UNIT – 4

Software testing scenarios
- More than required testing => Waste of time and money
- Less than required testing => High cost of support
- No testing => Impossible to support

Problems with Traditional testing (too little and too late)
- there could be many faulty requirement specifications and faulty software design
- Challenging and Costly, Time Taking, Infeasible

Quality Standard Documents
- Verification => Static Testing
  - checking requirement specifications, design, source code etc.,
  - Do not run Source Code (dead code, unused code, faulty logic etc.,)
- Validation => Dynamic Testing
  - Unit - testing source code by small modules
  - Integration - tested when integrated with other application
  - System -
  - User acceptance - testing by end user

  - Actual running of Source Code

Testing Strategy and Planning
- Planning
  - work breakdown structure, requirement review, resource allocation, effort estimation, tools selection, setting up communication channels, etc.,

- Test Prioritization
  - Focus on Features which are used most by user
- Risk Management
  - causes are unrealistic schedule, resource unavailability, skill unavailability, frequent requirement changes, etc.,
  - overconfidence of employees (test manager, HR, marketing team)
  - delay of resources (human and material)
- Effort Estimation
  - scheduling, resource planning and budget for a test project
  - project size, productivity, and test strategy
  - wideband Delphi technique, experience-based estimation

  - Test Point Analysis
    - Product size (no. of function points)
    - Test strategy (Quality level + Priority areas)
    - productivity (Experience + Skills)

Test Project Monitoring and Control

- Test Case Design
  - what kind, how many per modules and priority modules

  - Test Types
    - functionality, performance, usability, compatibility
    - Regression tests for applications having multiple versions
    - Verification and Validation
- Test Case Writing/Management
- Test Script Creation/Bed Preparation
  - installing the application on a machine that is accessible to all test teams
  - "Application under test", testing application in which it is meant for (example APP in android)
  - test data preparation is very tricky
- Test Case Execution
- Defect Tracking (Testers should stay until application is deployed, if Testers are on contract schedule should be taken care)
  - Defect Logging
  - Assign Defect
  - Fix Defect
  - Defect Verification
  - Defect Closure
- Test Case Closure

Test Bed Preparation
Assume it as... Hosting on GitHub... Giving access to a particular set of people or to a team and they will test it
I might get an error... If I run again, I might not get the same error...
To avoid these... They completely isolate the testing file with other files so that it works the same every time (whether it be an error or correct execution)

UNIT - 5

Product Release

Product Release Management Tasks

- Estimate cost of providing support
- Selection of software version to be shipped
- Decision for alpha, beta or regular release
- Create walk around for known defects
- Provide training to support staff
- Make customer support strategy

Product Release Management

- pressure to launch new versions, new features
- porting to new platform
- half-baked product

Product Release Types

- Alpha release
- Beta release
- Internal release
- Normal release

Production implementation tasks (product run smoothly, problems due to unforeseen circumstances, recommended to prepare a list of developer requirements)

- Check software interfaces
- Check hardware interfaces
- Create master data
- Create test data
- Create user accounts
- Check infrastructure for installation

User Training

- User manual (or) Tutorials up to date and sync with present version
- Not possible to train all users
- Very important step

Maintenance (more than 70% of all costs associated with software product development, implementation, and support and maintenance)

- Technology obsolescence
- Software defects
- Change in user requirements

Reasons for software maintenance

- Software defects
- New user requirement
- Changed user requirement
- Technology obsolescence
- Better technology

Software Maintenance Types

- Corrective
    - User finds a bug while using => reports => maintenance team plans and fixes => user uses
- Preventative
    - change in business/operative or hardware/software environment => affect software operation => maintenance to reuse product
- Perfective
    - change in business environment => additional/modified functionality needed
- Adaptive
    - change in software or hardware interface => adaptive maintenance to reuse software

Maintenance Cost

- Revenue Loss
- Opportunity Loss
- Productivity Loss

Maintenance Process

- Quick fix Model
    - Immediately fixed without any planning
- Boehm’s model
    - Boehm’s model is based on economic models and often involves calculating ROI, for any planned maintenance. If ROI turns out to be good, then it is carried out or else it is dropped
- Osborne’s model
    - Change requests
    - Quality Assurance
    - Metrics
    - Reviews
- Iterative enhancement model
    - Similar to iterative software development
    - High priority fixed first, low priority fixed next
- Reuse oriented model
    - Component-based software products
    - Existing components are analysed and changes are made

Maintenance Life Cycle (crucial part, lot of time and effort)

- List of defects
- Subset of defects
- Defect fixing planning/execution
- Patch application
- Test application
- Maintenance complete

Maintenance techniques

- Reengineering (reuse technology)
    - each defect is specifically analyzed to find out the root cause of the defect
- Forward engineering (opposite of reverse engineering)
    - we have ample documentation about the existing product
    - existing product needs to be extended so that the new needs can be fulfilled
- Reverse engineering
    - when nonexistent or sketchy documentation is available for the software product

Software release Case Study