Thakur Educational Trust's (Regd.)

THAKUR RAMNARAYAN COLLEGE OF ARTS & COMMERCE

ISO 21001:2018 Certified

PROGRAMME: B.Sc (I.T)

CLASS: S.Y.B.Sc (I.T)

SUBJECT NAME: SOFTWARE ENGINEERING

SEMESTER: IV

FACULTY NAME: Ms. SMRITI DUBEY

UNIT IV

Chapter 1 – Verification and Validation

Concepts:

Introduction

Planning Verification and Validation

Software Inspection

Automated Static Analysis

Cleanroom Software Development

Introduction

In software project management, software testing, and software engineering, **verification and validation (V&V) is the process of checking that a software system meets specifications and requirements so that it fulfills its intended purpose**. It may also be referred to as software quality control. It is normally the responsibility of software testers as part of the software development lifecycle.

Verification and Validation is the process of investigating that a software system satisfies specifications and standards and it fulfills the required purpose. **Barry Boehm described verification and validation as the following:**

Verification: Are we building the product, right?
Validation: Are we building the right product?

Verification:

Verification is the process of checking that a software achieves its goal without any bugs. It is the process to ensure whether the product that is developed is right or not. It verifies whether the developed product fulfills the requirements that we have. **Verification is Static Testing**.

Activities involved in verification:

1. Inspections
2. Reviews
3. Walkthroughs
4. Desk-checking

Validation:

Validation is the process of checking whether the software product is up to the mark or in other words product has high level requirements. It is the process of checking the validation of product i.e., it checks what we are developing is the right product. it is validation of actual and expected product. **Validation is the Dynamic Testing**.

Activities involved in validation:

1. Black box testing
2. White box testing
3. Unit testing
4. Integration testing

The role of Verification involves checking that the software contains to its specifications. We should check that it meets its specified functional and non – functional requirements. **The aim of Validation is to ensure that the software system meets the customers expectations**. It goes beyond checking that the system conforms to its specification to showing that the software does what the customer expects it to do. **The ultimate goal of the verification and validation is to establish confidence that the software system is "fit for purpose".**

Difference between verification and validation testing

Verification	Validation
We check whether we are developing the right product or not.	We check whether the developed product is right.
Verification is also known as static testing.	Validation is also known as dynamic testing.
Verification includes different methods like Inspections, Reviews, and Walkthroughs.	Validation includes testing like functional testing, system testing, integration, and User acceptance testing.
It is a process of checking the work-products (not the final product) of a development cycle to decide whether the product meets the specified requirements.	It is a process of checking the software during or at the end of the development cycle to decide whether the software follow the specified business requirements.
Quality assurance comes under verification testing.	Quality control comes under validation testing.
The execution of code does not happen in the verification testing.	In validation testing, the execution of code happens.
In verification testing, we can find the bugs early in the development phase of the product.	In the validation testing, we can find those bugs, which are not caught in the verification process.
Verification testing is executed by the Quality assurance team to make sure that the product is developed according to customers' requirements.	Validation testing is executed by the testing team to test the application.
Verification is done before the validation testing.	After verification testing, validation testing takes place.
In this type of testing, we can verify that the inputs follow the outputs or not.	In this type of testing, we can validate that the user accepts the product or not.

Planning Verification and Validation

Verification and Validation processes are intended to establish the existence of defects in a software system. Verification and Validation is an:

1. Expensive process : Sometimes half of the system development budget may be spent on V&V

2. Careful planning is needed to get the most out of the inspections and testing and to control the cost of verification and validation process.

3. We should start planning system validation and verification early in the development process model. The software development process model sometimes called the V model is an instantiation of the generic waterfall model that test plans should be delivered from the system specialization and design.

4. Test planning is concerned with establishing standards for the testing process not just with describing/producing tests. It helps managers allocate resources and estimate testing schedules.

5. The test plans are intended for software engineer involved in designing and carrying out system tests. The structure of the test plan is as follows:

Test plans obviously vary, depending on the project and the organization involved in the testing. Sections that would typically be included in a large system, are:

The testing processes
A description of the major phases of the system testing process. This may be broken down into the testing of individual sub-systems, the testing of external system interfaces, etc.

Requirements traceability
Users are most interested in the system meeting its requirements and testing should be planned so that all requirements are individually tested.

Tested items
The products of the software process that are to be tested should be specified.

Testing schedule
An overall testing schedule and resource allocation. This schedule should be linked to the more general project development schedule.

Test recording procedures
It is not enough simply to run tests; the results of the tests must be systematically recorded. It must be possible to audit the testing process to check that it has been carried out correctly.

Hardware and software requirements
This section should set out the software tools required and estimated hardware utilization.

Constraints
Constraints affecting the testing process such as staff shortages should be anticipated in this section.

System tests
This section, which may be completely separate from the test plan, defines the test cases that should be applied to the system. These tests are derived from the system requirements specification.

Software Inspections

Inspections are a formal type of review that involves checking the documents thoroughly before a meeting and is carried out mostly by moderators. A meeting is then held to review the code and the design.

Inspection meetings can be held both physically and virtually. The purpose of these meetings is to review the code and the design with everyone and to report any bugs found.

Software inspections involve people examining the source representation with the aim of discovering anomalies and defects. Inspections not require execution of a system so may be used before implementation. They may be applied to any representation of the system (requirements, design, configuration data, test data, etc.). They have been shown to be an effective technique for discovering program errors.

Inspection Roles

Various roles involved in an inspection are as follows:

Author or owner: The author is a programmer or designer who is responsible for producing the program or documents. Responsible for fixing defects discovered during the inspection process.

Inspector: Inspector provides review comments for the code. Finds error, omissions, and inconsistencies bin the program. May also identify the broader issues that are outside the scope of the inspection team.

Moderator or chairman: Moderator formally run the inspection according to process. Manages the process and facilitates the inspection. Also reports process results to the chief moderator.

Scribe: Scribe notes the inspection meeting results and circulates them to the inspection team after the meeting.

Reader: The reader presents the code or document at an inspection meeting.

Chief moderator: Chief moderator is responsible for inspection process improvement, checklist updating, standard development, etc.

Inspection Process

The inspection team moderator is responsible for inspection planning. This involves selecting an inspection team, organizing a meeting room and ensuring that the material is inspected and its specification are complete.

The program to be inspected is presented to the inspection team during the overview stage when the author of the code describes what the program is intended to do. This is followed by a period of individual preparation. Each inspection team member studies the specification and the program and looks for defects in the code.

The inspection itself should be fairly short (no more than two hours) and should focus on defect detection, standards conformance and poor quality programming. The inspection team should not suggest how these defects should be corrected nor should it recommend changes to other components.

Following the inspection the program author should make changes to correct identified problems. The time needed for an inspection and the amount of code that can be covered depends on the experience of the inspection team, the programming language and the application domain.

Advantages of inspections include:

1. During testing, errors can mask (hide) other errors. Because inspection is a static process, you don't have to be concerned with interactions between errors.

2. Incomplete versions of a system can be inspected without additional costs. If a program is incomplete, then you need to develop specialized test harnesses to test the parts that are available.

3. As well as searching for program defects, an inspection can also consider broader quality attributes of a program, such as compliance with standards, portability and maintainability.

Automated Static Analysis

Static analysis, also called static code analysis, is a method of computer program debugging that is done by examining the code without executing the program. The process provides an understanding of the code structure and can help ensure that the code adheres to industry standards.

Static analysis is used in software engineering by software development and quality assurance teams. Automated tools can assist programmers and developers in carrying out static analysis. The software will scan all code in a project to check for vulnerabilities while validating the code.

Static analysis is generally good at finding coding issues such as:

1. Programming errors
2. Coding standard violations
3. Undefined values
4. Syntax violations
5. Security vulnerabilities

How is static analysis done?

The static analysis process is relatively simple, as long as it's automated. Generally, static analysis occurs before software testing in early development. In the DevOps development practice, it will occur in the create phases.

Once the code is written, a static code analyzer should be run to look over the code. It will check against defined coding rules from standards or custom predefined rules. Once the code is run through the static code analyzer, the analyzer will have identified whether or not the code complies with the set rules.

It is sometimes possible for the software to flag false positives, so it is important for someone to go through and dismiss any. Once false positives are waived, developers can begin to fix any apparent mistakes, generally starting from the most critical ones. Once the code issues are resolved, the code can move on to testing through execution.

Types of static analysis

There are several static analysis methods an organization could use, which include:

1. **Control flow analysis** – This stage identifies and highlights loops with multiple exits on entry points and unreachable code.

2. **Data use analysis** – This stage highlights how variables in the program are used.

3. **Information analysis** – This phase of the analysis identifies the dependencies between input and output variables it shows how the value of each program variable is derived from other variable value.

4. **Interface analysis** – This analysis checks the consistency of routine and procedure declarations and their use.

5. **Path analysis** – The phase of semantic analysis identifies all possible paths through the program and sets out the statements executed in that path.

Cleanroom Software Development

Cleanroom software development is a software development philosophy that uses formal methods to support rigorous software inspection. The objective of this approach to software development is zero defect software.

A model of the Cleanroom process, adapted from the description given by Linger (Linger, 1994), is shown below. This shows how these essential strategies are integrated.

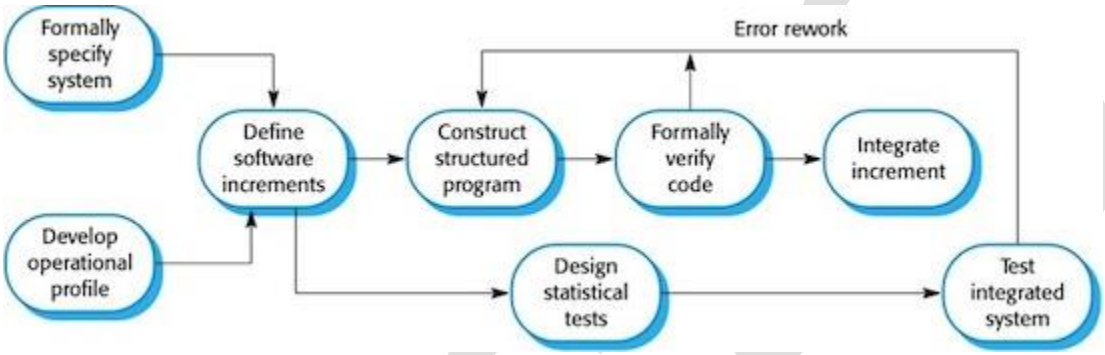

Figure 1: The Cleanroom process

Formal specification of the system is verified on incremental approach. The approach to incremental development in the Cleanroom process is to deliver critical customer functionality in early increments. Less important system functions are included in later increments. The customer therefore has the opportunity to try these critical increments before the whole system has been delivered.

If requirements problems are discovered the customer feeds back this information to the development team and requests a new release of the increment. As new increments are developed they are combined with the existing increments and the integrated system is tested. Rigorous program inspection is a fundamental part of the Cleanroom process.

The Cleanroom approach to software development is based on five key strategies:

1. **Formal specification**: The software to be developed is formally specified. A state-transition model which shows system responses to stimuli is used to express the specification.

2. Incremental development: The software is partitioned into increments which are developed and validated separately using the Cleanroom process. These increments are specified, with customer input, at an early stage in the process.

3. Structured programming: Only a limited number of control and data abstraction constructs are used. The program development process is a process of stepwise refinement of the specification. A limited number of constructs are used and the aim is to apply correctness-preserving transformations to the specification to create the program code.

4. Static verification: The developed software is statically verified using rigorous software inspections. There is no unit or module testing process for code components.

5. Statistical testing of the system: The integrated software increment is tested statistically to determine its reliability. These statistical tests are based on an operational profile which is developed in parallel with the system specification.