SE2250 – Software Construction

Lecture #9

Specification documents and test plans

25 March, 2019

Question 1

- Which one of the following is NOT a game design goal?
  A. Greater good
  B. Belonging to community
  C. Flow
  D. Lusory attitude
  E. All are goals

Question 2

- Which statement is correct?
  A. We can not change flow to accommodate user abilities
  B. Flow is the movement of game objects
  C. Flow describes order in which events occur in the game
  D. Flow is between boredom and frustration

Question 3

- Following all guidelines for game design, player guidance, and game balance is mandatory for delivering a successful, high selling game?

A. TRUE

B. FALSE

Question 4

- What would you expect to see the most often in video game design and development?
  A. Probabilities
  B. Distributions
  C. Spreadsheets
  D. Permutations

Outline

- Introduction
- Development testing
  - Test driven development
- Release testing
- User testing
- Key points

Introduction

- Testing is intended to show that a program does what it is intended to do and to discover program defects before it is put into use
- You check the results of the test run for errors, anomalies or information about the program's non-functional attributes

- **Can reveal the presence of errors - NOT their absence**

Introduction

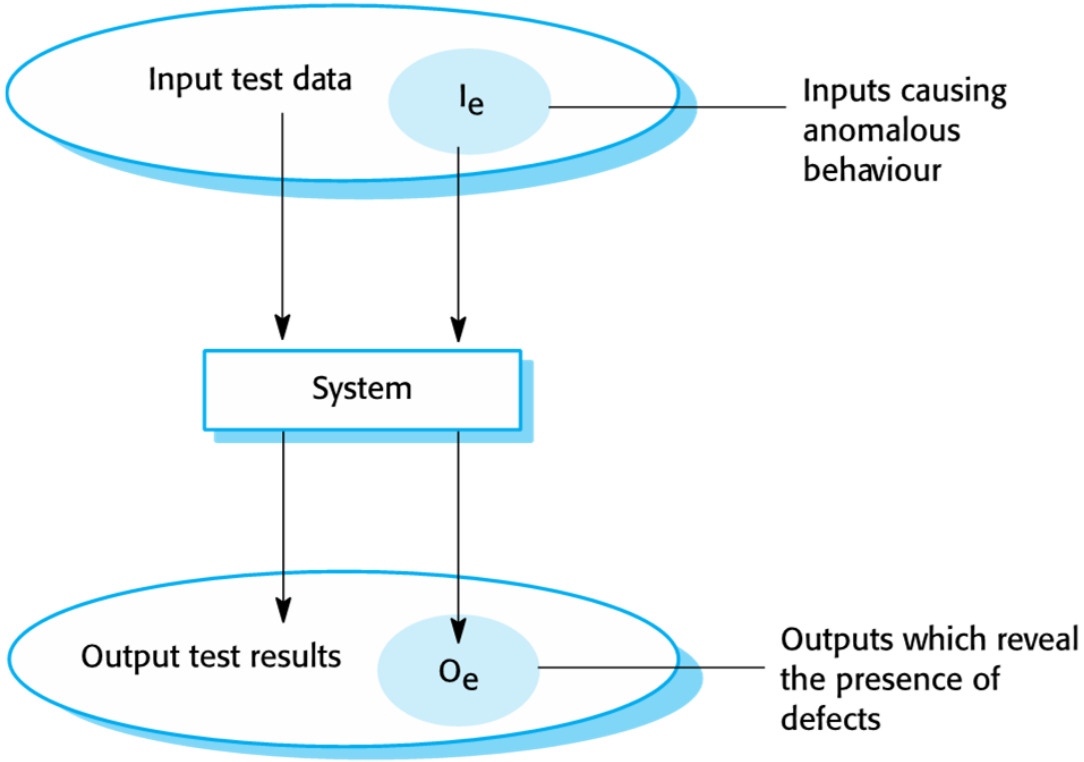
- To demonstrate that the software meets its requirements
  - At least one test for every requirement in the requirements document
  - Tests for all of the system features, plus combinations of these features
- To discover situations in which the behavior of the software is incorrect, undesirable or does not conform to its specification
  - Concerned with rooting out undesirable system behavior such as system crashes, unwanted interactions with other systems, incorrect computations and data corruption

Introduction

- Validation testing
  - To demonstrate to the developer and the system customer that the software meets its requirements
  - A successful test shows that the system operates as intended
- Defect testing
  - To discover faults or defects in the software where its behaviour is incorrect or not in conformance with its specification
  - A successful test is a test that makes the system perform incorrectly and so exposes a defect in the system

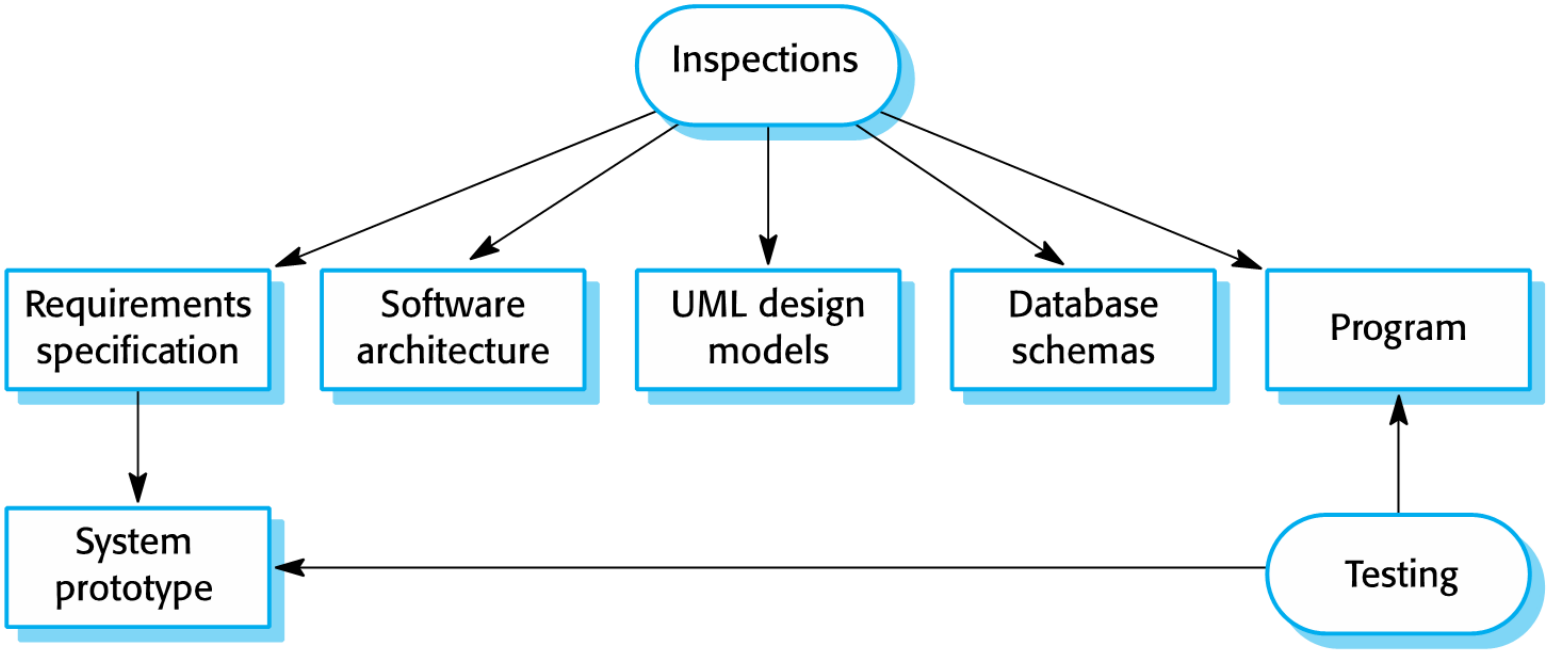
Introduction

- An input-output model of program testing



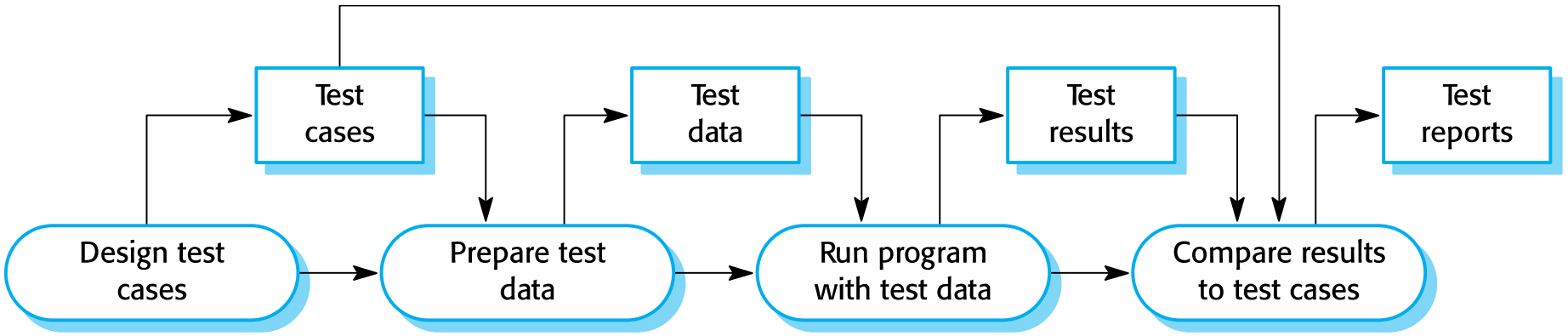
Introduction

- Inspections vs testing
  - Inspections - concerned with analysis of the static system representation to discover problems (static verification)
  - Testing - concerned with exercising and observing product behaviour (dynamic verification)



Introduction

- Software testing process



Introduction

- Testing stages:
  - **Development testing** - the system is tested during development to discover bugs and defects
  - **Release testing** - a separate testing team test a complete version of the system before it is released to users
  - **User testing** - users or potential users of a system test the system in their own environment

Development testing

- **Development testing** - testing activities that are carried out by the team developing the system.
  - Unit testing - individual program units/objects/classes are tested
  - Component testing - several individual units are integrated to create composite components
  - System testing - some or all of the components in a system are integrated and the system is tested as a whole

Development testing – Unit testing

- **Unit testing**
- The process of testing individual components in isolation
- It is a defect testing process
- Units may be:
  - Individual functions or methods within an object
  - Object classes with several attributes and methods
  - Composite components with defined interfaces used to access their functionality

- **Unit -** the smallest testable part of an application

Development testing – Unit testing

- Complete test coverage of a class involves
  - Testing all operations associated with an object
  - Setting and interrogating all object attributes
  - Exercising the object in all possible states

Development testing – Unit testing

- Unit testing – automated testing
  - Whenever possible, unit testing should be automated so that tests are run and checked without manual intervention
  - In automated unit testing, you make use of a test automation framework (such as JUnit) to write and run your program tests
  - Unit testing frameworks provide generic test classes that you extend to create specific test cases. They can then run all of the tests that you have implemented and report, often through some GUI, on the success of the tests.

Development testing – Unit testing

- Unit testing – components:
  - *A setup part* where you initialize the system with the test case, namely the inputs and expected outputs.
  - *A call part* where you call the object or method to be tested
  - *An assertion part* where you compare the result of the call with the expected result. If the assertion evaluates to true, the test has been successful; if false, then it has failed

  - Unity Demo

Development testing – Unit testing

- Unit testing – two types:
  - The first of these should reflect normal operation of a program and should show that the component works as expected
  - The other kind of test case should be based on testing experience of where common problems arise. It should use abnormal inputs to check that these are properly processed and do not crash the component

Development testing – Unit testing

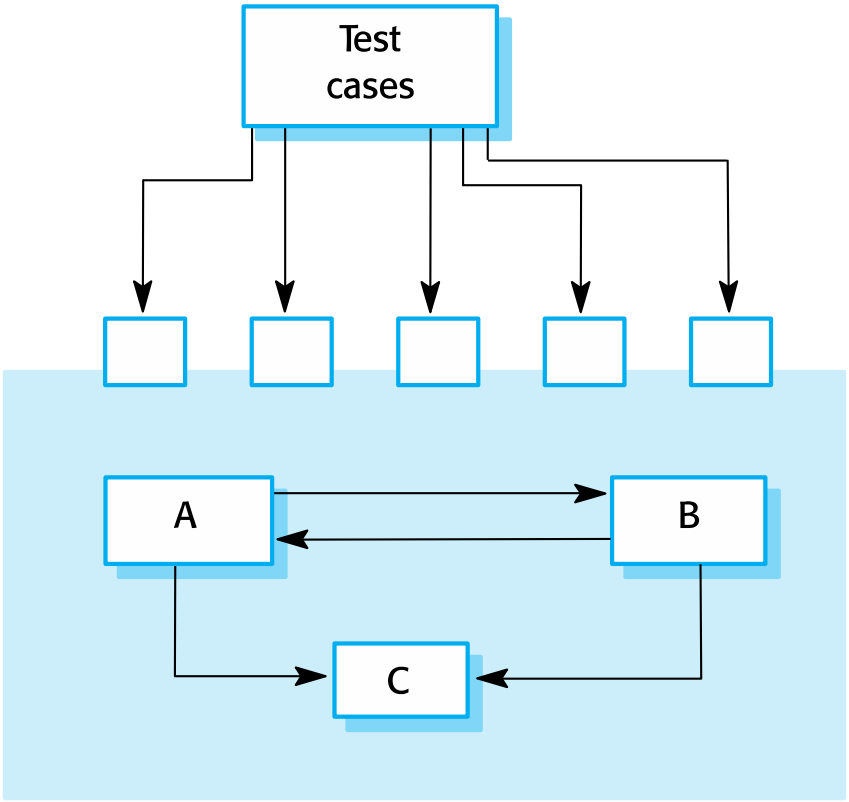
- **Tips:**
  - Choose inputs that force the system to generate all error messages
  - Design inputs that cause input buffers to overflow
  - Repeat the same input or series of inputs numerous times
  - Force invalid outputs to be generated
  - Force computation results to be too large or too small

Development testing – Component testing

- Software components are often composite components that are made up of several interacting objects
  - Example: In the insurance software, the policy component deals with each aspect of the insurance policy
- You access the functionality of these objects through the defined component interface.
- Testing composite components should therefore focus on showing that the component interface behaves according to its specification
  - *Assumption* - unit tests on the individual objects within the component have been completed

Development testing – Component testing

- Component testing



Development testing – Component testing

- Interface testing guidelines
  - Design tests so that parameters to a called procedure are at the extreme ends of their ranges
  - Always test pointer parameters with null pointers
  - Design tests which cause the component to fail
  - Use stress testing in message passing systems
  - In shared memory systems, vary the order in which components are activated

Development testing – System testing

- System testing during development involves integrating components to create a *version of the system* and then testing the integrated system
- The focus in system testing is testing the interactions between components
- System testing checks that components are compatible, interact correctly and transfer the right data at the right time across their interfaces
- **System testing tests the emergent behaviour of a system**

Development testing – System testing

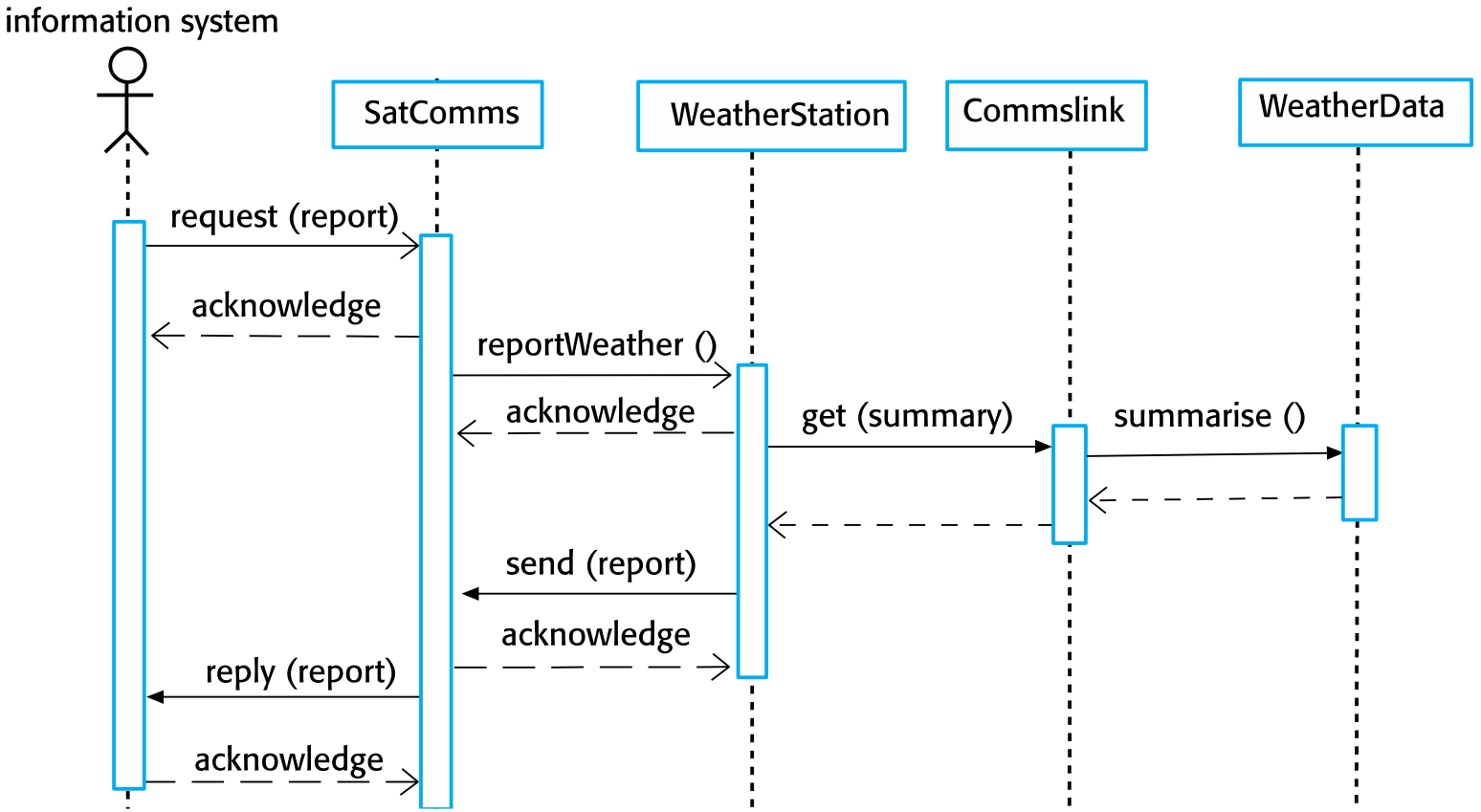
- During system testing, reusable components that have been separately developed and off-the-shelf systems may be integrated with newly developed components. The complete system is then tested.
- Components developed by different team members or sub-teams may be integrated at this stage. System testing is a collective rather than an individual process.
  - In some companies, system testing may involve a separate testing team with no involvement from designers and programmers.

Development testing – System testing

- The use-cases developed to identify system interactions can be used as a basis for system testing.
- Each use case usually involves several system components so testing the use case forces these interactions to occur.
- The sequence diagrams associated with the use case documents the components and interactions that are being tested.

Development testing – System testing

- Test cases derived from sequence diagrams



Development testing – System testing

- Exhaustive system testing is impossible
- Use testing policies

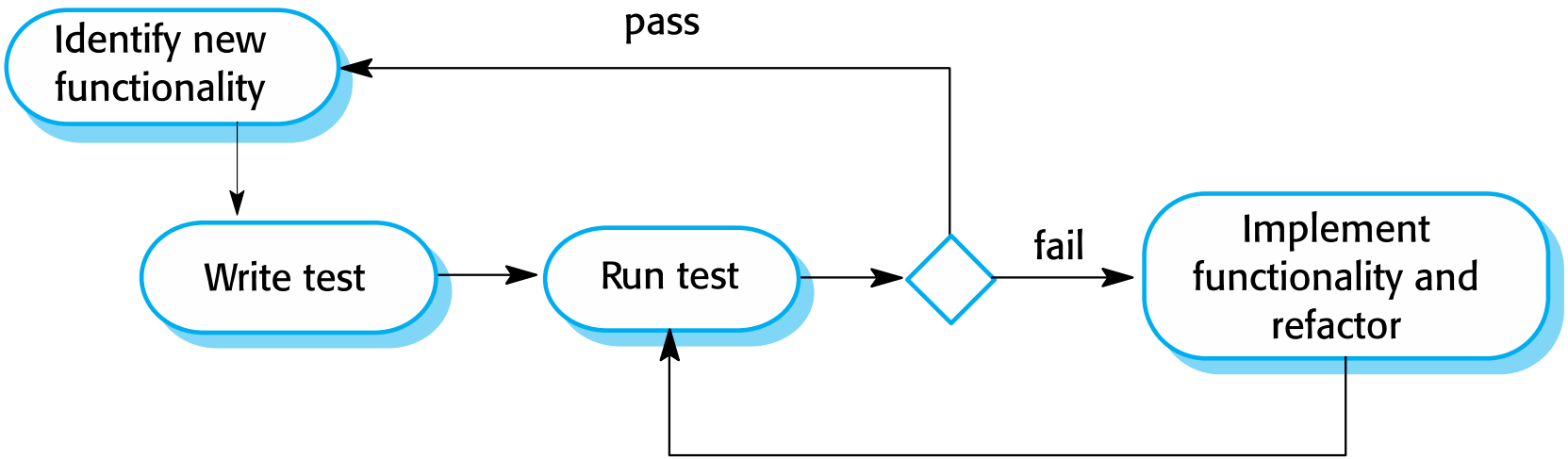
- Examples of testing policies:
  - All system functions that are accessed through menus should be tested
  - Combinations of functions (e.g. text formatting) that are accessed through the same menu must be tested
  - Where user input is provided, all functions must be tested with both **correct** and **incorrect** input

Development testing – Test-driven development

- Test-driven development (TDD)
  - Tests are written before code.
  - ‘passing’ the tests is the critical driver of development.
  - You develop code incrementally, along with a test for that increment. You don’t move on to the next increment until the code that you have developed passes its test.
  - TDD was introduced as part of agile methods such as Extreme Programming. However, it can also be used in plan-driven development processes.

Development testing – Test-driven development

- Test-driven development (TDD)



Release testing

- **Release Testing**
- Testing a particular release of a system that is intended for use outside of the development team
- Goal: convince the supplier of the system that it is good enough for use
  - Has to show that the system delivers its specified functionality, performance and dependability, and that it does not fail during normal use
- Usually a black-box testing process where tests are only derived from the system specification

Release testing

- Release testing is a form of system testing
- Differences:
  - A **separate team** that has not been involved in the system development, should be responsible for release testing.
  - **System** testing by the development team should focus on discovering **bugs** in the system (defect testing). The objective of **release** testing is to check that the system **meets its requirements** and is good enough for external use (validation testing).

Release testing

- Requirements-based testing involves examining each requirement and developing a test or tests for it
- Examples:
  - Set up a patient record with no known allergies. Prescribe medication for allergies that are known to exist. Check that a warning message is not issued by the system.
  - Set up a patient record with a known allergy. Prescribe the medication to that the patient is allergic to, and check that the warning is issued by the system.
  - Prescribe two drugs that the patient is allergic to. Check that two warnings are correctly issued.
  - …

SE2250 Project

Release testing

- Performance testing
  - Part of release testing
  - Tests should reflect the profile of use of the system
  - Performance tests usually involve planning a series of tests where the load is steadily increased until the system performance becomes unacceptable
  - Stress testing is a form of performance testing where the system is deliberately overloaded to test its failure behaviour

User testing

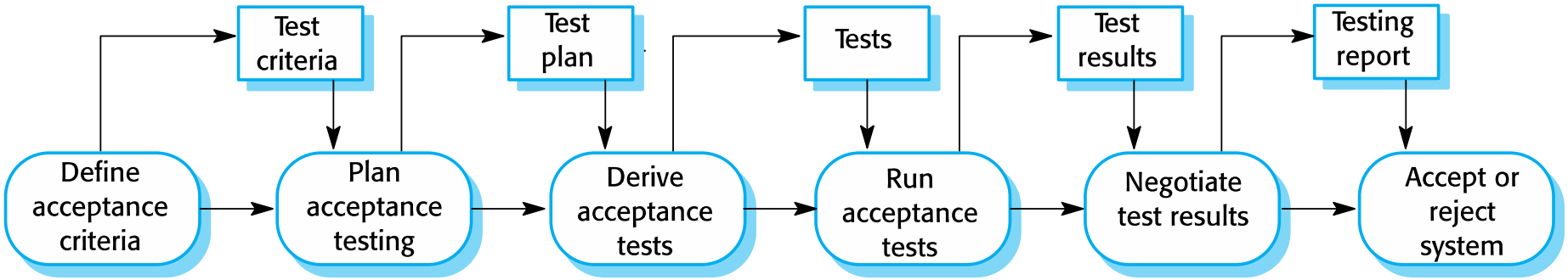
- **User testing**
  - Users or potential users of a system test the system in their own environment
  - User testing is essential, even when comprehensive system and release testing have been carried out.
    - Reason - influences from the user's working environment have a major effect on the reliability, performance, usability and robustness of a system. These cannot be replicated in a testing environment.

User testing

- Alpha testing
  - Users of the software work with the development team to test the software at the developer's site.
- Beta testing
  - A release of the software is made available to users to allow them to experiment and to raise problems that they discover with the system developers.
- Acceptance testing
  - Customers test a system to decide whether or not it is ready to be accepted from the system developers and deployed in the customer environment. Primarily for custom systems.

User testing

- Acceptance testing



Key Points

- Key points:
  - Testing can only show the presence of errors in a program. It cannot demonstrate that there are no remaining faults.
  - Development testing is the responsibility of the software development team. A separate team should be responsible for testing a system before it is released to customers.
  - Development testing includes unit testing, in which you test individual objects and methods, component testing in which you test related groups of objects and system testing, in which you test partial or complete systems.

Key Points

- Key points:
  - When testing software, you should **try to 'break'** the software by using experience and guidelines to choose types of test case that have been effective in discovering defects in other systems.
  - Wherever possible, you should write automated tests. The tests are embedded in a program that can be run every time a change is made to a system.
  - Test-first development is an approach to development where tests are written before the code to be tested.
  - Acceptance testing is a user testing process where the aim is to decide if the software is good enough to be deployed and used in its operational environment.