ISTQB foundation level

SDLC -> Software Development Life Cycle Models like Waterfall and Agile

SRS -> Software Requirements Specification

HLD -> High Level Design

Planning for testing is the most important part of testing.

Difference of Vocabularies:

Quality Assurance: Adherence to process, it is `proactive and prevents`, are we doing the right thing

Quality Control: Testing, test design, test execution. `Reactive and detects`. So Testing lives IN the quality Control.

Main umbrella is Quality Management, under which is QA, under which is QC, under which is Testing.

Error, Defect, and Failure are not the same thing.

Error is a mistake. lets say customer and software team are not on the same page on understanding stuff.

Defect is Bug. It is a mistake `in the software.`

A bug(defect) may or may not lead to the failure of a software product.

Failure is simply an observable defect.

These should be caught in testing.

Test Process Fundamentals.

- Test Planning

What test techniques do you use and how? Schedule and deadlines?

- Test Monitoring and Control

What was planned vs what is actually happening? Are we on track on plan and deadlines?

- Test Analysis

Business requirement, functional and non-funcional requrements, designs, documenation,code etc.

All of these are test basis upon which the tests will be written.

- Test Design and Implementation

All so far was to analyze what to test. Now it is time to how to test.

This process entails designing and prioritizing the tests cases, identifying test data.

Design is how to test, implementation is `Do we have eveything we need to test?`

Implemenattion entails creating procedures and automated test scripts, setting up environment,preparing test data.

- Test Execution and Completion

SUT -> System Under Test.

In this step you execute the designed and implemented tests and compare the results with the expected results.

Test completion means creating a test report, also ensureing that all defetc report are documented.

SDLC Models

Regardless which model you choose, there is a common ground of stages:

- Requirement analysis
- Design
- Development
- Testing
- Deployment
- Maintenance

There are 2 Main Models:

1. **Sequential**

- V-Model:

Unlike the waterfall model, the V-Model integrates the stages with testing. Main idea in V-Model is `Early testing`

Each development phase is linked with a testing phase like

low-level design ==> Unit testing.

High-level design ==> Integration testing.

System-design-and-development ==> System testing.

Analysis ==> acceptance testing.

Unlike Waterfall, which is a linear process, the V-Model is more synchronized approach. Testing starts earlier than waterfall.

- Waterfall

simple and in-bulk deployment but not flexible and could be slow. one blockage could cause next steps to be blocked.

Also any phase can become a bottleneck and very late feedback threat.

It adds additional stress to testers.

Each of these steps given above are done sequentially for each big feature requested.

First business analyst makes analysis, then it is designed, then devs develop it and it is handed to tester as version 1.0,

then testers test, if bugs, send back to devs, they fix them, deliver the next version (1.1) and so on then delivered to customer.

then as the situatuion requires, minor updates and maintenance is undertaken.

2. **Iterative(incremental)**

instead of working in bulk, you work in small increments and in each step, you test and fix the bugs. so instead of 10 features

at a time, you work on one or two features at a time.

In this model, work is divided into features or by `fixed time cycles`

Instead of making one major activity at a time, you do everything in small increments in fixed time cycles.

- RUP:
-

Rational Unified Process. bigger deliveries

longer iterations.

- Scrum:

iteartions are shorter, but more frequent. smaller deliveries Part of AGILE

- Kanban:
- Part of AGILE fluid roles, tasks are shared by everyone, no scrum-master, timelines evolve.
- Spiral:

experimental increments, most flexible model.

Test Levels

- **Component (unit - module) Testing**

Component is the smallest unit of testing which can be tested in isolation.

- **Integration Testing**

We had tested units (components) individually before, now we connect them and test them together.

Each time, we add another unit and test again.

indivudually working units may go crazy when connected together. Integration tests this.

Here we tests `component integration` which is testing components together and `system integration` means whether this integration is

compatible with other, external components.

Here you test:

- subsystems
- databases
- microservices

- **System Testing**

Once the all units are tested and also integration test is completed, now this interconnected units compose a system.

This level is testing the whole system.

This is like full integration testing of all modules. You dont test different combinations of the components but test the system as a whole.

This is where End2End testng or Front2Back testing comes in.

2 POINT:

- tester does not test from WITHIN the system but from OUTSIDE the system using interfaces or endpoints.
- considers system's paths and flows. So tests a path of actions in the system.

- **Acceptance Testing**

similar to system testing, but we test the system from the end-user perspective.

Acceptance testing has `Alpha` and `Beta` levels.

both carried out by independent tester or potential customers.

Alpha happens at the developer's site whereas beta testing happens at the customer's site.

Test Types !!

A test type is a group of test activities aimed at testing a specific aspect of a system.

Functional Testing

Testing the functionality of a system. Does it work as expected? function = work.

Unit testing, integration testing, system testing, UI testing are all types of **functional** testing.

System and UI testing compose acceptance testing so functional testing also covers acceptance testing.

What are functional coverage and coverage gap?

Coverage ==> how much of the system is covered by tests.

Coverage gap ==> how much of the system is not covered by tests.

Non-Functional Testing

It is not about the functionality but regarding User experience.

Like UI, security, speed performance, usability. You ask `how` questions like how fast, how secure, how stable.

all the functions could be working, so passing functional test, but it could be slow, crashing, unstable, etc.

Non-functional testing includes:

- usability (can be fixed in alpha beta testing)
- performance (needs stress testing, load testing, endurance testing)
- security (needs pentesting, ethical hacking, white-hat hacking.)

BlackBox- Whitebox Testing

This is another way of grouping the test types. a counterpart to functional testing - non-functional testing.

Blackbox: you dont look into what is inside the system, you look at the system as a whole. You rely on documentation and requirements.

send input, verify output.

Whitebox: knowing what is happenign inside the box and how it works. this is mostly done by devs during unittests.

Static Testing

Static Testing

Unlinke dynamic testing, in static testing, the code is never executed. it is used *early* in the SDLC process to ID issues that may cause problems in the later dev stages.

Static analysis is a tool-dtiven evaluation of any work product with structure (like code, documentation, etc.) it is especially important for **safety-critical systems**.

Static analysis is often incorporated into the CI/CD process.

Mesela bir belgeyi, specificationu, documentationu alip, onun icinde mantik analizleri yapiyoruz. Hukuk metni okur gibi.

Static Testing vs Dynamic Testing

They can have the same objectives like assessing quality of design, product, security, performance, etc; identify defects. But static and dynamic testing should be regarded as complementary to each other as they aim to find defects as soon as possible.

Dynamic testing is hard. requires lots of overhead like designing, implementing, executing, reporting the test results.

Static testing: defects are found in the work products, no code execution, can identify hhidden defects.

Reviews are manual static testing , reading manual and code documentation and requirements specifications.

There are 4 types of static testing:

- **Informal reviews**

reviewing without a formal process and documenting outputs. like 2 devs helping each other.

- **Formal reviews**

requires team effort, usually technical pairs of the author and other bodies meet and discuss for defects and document. no author lead.

- **Software inspection**

demands detailed examination of the work product using clearly defined goals.

- **Walkthroughs**

lead by the work product, aim to find defects and alternative implementations. Biri elinde kagit tek tek uzerinden geciriyor takimi.

==========