Software Engineering 2UCCE501

Module 5

Module 5 Testing & Maintenance

5.1 Testing Concepts: Purpose of Software Testing, Testing Principles, Goals of Testing, Testing aspects: Requirements, Test Scenarios, Test cases, Test scripts/procedures,

5.2 Strategies for Software Testing, Testing Activities: Planning Verification and Validation, Software Inspections, FTR

5.3 Levels of Testing : unit testing, integration testing, regression testing, product testing, acceptance testing and White-Box Testing

5.4 Black-Box Testing: Test case design criteria, Requirement based Testing, Boundary value analysis, Equivalence Class Partitioning

5.5 Object Oriented Testing: Review of OOA and OOD models, class testing, integration testing, validation testing

5.6 Reverse & Reengineering, types of maintenance

Strategies for Software Testing

- Testing is a **set of activities that can be planned in advance** and conducted systematically.

- A number of **software testing strategies provide** the **software developer** with a **template** for testing and have the following generic characteristics:

Strategies for Software Testing

1. Testing begins at the **component level** and works "**outward**" toward the **integration** of the entire computer-based **system**.

2. **Different testing techniques** are appropriate at **different points in time**.

3. Testing is **conducted by the developer** of the software and (**for large projects**) an **independent test group**.

4. **Testing and debugging are different activities**, but debugging must be accommodated in any testing strategy.

- A testing strategy must implement low level and high level tests

- A strategy must provide guidance for the practitioner and a set of milestones for the manager.

Strategic Issues

Best strategy will fail if a series of overriding issues are not addressed

- Following are the strategic issues to be considered:

1. ***Specify product requirements in a quantifiable manner long before testing commences.***
    - objective of testing is to find errors
    - a good testing strategy also assesses other quality characteristics as well.
    - Measurable requirements to be specified for unambiguous results

Strategic Issues

2. State testing objectives explicitly

- specific objectives of testing should be stated in measurable terms.
- For example, test effectiveness, test coverage, the cost to find and fix defects, frequency of occurrence, and test work-hours should be stated within the test plan.

3. *Understand the users of the software and develop a profile for each user category.*

- Use cases that describe the **interaction scenario for each class of user** can **reduce overall testing effort** by focusing testing on actual use of the product.

Strategic Issues

4. *Develop a testing plan that emphasizes "rapid cycle testing."*

- Is mindset and skill set to carry out testing more quickly , less expensive and best results.
- The feedback generated from these rapid cycle tests can be used to control quality levels and the corresponding test strategies.

5. Build "robust" software that is designed to test itself.

- Software should be capable of diagnosing certain classes of errors.
- The design should accommodate automated testing and regression testing.

Strategic Issues

6. Use effective technical reviews as a filter prior to testing

- Technical reviews can be as effective as testing in uncovering errors.
- Reviews can reduce the amount of testing effort that is required to produce high quality software.

7. Develop a continuous improvement approach for the testing process.

- The test strategy should be measured.
- The metrics collected during testing should be used as part of a statistical process control approach for software testing.

Strategic Issues

1. ***Specify product requirements in a quantifiable manner long before testing commences.***
2. ***State testing objectives explicitly***
3. ***Understand the users of the software and develop a profile for each user category.***
4. ***Develop a testing plan that emphasizes "rapid cycle testing."***
5. ***Build "robust" software that is designed to test itself.***
6. ***Use effective technical reviews as a filter prior to testing***
7. ***Develop a continuous improvement approach for the testing process.***

Verification and Validation

- **Verification** refers to the set of activities that ensure that software correctly implements a specific function(**algorithm**).

- **Validation** refers to a different set of activities that ensure that the software that has been built is traceable to **customer requirements**.

- Verification and validation encompasses a wide array of SQA activities.

Verification and Validation

- SQA includes following activities:
  - formal technical reviews
  - quality and configuration audits
  - performance monitoring
  - Simulation
  - feasibility study
  - documentation review
  - database review
  - algorithm analysis
  - development testing
  - qualification testing
  - installation testing

Testing defines the principles for quality assurance and error detection.

Formal Technical Reviews (FTR)

- Formal Technical Review (FTR) is a **software quality control activity** performed by software engineers (and others).

The objectives of an FTR are:

(1) to **uncover errors in function, logic, or implementation** for any representation of the software

(2) To **verify** that the **software** under review **meets its requirements**

(3) to **ensure** that the **software** has been **represented according to predefined standards**

(4) to **achieve software** that is developed in a **uniform manner**

(5) to make **projects more manageable**.

Formal Technical Reviews (FTR)

- the FTR serves as a training ground, enabling junior engineers to observe different approaches to software analysis, design, and implementation.

- The FTR is actually a class of reviews that includes ***walkthroughs* and *inspections.***

- **FTR is conducted as a meeting** and will be successful only if it is properly planned, controlled, and attended.

Formal Technical Reviews (FTR)

- **The Review Meeting**
  - Between **three and five people** (typically) should be involved in the review.
  - **Advance preparation** should occur but should require no more than two hours of work for each person.
  - The **duration** of the review meeting should be **less than two hours**.
  - The review meeting is attended by the review leader, all reviewers, and the producer.
  - One of the reviewers takes on the role of a *recorder*
  - The producer proceeds to “walk through”

Formal Technical Reviews (FTR)

- At the end of the review, all attendees of the FTR must decide whether to:
  (1) Accept the product without further modification
  (2) reject the product due to severe errors
  (3) accept the product with minor revisions

Formal Technical Reviews (FTR)

- **Review Reporting and Record Keeping**
  - During the FTR, a reviewer (the recorder) records all issues that have been raised.
  - review issues list is produced
  - What was reviewed?
  - Who reviewed it?
  - What were the findings and conclusions?

Formal Technical Reviews (FTR)

- **Review Guidelines**
- The following represents a minimum set of guidelines for formal technical reviews:

1. **Review the product, not the producer.**
2. **Set an agenda and maintain it.**

   An FTR must be kept on track and on schedule.

3. **Limit debate and rebuttal.**

   When an issue is raised by a reviewer, there may not be universal agreement on its impact.

4. **Enunciate problem areas, but don't attempt to solve every problem noted.**

   A review is not a problem-solving session.

5. **Take written notes.**

   It is sometimes a good idea for the recorder to make notes on a wall board, so that wording and priorities can be assessed by other reviewers as information is recorded.

Formal Technical Reviews (FTR)

6. Limit the number of participants and insist upon advance preparation

Keep the number of people involved to the necessary minimum.

7. Develop a checklist for each product that is likely to be reviewed.

A checklist helps the review leader to structure the FTR meeting and helps each reviewer to focus on important issues.

8. Allocate resources and schedule time for FTRs.

For reviews to be effective, they should be scheduled as tasks during the software process.

9. Conduct meaningful training for all reviewers