented needs.

Software Validation

- **What are the three major categories of software quality factors?**

  Product Operation, Product Revision, Product Transition.

- **What does integrity measure in software quality?**

  The extent to which unauthorized access to software or data is prevented.

- **Which quality factor describes how much effort is needed to find and correct errors?**

Maintainability.

- **Why is portability important in software systems?**

It measures the ease of moving a program between different hardware or software environments.

Software Validation

- **Reusability refers to using the software in the exact same environment.**

  Correct: It refers to using the program (or parts of it) in *other* applications.

- **What does interoperability mean?**

  The effort required to connect and operate software with other systems.

- **What is the main purpose of a Formal Technical Review (FTR)?**

  To uncover errors and verify that the software meets its requirements.

Why is testability an important quality factor?

Because tests confirm that the software performs its intended function and meets quality standards.

Software Validation

- A **Formal Technical Review (FTR)** is a structured software quality assurance activity performed by software engineers to evaluate the correctness and completeness of software documents or code.
  Its main goals are:
- **Detect errors early** in function, logic, requirements, or implementation.
- **Verify compliance** with system requirements.
- **Ensure the software follows predefined standards** (coding, documentation, design).
- **Promote uniformity** across the development team so all artifacts use consistent styles and formats.
- **Improve project manageability**, reducing rework, cost, and schedule delays.
- FTR is important because it prevents defects from moving to later stages of development, where they become more expensive and harder to fix.

Software Validation

- **What is the main purpose of a Formal Technical Review (FTR)?**

  To uncover errors and verify that the software meets its requirements.
- **Why is testability an important quality factor?**
  Because tests confirm that the software performs its intended function and meets quality standards.

Software Validation

- **Which factor relates to ease of learning and operating a program?**

  Usability.

- The ability of a system to work with another system is called **interoperability**.

- **Why is maintainability important?**

It shows how easy it is to find and fix errors in the program.

- **What is portability?**

The effort required to move the software to another hardware or software environment.

Software Validation

- **Objectives of a Formal Technical Review**
- **1. Uncover Errors in Function, Logic, or Implementation**

  - FTR helps detect:

- Functional mistakes
- Logical inconsistencies
- Incorrect implementation
- Missing or ambiguous requirements
- Misinterpretation of specifications

→ This ensures the software model or code is technically sound.

Software Validation

2- Verify That the Software Meets Its Requirements

- An FTR checks whether:

- The representation correctly reflects the documented requirements
- The design aligns with functional and non-functional requirements
- The implementation satisfies the intended system behavior

→ This aligns product evolution with stakeholder expectations.

Software Validation

3- Ensure Compliance With Predefined Standards

- FTR ensures the software or document:

- Follows organizational coding standards
- Uses standard naming conventions
- Applies architectural patterns correctly
- Adheres to industry best practices and IEEE/ISO standards

→This improves maintainability and consistency.

Software Validation

4- Achieve Uniformity Across the Development Team

- Formal reviews enforce:

- Standardized documentation style
- Consistent design notations (UML, ERD, DFD, etc.)
- Unified coding practices

→ This reduces confusion and improves collaboration.

Software Validation

5- Make Projects More Manageable

- By discovering errors early and enforcing uniformity, FTR helps:

- Reduce rework
- Minimize schedule slippage
- Improve quality of project planning
- Enable more reliable cost estimation

→ Overall, FTR increases project stability and predictability.