Software Testing (UE18CS400SB)
Unit 1

Aronya Baksy

August 2021

1 Introduction

- **Testing**: A process consisting of all life-cycle activities both static and dynamic concerned with **planning**, **preparation** and **evaluation** of a software product and the related work to determine that it **satisfies the specific requirements** and it fits the purpose, as well as to **detect defects**
- Provides stakeholders with info about quality of the software product under test
- Software testing ensures quality of software and accelerates software development
- Testing levels: Unit test, integration test, system test and acceptance test

2 Software Quality

- Quality is conformance to standards or requirements (PMBOK)
- **ISO 9000**: totality of features and characteristics of a product/service that bear on its ability to satisfied stated and implied needs
- **IEEE definition** of quality:
  - The degree to which a system/component/process meets requirements
  - The degree to which a system/component/process meets user needs or expectations
- **Pressman's definition** of quality: Conformance to explicitly stated functional and performance requirements, explicitly documented development standards and implicit characteristics that are expected of all professionally developed software
- Quality is a process than a product. A continuous process of improvement and use of lessons learnt in the past to enhance processes
- Software defects are costly (money/reputation/danger to human life)
- Reasons for software failures:
  - Uniqueness of the product
  - High complexity
  - Limited opportunities to detect bugs
  - Software dev environment
  - Teamwork

2.1 Software Quality Factors

- **Correctness**: With regards to conforming to requirements
- **Reliability**
- **Efficiency**
- **Usability**
- **Maintainability**
- **Flexibility** with regard to changing requirements and adding functionality

- **Testability**
- **Portability** across platforms and environments (cloud or native)
- **Interoperability** with other services and products
- **Reusability**

2.2 Deming's PDCA Cycle

- **Plan** improvements to existing practices (establish objectives and processes needed to deliver them)
- **Do**: Implementation of the plan and objectives of the previous step
- **Check** to see if desired results are achieved (compare with expected outcomes to measure similarity and differences)
- **Act**: Implement corrective solutions, process improvement appens in this phase

2.3 Cost of Quality

- Total price of all effort needed to achieve product/service quality (3-5% of total program cost devoted to cost of quality)
- The quality cost has 4 main components:
  - **Prevention Cost**
  - **Appraisal Cost**
  - **Internal Failure Cost**
  - **External Failure Cost**

2.4 Project Quality Management

- The processes required to ensure that the project satisfies the needs for which it was undertaken
- Includes all activities of which the overall management function that determine the quality policy, objectives and responsiblities and implements them using quality planning, assurance, control and improvement within the quality system.
- PQM addresses both management and end product of the project

2.4.1 PQM Processes

- **Quality Planning**
- **Quality Assurance**
- **Quality Control**
- **Quality Improvement**

Quality Assurance	**Quality Control**
Process-oriented	Product-oriented
Focus on preventing quality issues	Focus on identifying quality issues in products
Performed throughout the cycle	Performed after product is built
A staff function	Line function
e.g.: Reviews, Audits	e.g.: Software testing at various levels

2.4.2 Quality Control using Fish-Bone Analysis

- A multi-purpose representation used for brainstorming, but is mostly used for cause-effect analysis
- The fish head represents the effect that is observed
- The bones on the fish's spine represent the causes of the effect that was observed
- Used in Agile-based development life cycles like Scrum and Kanban
- Steps in building a Fish-bone diagram:
  1. Agree on the problem statement (**effect**). This forms the fish head. The problem statement must be clear, specific but not defined in terms of any solutions.
  2. Agree on major categories of **causes**. These are the branches on the fish spine
  3. Brainstorm all possible causes of the problem. As each idea is put forth, the facilitator puts it on the appropriate category (branch) of the diagram
  4. Ask "why does it happen" and form sub branches from the main branches
  5. Generate deeper levels of causes and organize them under relevant categories of causes.

3 Software Developement Life Cycle

- Generic phases of SDLC: Requirement analysis, planning, design, implementation, testing, deployment and maintenance
- **Verification**: Evaluating whether the product of a particular phase satisfies the conditions imposed at the start of that phase
- **Validation**: Evaluating a system/component during or at the end of the development process to determine whether it satisfies the specified requirements.
- Every phase of SDLC is characterized by: entry, task, verification, and exit activities.
- Attributes of an SDLC model:
  - Activities performed the sequence and their deliverables
  - Methods of validation and deliverables
  - Methods of verification for each activity, including communication mechanisms between activities

3.1 Waterfall Model

- Progress through each phase flows from top to bottom, in a cascading manner.
- **Strengths**: simple, useful for projects divided into components
- **Drawbacks**: Delay in feedback among phases, errors in one phase are not detected till next one

3.2 Prototyping Models

- Early and frequent feedback increases chances of meeting customer requirements, and allows project to adapt to rapid changes easily
- **Prototyping**: constant user interaction produces a prototype from which the SRS is derived. Prototype is discarded after this, and then the development proceeds once client accepts the SRS

3.3 Iterative Model

- Start with a skeleton code and iteratively refine it till a complete product is ready
- It is represented in the form of a spiral and hence called a **spiral model** as well

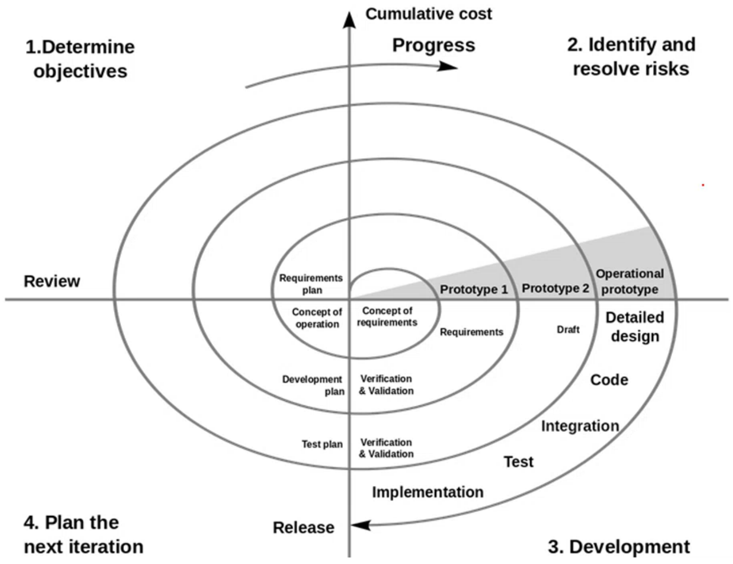

Figure 1: Spiral Model

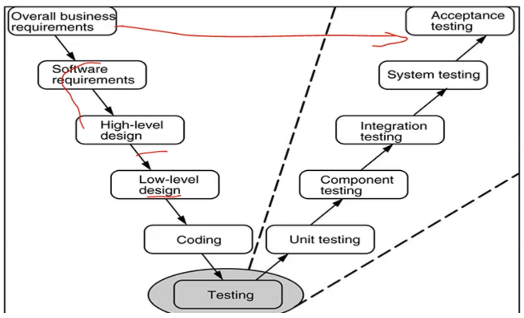

Figure 2: V Model

3.4 V Model

- The test planning and the development activity is done at the same time
- The development and testing activities form the 2 lines of the V, hence the shape and the name of the model.
- Test planning is done early, but tests are executed only after implementation
- Apart from the shape, V also stands for **Verification** and **Validation**.
- In a V model, every activity is sequential (one phase only starts after the previous one finishes)

3.5 Modified V Model

- Modified V model recognizes that different parts of a product are in different stages of evolution
- Each part entires the appropriate testing phase when the appropriate criteria are met

4 Software Test Life Cycle

- A sequence of specific ccations performed during the testing process to ensure that softwre quality objectives are met
- Includes both verification and validation
- The phases of a normal STLC are:
  - Contract Signing
  - Requirement Analysis
  - Test planning

  - Test development
  - Test environment setup
  - Test execution
  - Retest defects
  - Closure

4.1 STLC Phases

4.1.1 Requirement Analysis

- **Entry Criteria**: SRS and architecture design available, test acceptance criteria defined
- **Activity**:
  1. Identify and prioritize tests to perform
  2. Prepare an RTM
  3. Identify the test environment and perform automation feasibility analysis
- **Exit Criteria**: Signed off RTM and authorized automation feasibility report
- **Deliverables**: RTM, automation feasiblity report

4.1.2 Test Planning

- **Entry Criteria**: SRS, RTM, automation feasibility report
- **Activity**:
  1. Prepare test plan/strategy document
  2. Estimate test effort
  3. Resource planning
  4. Training requirements
- **Exit Criteria**: Approved test plan document, signed off effort estimate
- **Deliverables**: Test plan document, effort estimate document

4.1.3 Test Case Development

- **Entry Criteria**:SRS, RTM, test plan, automation analysis report
- **Activity**:
  1. Create test cases
  2. Automation scripts if applicable
  3. Review and baseline test cases/scripts
  4. Create test dataset
- **Exit Criteria**: Reviewed and signed test cases/scripts, test dataset
- **Deliverables**: Test cases, scripts, test dataset

4.1.4 Test Environment Setup

- **Entry Criteria**: System design and architecture, environment setup plan
- **Activity**:
  1. Understand the required setup
  2. Prepare hardware & software requirements and envt. checklist
  3. Perform smoke test on build and accept based on smoke test result
- **Exit Criteria**: Working test envt as per plan, complete test data setup and successful smoke test
- **Deliverables**: Ready test envt and test dataset, smoke test result

4.1.5 Test Execution

- **Entry Criteria**: Baselined RTM, test plan, test scripts/cases/envt/data, unit/integration test report
- **Activity**:
  1. Run tests, document results and log defects
  2. Update test assets
  3. Map defects to test cases in RTM
  4. Perform regression test
- **Exit Criteria**: All planned tests are executed, defects logged and tracked
- **Deliverables**: Filled RTM with exec status, updated test cases and results, bug reports

4.1.6 Test Cycle Closure

- **Entry Criteria**: Completed testing with results, defect logs available
- **Activity**:
  1. Evaluate completion critera (time, cost, coverage, biz objectives)
  2. Test metric calcuation
  3. Document learnings from project
  4. Prepare closure report (report of quality of work product to customer)
  5. Test result analysis to find distr. of bugs by type and severity
- **Exit Criteria**: Signed off test closure report
- **Deliverables**: Test closure report, test metrics

5 Classification of Testing Types

- **Based on method**:
  1. White-box testing
  2. Black-box testing
- **Based on requirement type**:
  1. Functional testing: can be black or white box
  2. Non-functional testing: load/stress test, usability test, l10n testing
- **Based on life-cycle phase**: Focus changes from unit to sub-system to entire system
  1. Unit testing: uses white-box testing
  2. Integration testing: Using white/black box tests
  3. System testing: uses black-box
- **Based on need**:
  1. Regression testing: ensure that changes have not impacted existing behaviour
  2. α testing (limited user base, dev site) and β testing (larger user base, cust. site)
  3. Acceptance testing