Software Verification & Validation

Wednesday, December 9, 2020 19:09

1. **Verification and Validation**

Introduction

1) **Verification**: the software should confirm to its specification. We check whether we are developing the right product or not.
2) **Validation**: the software should do what the user really requires. We check whether the developed product is right.

It is also known as **V & V** process. It has two principal objectives:

- Discovery of defects in a system
- Assessment of whether or not the system is useful and usable in an operational situation.

Goals:

- Establish confidence that the software is fit for its purpose.
- It doesn't mean completely free of defects, rather, it must be good enough for its intended use and the type of use will determine the degree of confidence that is needed. The confidence depends on system's purpose, user expectations and marketing environment.

Types of V & V:

- Static Verification (Software Inspections): concerned with analysis of the static system representation to discover problems. The code is not executed. The product is evaluated by going through the code. Types of static verification:
  - Walkthrough (informal)
  - Inspection (formal than walkthrough)
  - Reviews (one or more persons check the changed documents, data or code to determine if the changes are correct.)
- Dynamic Verification (Software Testing): concerned with exercising and observing product behavior. Here, the system is executed with test data and its operational behavior is observed. Types:
  - Functional testing: It includes unit testing, integration testing and system testing.
  - Non-functional testing: It includes user acceptance 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.

Verification and validation planning

Planning should start early in the development process. The plan should identify the balance between static verification and testing. Test plan is about defining standards for the testing process.

Feasibility:

- Will current software and hardware technology meet the needs of user?
- Is it cost effective from business viewpoint, or is it within existing budgetary constraints?
- More detailed analysis: cheap and quick or expensive and slow.

Structure of a software test plan

- **The testing process**: A description of the major phases of the testing process.
- **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 for this 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 software tools required and estimated hardware utilization.
- **Constraints**: Constraints affecting the testing process such as staff shortages should be anticipated in this section.

Software inspection

- These involve people examining the source representation with the aim of discovering anomalies (unusual behaviors) and defects.
- Inspections does 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.).
- An effective technique for discovering program errors.

Inspections and testing

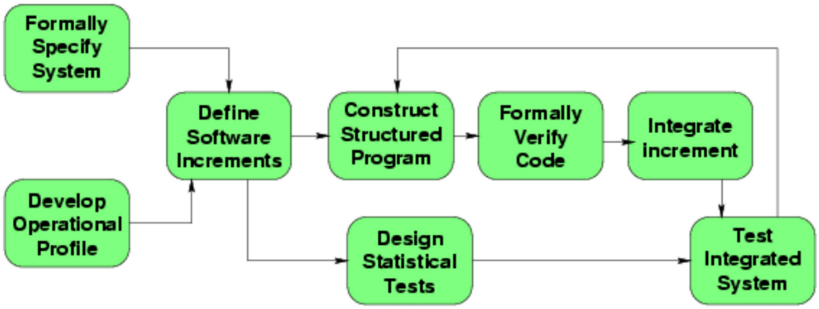
- Inspections and testing are complementary and not opposing verification techniques.
- Both should be used during the V & V process.
- Inspections can check conformance with a specification but not conformance with the customer's real requirements.
- Inspections cannot check non-functional characteristics such as performance, usability, etc.

Program Inspections

- Formalized approach to document reviews.
- Intended explicitly for defect detection (not correction).
- Defects may be logical errors, anomalies in the code that might indicate an erroneous condition (e.g. an uninitialized variable) or non-compliance with standards.

Cleanroom software development (process component) (1.4 Cleanroom software development (odu.edu)**)**

- The name is derived from the 'Cleanroom' process in semiconductor fabrication. The philosophy is defect avoidance rather than defect removal. The cleanroom process was originally developed by Harlan Mills and several of his colleagues including Alan Hevner at IBM.
- The cleanroom software engineering process is a software development process intended to produce software with a certifiable level of reliability. This kind of testing depends heavily on walkthroughs, inspection, and formal verification.
- This technique reportedly produces documentation and code that's extra reliable and fixable than various development methods relying heavily on code execution-based testing.
- Software development process is based on:
  - **Formal specification**: The computer code to be developed is formally given. A state-transition model that shows system responses to stimuli is employed to precise the specification.
  - **Incremental development**: The computer code is partitioned into increments which are developed and validated separately using clean room process.
  - **Structured Programming**: Only a restricted range of management and information abstraction constructs are used. The program development method is that the method of stepwise refinement of the specification.
  - **Static verification using correctness arguments**: The developed computer code is statically verified using rigorous computer code inspections. There's no unit or module testing method for code components.
  - **Statistical testing to determine program reliability**: The integrated computer code increment is tested statistically to work out its responsibility. Statistical testing focuses on how faulty programs can affect its operating conditions rather than finding defects. Here, software is tested with the test data that statistically models the working environment. (Software Engineering | Cleanroom Testing - GeeksforGeeks)
- **Cleanroom Process:**



- **Cleanroom process teams:**
  - **Specification team**: Responsible for developing and maintaining the system specification.
  - **Development team**: Responsible for developing and verifying the software. The software is not executed during this process.
  - **Certificate team**: Responsible for developing a set of statistical tests to exercise the software after development. Reliability growth models used to determine when reliability is acceptable.

2. **Software Testing**
   a. **Introduction:**
      - Testing is intended to show that a program does what it is intended to do and to discover program defects before it is put into use.
      - The testing process has two distinct goals:
        - To demonstrate to the developer and the customer that the software meets its requirements.
        - To discover situations in which the behavior of the software is incorrect, undesirable, or does not conform to its specification.
      - The first goal leads to validation testing, where you expect the system to perform correctly using a given set of test cases that reflect the system's expected use.
   b. **Types of Testing**
      i. Defect testing: It is the tests designed to discover system defects. A successful defect test is one which reveals the presence of defects in a system.
      ii. Validation testing: It is intended to show that the software meets its requirements. A successful test is one that shows that the requirement has been properly implemented.
   c. **Testing approaches**
      i. White box testing
      ii. Black box testing
   d. **Types**
      i. Unit Test
      ii. System Test
      iii. Integration Test

        iv. Validation
    e. Testing work benches
3. **Critical system Validation**
    a. Introduction
    b. Formal methods and critical systems
    c. Reliability validation
    d. Safety assurance
    e. Security assessment
4. **Software Cost Estimation**
    a. Introduction
    b. Productivity
    c. Estimation techniques
        i. Expert Judgement
        ii. COCOMO 2nd
        iii. Functional point
        iv. KLOC
5. **Software Re-engineering**
    a. Introduction
    b. Source code translation
    c. Reverse engineering