COURSE AGENDA

- **Testing in Software Development**
- **Testing in SDLC**

TURKCELL

TURKCELL AKADEMİ

YILDIZ TEKNİK ÜNİVERSİTESİ 1911

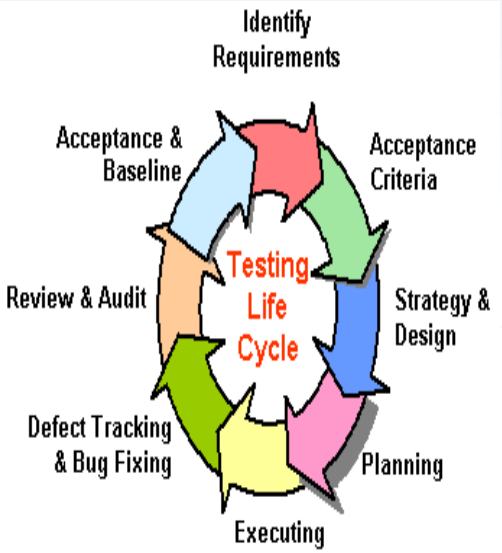
TURKCELL AKADEMİ

Testing in Software Development

TESTING IN SOFTWARE DEVELOPMENT

- SOFTWARE DEVELOPMENT MODELS
- SOURCE OF SOFTWARE DEFECTS
- TEST LEVELS

TESTING IN SOFTWARE DEVELOPMENT



- Testing needs to begin in parallel with the requirement analysis phase of the project and as early as possible in the life cycle.
- Testing activities should be carried out in parallel with development activities.
- Testing is a continuous activity that continued throughout the software development process.
- There is a testing process that correspons to a each software development process.
- Coding and testing should be done together for an effective testing.
- Testers should be involved in reviewing documents as soon as drafts are available in the development cycle
- There are different models for software development.

TURKCELL

TURKCELL AKADEMİ

TESTING IN SOFTWARE DEVELOPMENT

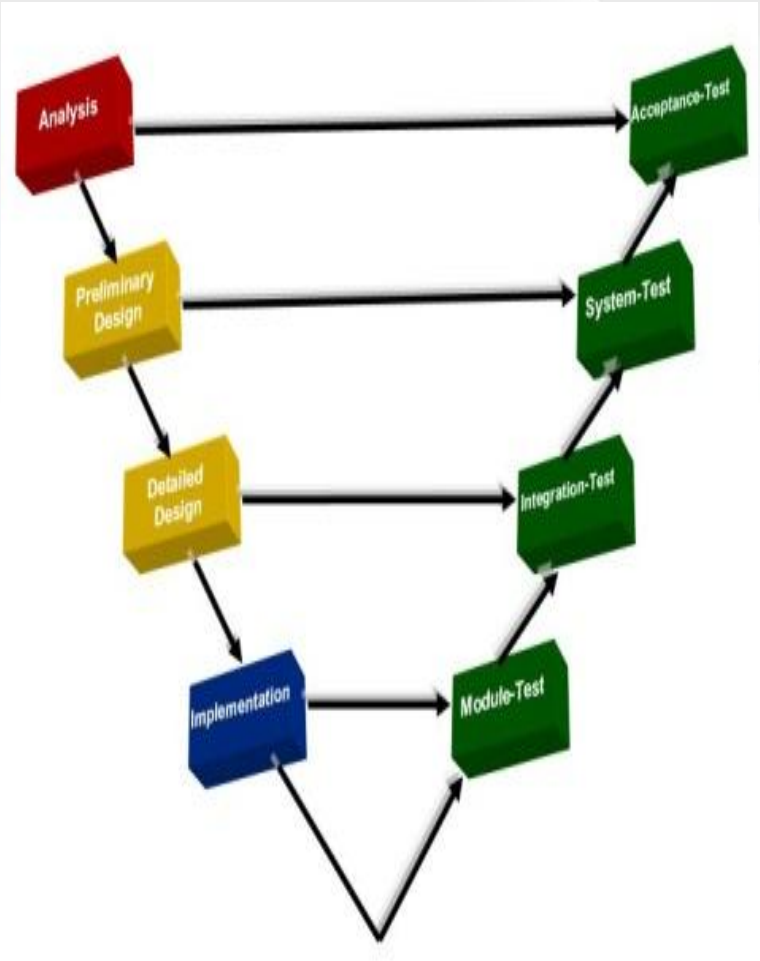
- SOFTWARE DEVELOPMENT MODELS
- SOURCE OF SOFTWARE DEFECTS
- TEST LEVELS

SOFTWARE DEVELOPMENT MODELS

V-Model

- V-Model is one of the most commonly used software development models.
- In practice, a V-model may have more, fewer or different levels of development and testing, depending on the project and the software product.
- V-Model is in the Sequential Development Model (Requirements-Driven Model) group. There are also traditional Waterfall models in this way.

SOFTWARE DEVELOPMENT MODELS



V-Model

V-Model has two basic concepts :

- **Verification:** Verification is concerned with evaluating a work product, component or system to determine whether it meets the requirements set. In fact, verification focuses on the question 'Is the deliverable built according to the specification?'.
- **Validation:** Validation is concerned with evaluating a work product, component or system to determine whether it meets the user needs and requirements. Validation focuses on the question 'Is the deliverable fit for purpose, e.g. does it provide a solution to the problem?‘.

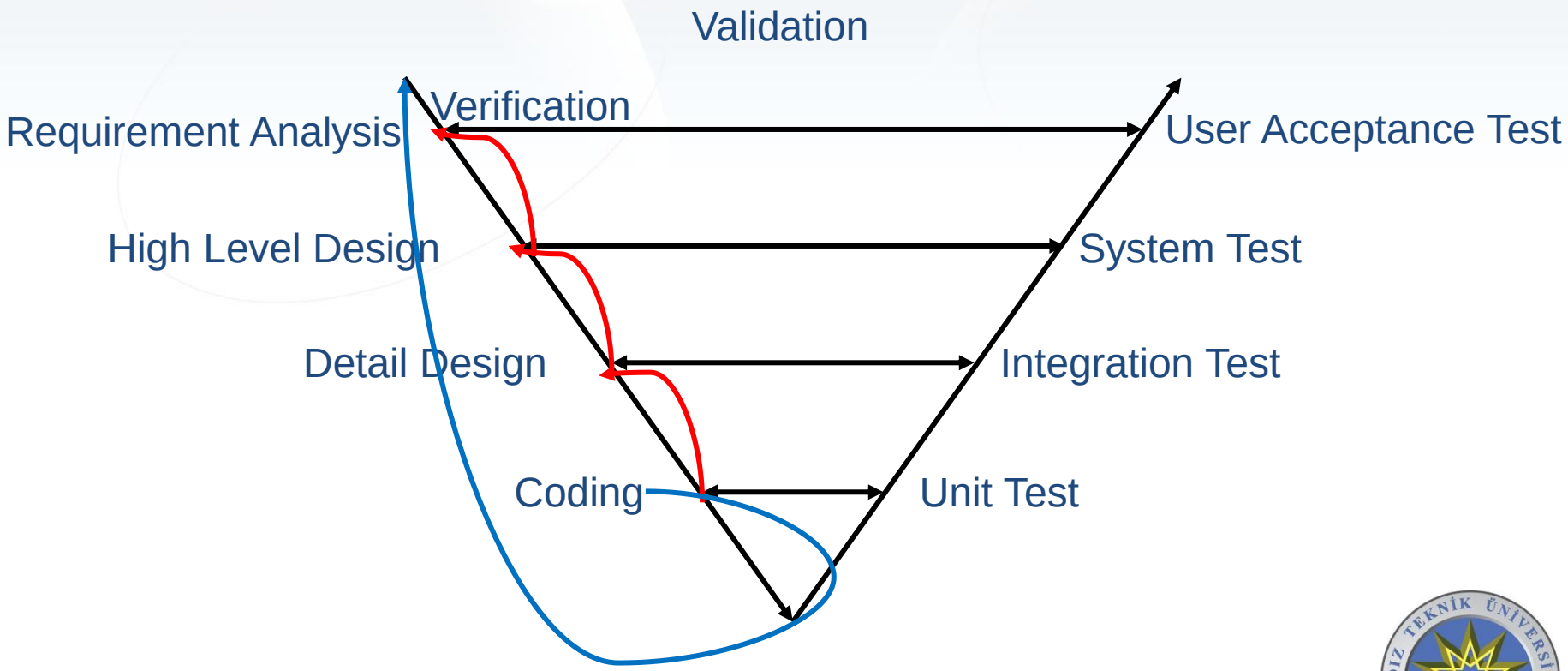
TURKCELL

TURKCELL AKADEMİ

YILDIZ TEKNİK ÜNİVERSİTESİ 1911

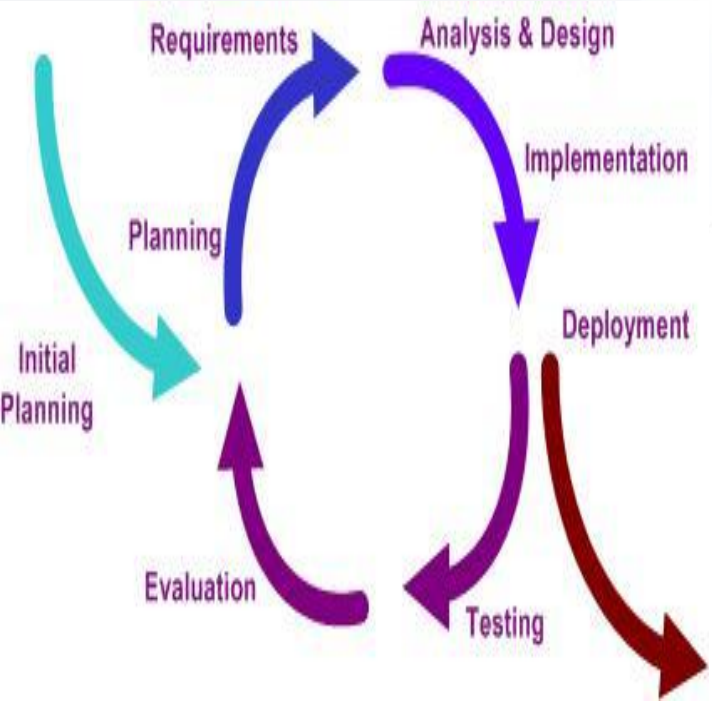
SOFTWARE DEVELOPMENT MODELS

V-Model



SOFTWARE DEVELOPMENT MODELS

Iterative Development Model



- Instead of one large development time line from beginning to end, we cycle through a number of smaller self-contained life cycle phases for the same project.
- Examples of iterative or incremental development models are prototyping, Rapid Application Development (RAD), Rational Unified Process (RUP) and agile development.
- A common feature of iterative approaches is that the delivery is divided into increments or builds with each increment adding new functionality. The increment produced by an iteration may be tested at several levels as part of its development. Regression testing is increasingly important on all iterations after the first one.

SOFTWARE DEVELOPMENT MODELS

Iterative Development Model

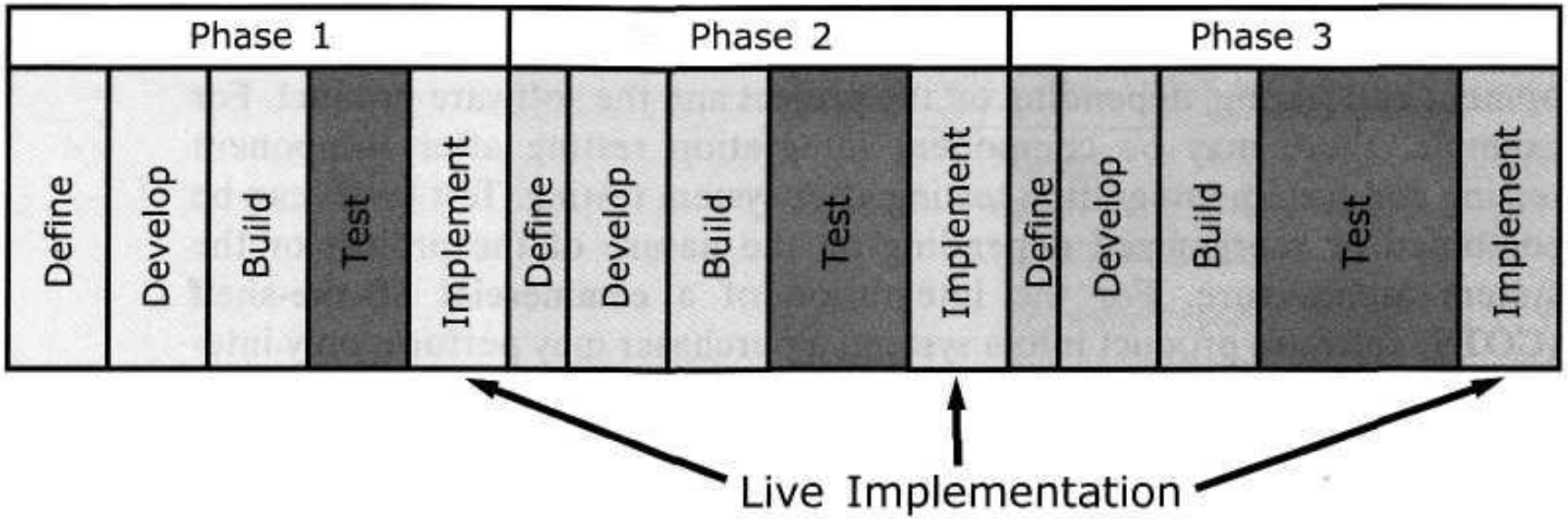


FIGURE 2.3 Iterative development model

SOFTWARE DEVELOPMENT MODELS

Rapid Application Development (RAD)

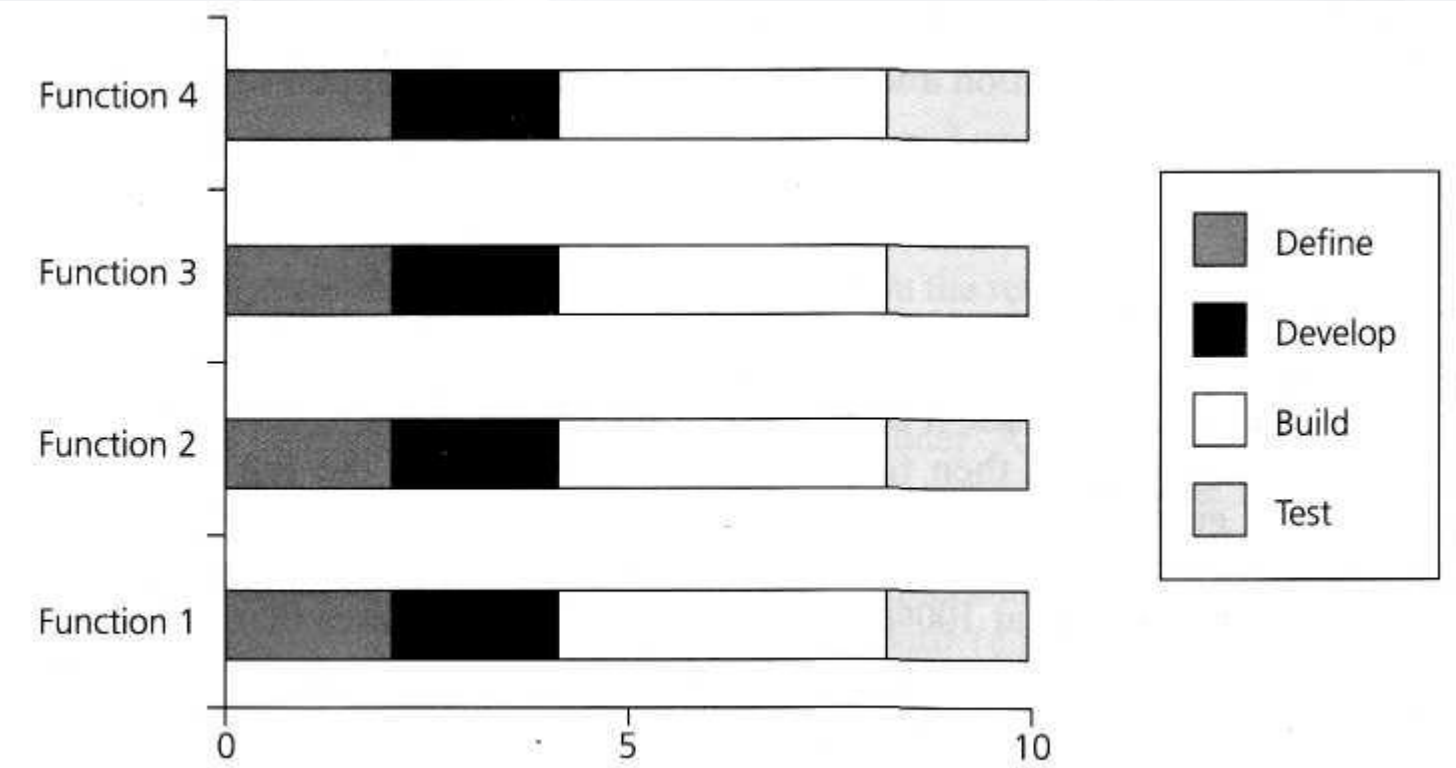


FIGURE 2.4 RAD model

SOFTWARE DEVELOPMENT MODELS

Agile Development Model

- Extreme Programming (XP) is currently one of the most well-known agile development life cycle models. The methodology claims to be more human friendly than traditional development methods.
- It promotes the generation of business stories to define the functionality.
- It demands an on-site customer for continual feedback and to define and carry out functional acceptance testing.

SOFTWARE DEVELOPMENT MODELS

Agile Development Model

- It promotes pair programming and shared code ownership amongst the developers.
- It states that component test scripts shall be written before the code is written and that those tests should be automated.
- It states that integration and testing of the code shall happen several times a day.
- It states that we always implement the simplest solution to meet today's problems.

TESTING IN SOFTWARE DEVELOPMENT

- SOFTWARE DEVELOPMENT MODELS
- SOURCE OF SOFTWARE DEFECTS
- TEST LEVELS

SOURCE OF SOFTWARE DEFECTS

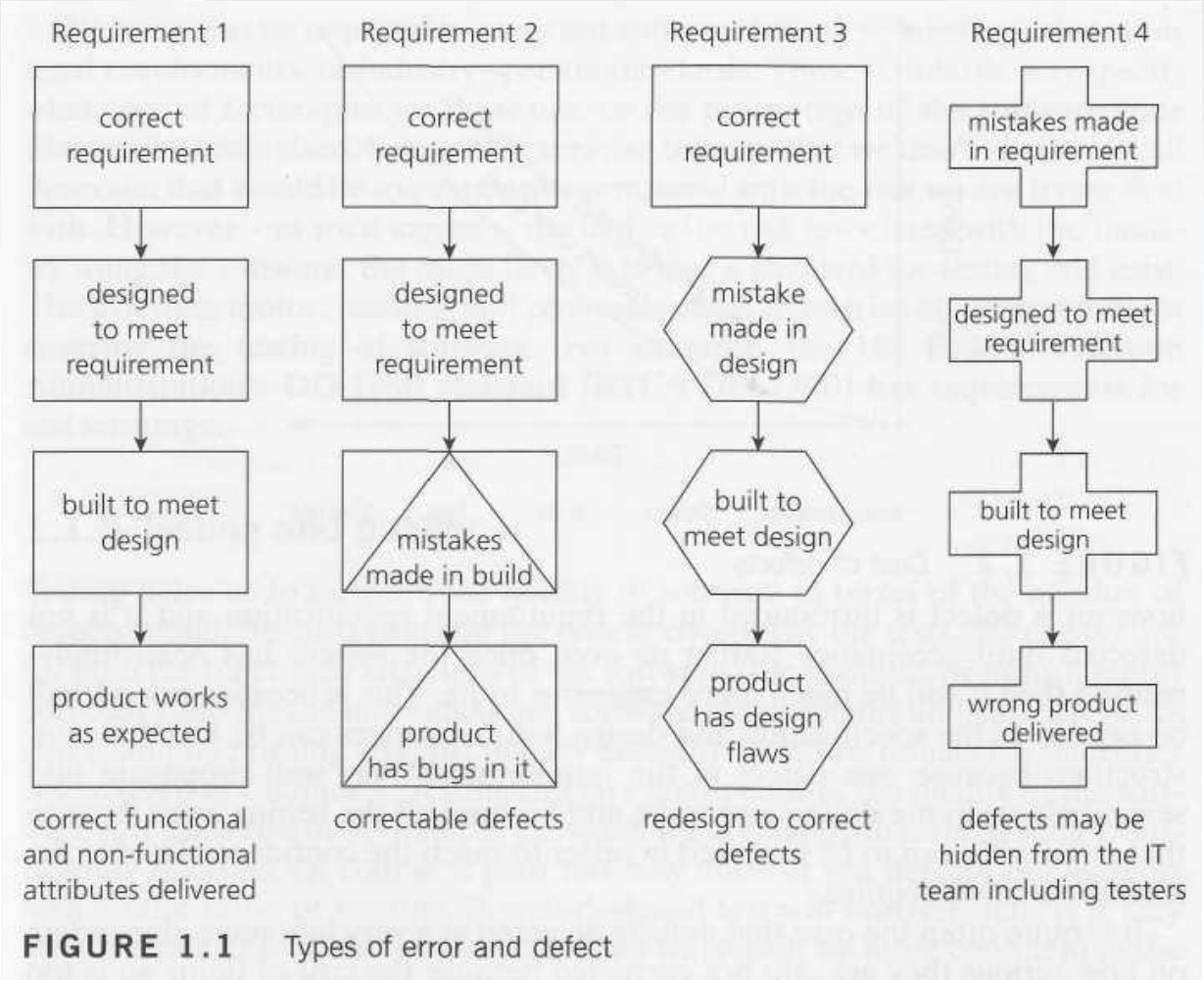


FIGURE 1.1 Types of error and defect

SOURCE OF SOFTWARE DEFECTS

When we think about what might go wrong we have to consider defects and failures arising from:

- Errors in the specification, design and implementation of the software and system
- Errors in use of the system
- Environmental conditions
- Intentional damage
- Potential consequences of earlier errors, intentional damage, defects and failures

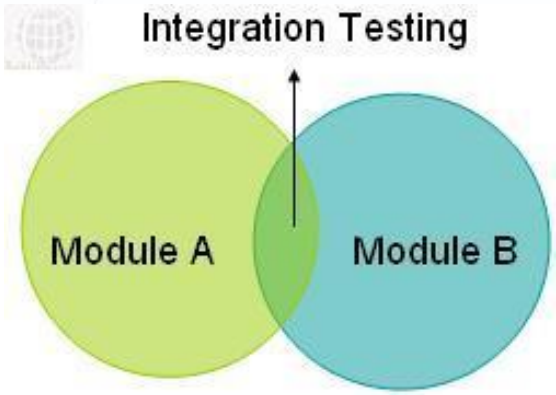
TESTING IN SOFTWARE DEVELOPMENT

- ✓ SOFTWARE DEVELOPMENT MODELS
- ✓ SOURCE OF SOFTWARE DEFECTS
- → TEST LEVELS

Component / Unit Testing

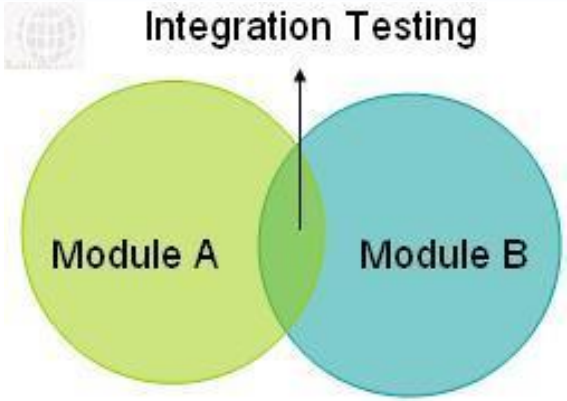
- Component testing, also known as unit, module and program testing, searches for defects in, and verifies the functioning of software (e.g. modules, programs, objects, classes, etc.) that are separately testable.
- Typically, component testing occurs with access to the code being tested and with the support of the development environment and in practice usually involves the programmer who wrote the code.
- Defects are typically fixed as soon as they are found, without formally recording the incidents found.
- One approach in component testing is to prepare and automate test cases before coding.

Integration Testing



- Tests interfaces between components, interactions to different parts of a system such as an operating system, file system and hardware or interfaces between systems.
- Integration testing tests the interactions and the functional relationships between software components and is done after component testing.
- The best choice is to start integration with those interfaces that are expected to cause most problems. Doing so prevents major defects at the end of the integration test stage.

Integration Testing

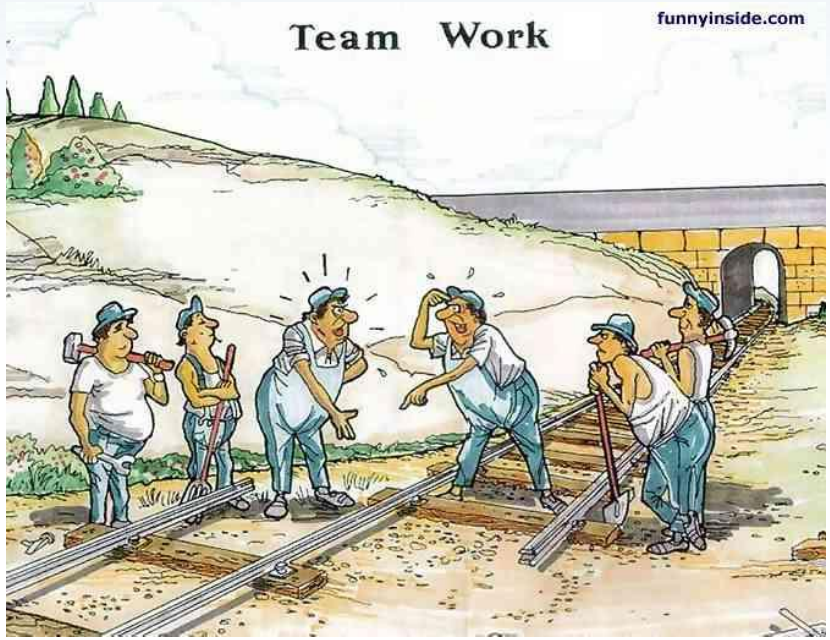


- In order to reduce the risk of late defect discovery, integration should normally be incremental.
- Ideally testers should understand the architecture and influence integration planning.
- If integration tests are planned before components or systems are built, they can be developed in the order required for most efficient testing.
- The greater the scope of integration, the more difficult it becomes to isolate failures to a specific interface, which may lead to an increased risk.

Integration Testing

Basic Integration Strategies:

- Top-Down
- Bottom-Up
- Functional Incremental
- Big-Bang



TEST LEVELS

System Testing

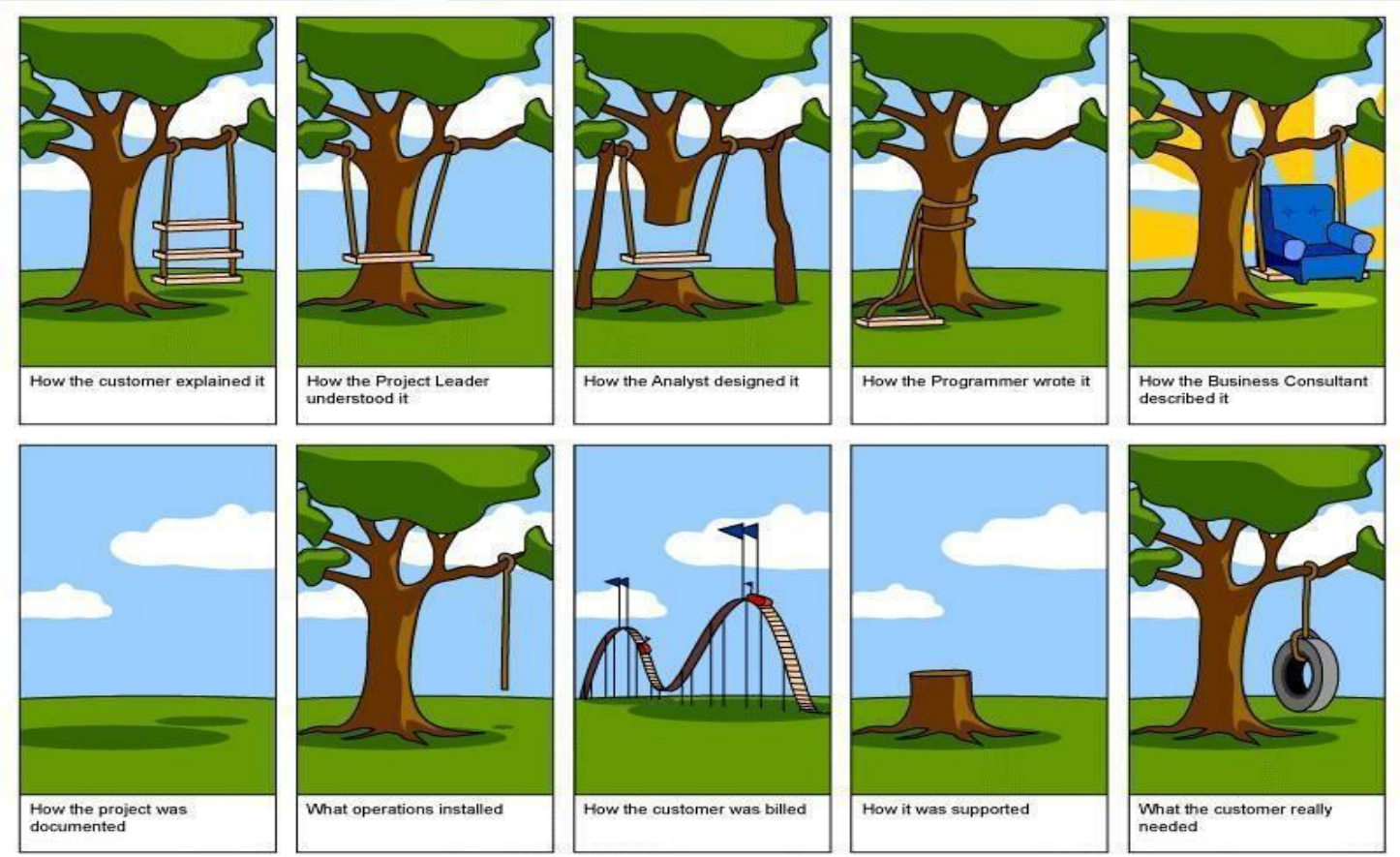
- System testing is concerned with the behavior of the whole system/product as defined by the scope of a development project or product.
- System testing is most often the final test on behalf of development to verify that the system to be delivered meets the specification and its purpose may be to find as many defects as possible.
- Most often it is carried out by specialist testers that form a dedicated, and sometimes independent, test team within development, reporting to the development manager or project manager.

TEST LEVELS

System Testing

- It may include tests based on risks and/or requirements specification, business processes, use cases, or other high level descriptions of system behavior, interactions with the operating system, and system resources.
- System testing should investigate both functional and non-functional requirements of the system.
- The test environment should correspond to the final target or production environment as much as possible in order to minimize the risk of environment-specific failures not being found by testing.

TEST LEVELS

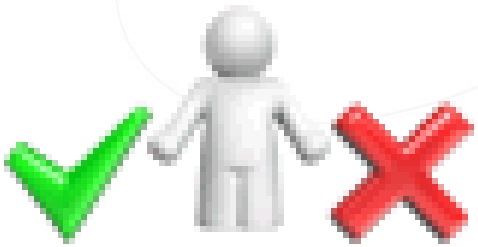


TURKCELL

TURKCELL AKADEMİ

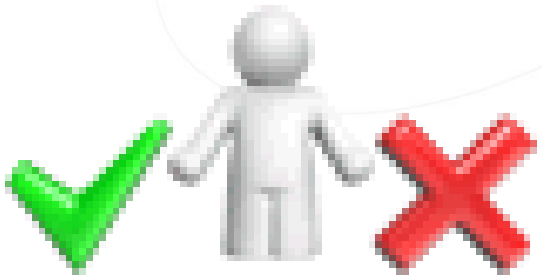
YILDIZ TEKNİK ÜNİVERSİTESİ 1911

User Acceptance Testing



- The goal of acceptance testing is to establish confidence in the system, part of the system or specific non-functional characteristics, e.g. usability, of the system and most often the responsibility of the user or customer.
- The user acceptance test validates whether the system meets the requirements for operation.
- The user acceptance test is performed by the users and application managers.
- The user acceptance test focuses mainly on the functionality thereby validating the fitness-for-use of the system by the business user.

User Acceptance Testing



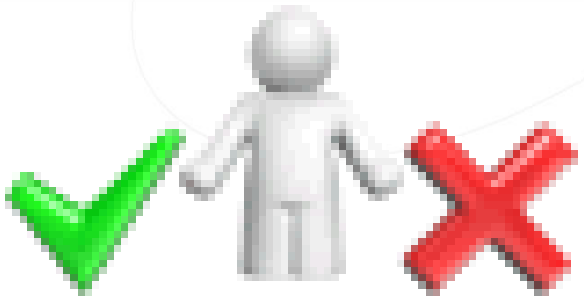
- **User Acceptance Test (Functionality testing with business user)**

Focuses mainly on the functionality thereby validating the fitness-for-use of the system by the business user.

- **User Acceptance Test (Contract and compliance acceptance testing)**

Contract acceptance testing is performed against a contract's acceptance criteria for producing custom-developed software. Compliance acceptance testing is performed against the regulations.

User Acceptance Testing

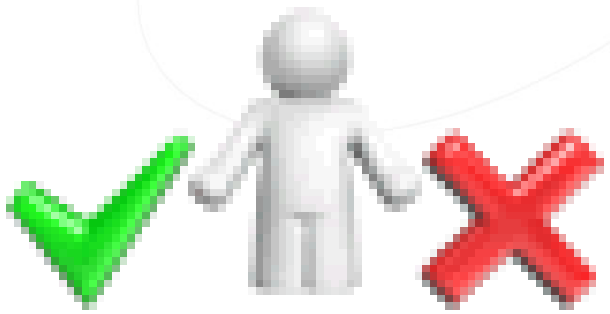


User Acceptance Test (Operational / Production acceptance test)

The operational acceptance test may include testing of

- Backup / Restore
- User management
- Maintenance tasks
- Periodic check of security vulnerabilities
- Disaster recovery

User Acceptance Testing



User Acceptance Test (Alpha-Beta Testing)

- The test is made by system' s real users.
- Alpha testing takes place at the developer's site.
- Beta testing, or field testing, sends the system to a cross-section of users who install it and use it under real-world working conditions.
- Detecting the defects on beta tests reduces the confidence of the system.

TURKCELL AKADEMİ

Teşekkürler...