

Chapter 8 – Software Testing

Program testing



- ✧ 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.
- ✧ When you test software, you execute a program using **artificial data**.
- ✧ 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.
- ✧ Testing is part of a more general verification and validation process, which also includes **static** validation techniques.

Program testing goals



- ✧ To demonstrate to the developer and the customer that the software meets its requirements.
 - For custom software, this means that there should be at least one test for every requirement in the requirements document. For generic software products, it means that there should be tests for all of the system features, plus combinations of these features, that will be incorporated in the product release.
- ✧ To discover situations in which the behavior of the software is incorrect, undesirable or does not conform to its specification.
 - Defect testing is concerned with rooting out undesirable system behavior such as system crashes, unwanted interactions with other systems, incorrect computations and data corruption.

Validation and defect testing



- ✧ The first goal leads to validation testing
 - You expect the system to perform correctly using a given set of test cases that reflect the system's expected use.
- ✧ The second goal leads to defect testing
 - The test cases are designed to expose defects. The test cases in defect testing can be deliberately obscure and need not reflect how the system is normally used.

Verification vs validation



- ✧ Verification:
 - "Are we building the product **right**".
- ✧ The software should conform to its specification.
- ✧ Validation:
 - "Are we building the **right** product".
- ✧ The software should do what the user really requires.

V & V confidence



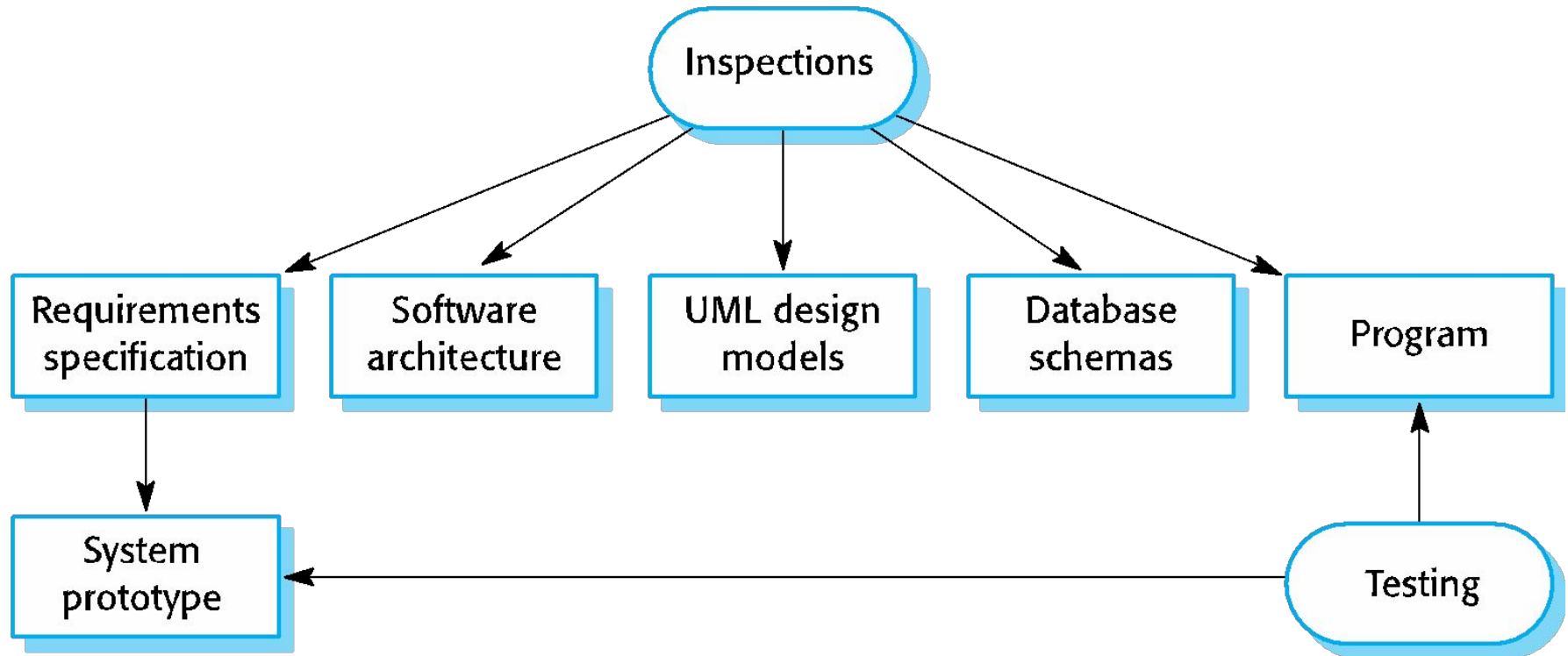
- ✧ Aim of V & V is to establish confidence that the system is 'fit for purpose'.
- ✧ Depends on system's purpose, user expectations and marketing environment
 - Software purpose
 - The level of confidence depends on how critical the software is to an organisation.
 - User expectations
 - Users may have low expectations of certain kinds of software.
 - Marketing environment
 - Getting a product to market early may be more important than finding defects in the program.

Inspections and testing

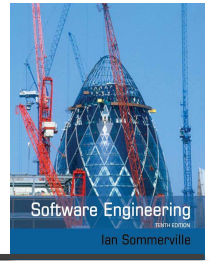


- ✧ **Software inspections** Concerned with analysis of the static system representation to discover problems (**static verification**)
 - May be supplemented by tool-based document and code analysis.
- ✧ **Software testing** Concerned with exercising and observing product behaviour (dynamic verification)
 - The system is executed with test data and its operational behaviour is observed.

Inspections and testing



Software inspections



- ✧ These involve people **examining** the source representation to discover anomalies and defects.
- ✧ Inspections do not require execution of a system so may be used before implementation.
- ✧ They may be applied to any representation of the system (requirements, design, configuration data, test data, etc.).
- ✧ They are an effective technique for discovering program errors.

Advantages of inspections



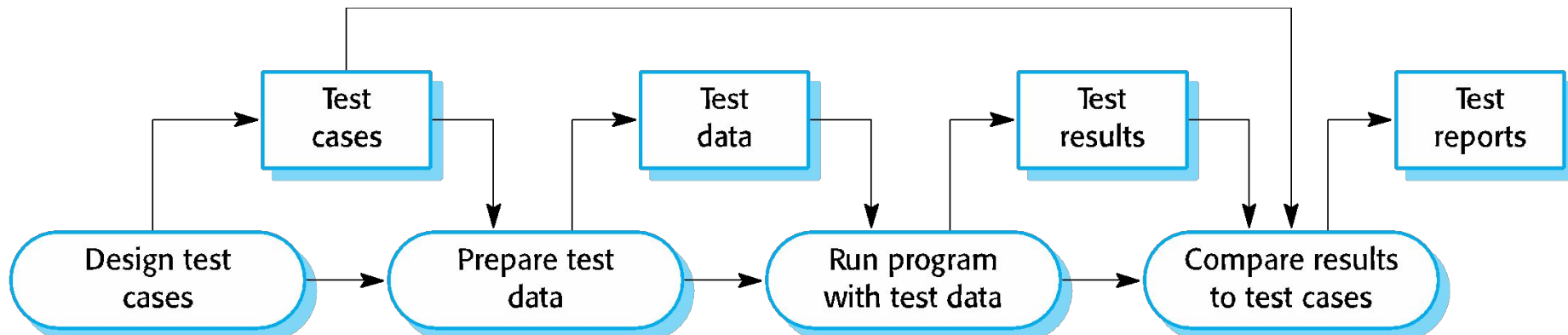
- ✧ During testing, errors can mask (hide) other errors. Because inspection is a static process, you don't have to be concerned with interactions between errors.
- ✧ Incomplete versions of a system can be inspected without additional costs. If a program is incomplete, then you need to develop specialized test harnesses to test the parts that are available.
- ✧ As well as searching for program defects, an inspection can also consider broader quality attributes of a program, such as compliance with standards, portability and maintainability.

Inspections and testing



- ✧ Inspections and testing are complementary and not opposing verification techniques.
- ✧ Both should be used during the V & V process.
- ✧ Inspections can check conformance with a specification but not conformance with the customer's real requirements.
- ✧ Inspections cannot check non-functional characteristics such as performance, usability, etc.

A model of the software testing process



Stages of testing



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

Development testing

Development testing



- ✧ Development testing includes all testing activities that are carried out by the team developing the system.
 - **Unit testing**, where individual program units or object classes are tested. Unit testing should focus on testing the functionality of objects or methods.
 - **Component testing**, where several individual units are integrated to create composite components. Component testing should focus on testing component interfaces.
 - **System testing**, where some or all of the components in a system are integrated and the system is tested as a whole. System testing should focus on testing component interactions.

Unit testing



- ✧ Unit testing is 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.

Object class 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.
- ✧ Inheritance makes it more difficult to design object class tests as the information to be tested is not localised.

The weather station object interface



WeatherStation

identifier

reportWeather ()

reportStatus ()

powerSave (instruments)

remoteControl (commands)

reconfigure (commands)

restart (instruments)

shutdown (instruments)

Weather station testing



- ✧ Need to define test cases for reportWeather, calibrate, test, startup and shutdown.
- ✧ Using a state model, identify sequences of state transitions to be tested and the event sequences to cause these transitions
- ✧ For example:
 - Shutdown -> Running-> Shutdown
 - Configuring-> Running-> Testing -> Transmitting -> Running
 - Running-> Collecting-> Running-> Summarizing -> Transmitting -> Running

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 or otherwise of the tests.

Choosing unit test cases



- ✧ The test cases should show that, when used as expected, the component that you are testing **does what it is supposed to do**.
- ✧ If there are **defects** in the component, these should be revealed by test cases.
- ✧ This leads to 2 types of unit test cases:
 - The first should reflect the 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.

Component testing



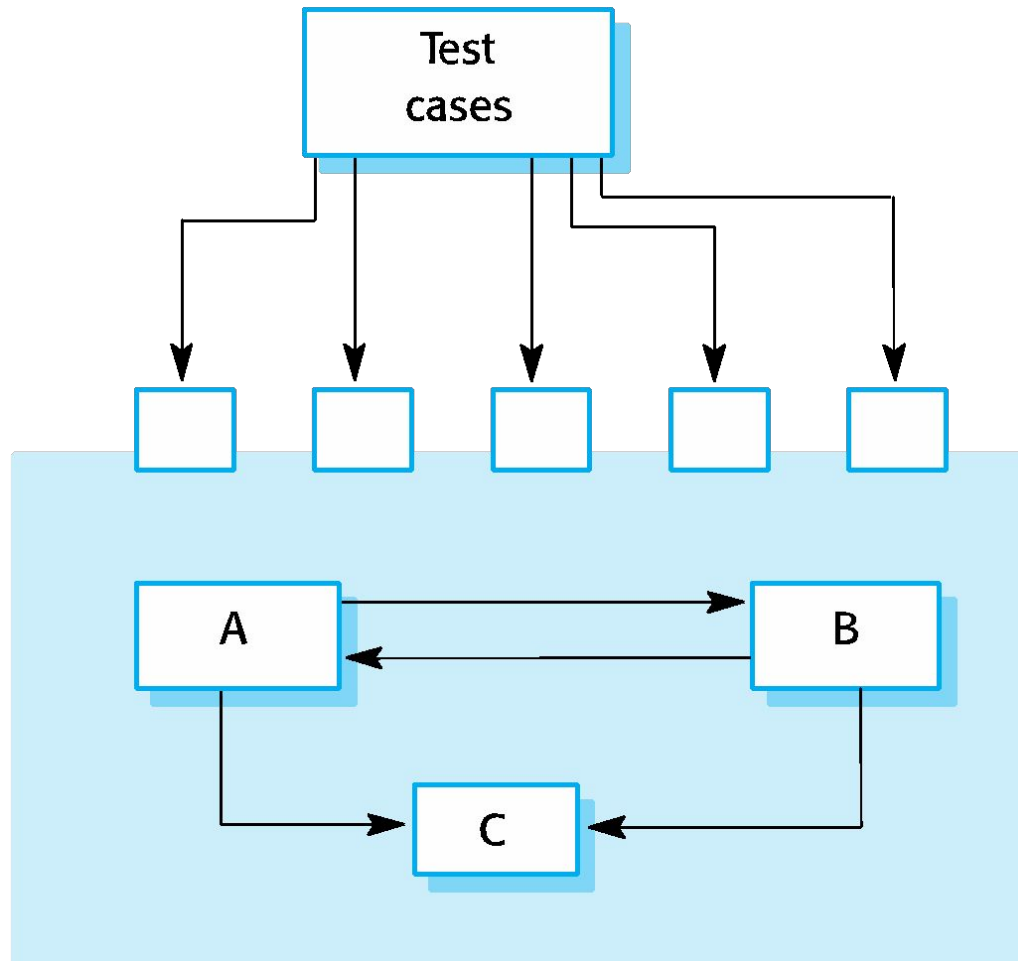
- ✧ Software components are often **composite** components that are made up of several interacting objects.
 - For example, in the weather station system, the reconfiguration component includes objects that deal with each aspect of the reconfiguration.
- ✧ You access the functionality of these objects through the defined **component interface**.
- ✧ Testing composite components should therefore focus on showing that the component interface behaves according to its specification.
 - You can assume that unit tests on the individual objects within the component have been completed.

Interface testing



- ✧ Objectives are to detect faults due to interface errors or invalid assumptions about interfaces.
- ✧ Interface types
 - **Parameter interfaces** Data passed from one method or procedure to another.
 - **Shared memory interfaces** Block of memory is shared between procedures or functions.
 - **Procedural interfaces** Sub-system encapsulates a set of procedures to be called by other sub-systems.
 - **Message passing interfaces** Sub-systems request services from other sub-systems

Interface testing



Interface errors



✧ Interface misuse

- A calling component calls another component and makes an error in its use of its interface e.g. parameters in the wrong order.

✧ Interface misunderstanding

- A calling component embeds assumptions about the behaviour of the called component which are incorrect.

✧ Timing errors

- The called and the calling component operate at different speeds and out-of-date information is accessed.

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.

System and component testing



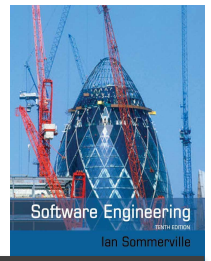
- ✧ During system testing, reusable components that have been separately developed and off-the-shelf systems may be integrated with newly developed components. The complete system is then tested.
- ✧ Components developed by different team members or sub-teams may be integrated at this stage. System testing 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.

Use-case testing

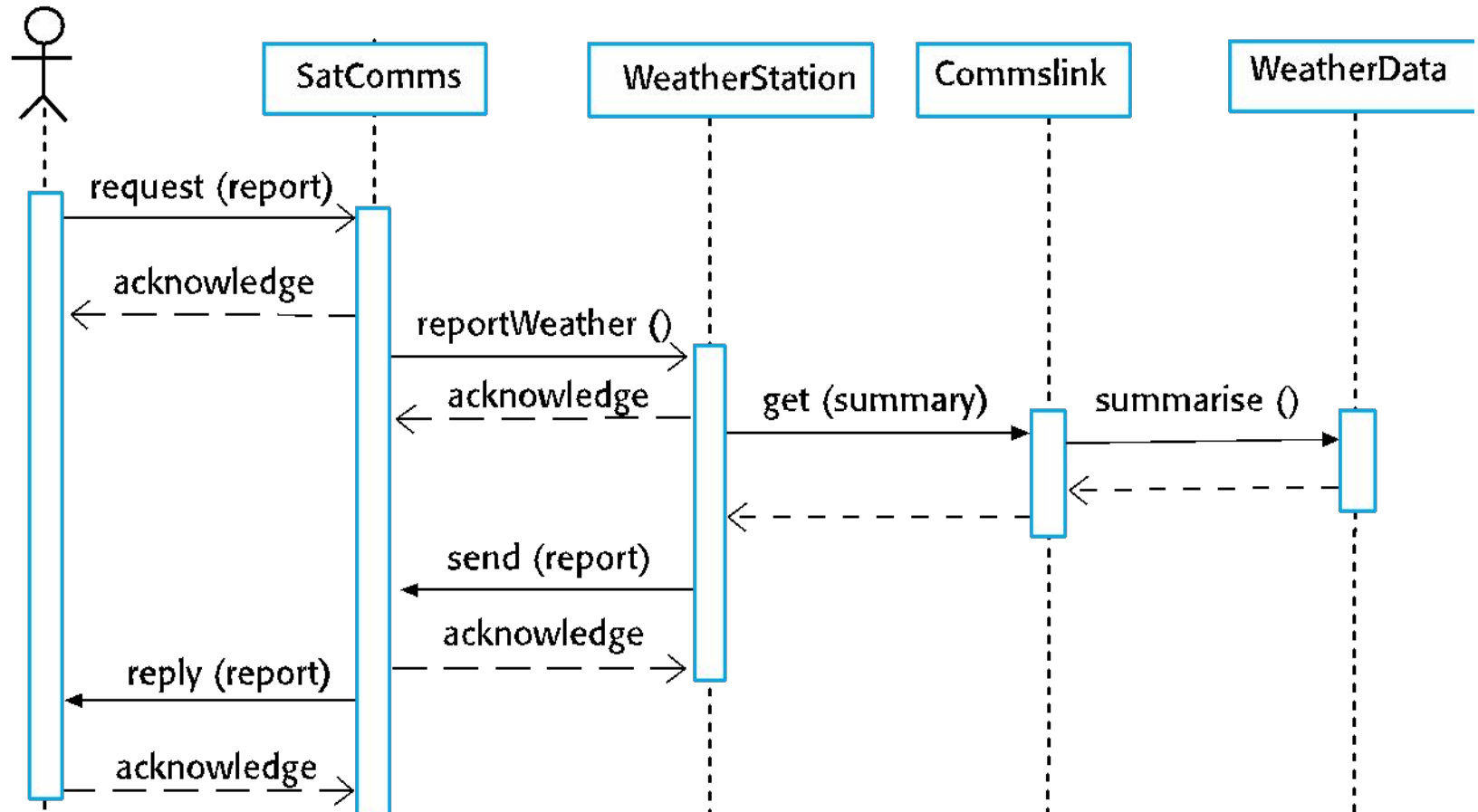


- ✧ The use-cases developed to identify system interactions can be used as a basis for system testing.
- ✧ Each use case usually involves several system components so testing the use case forces these interactions to occur.
- ✧ The sequence diagrams associated with the use case documents the components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

Collect weather data sequence chart



information system



Test cases derived from sequence diagram



- ✧ An input of a request for a report should have an associated acknowledgement. A report should ultimately be returned from the request.
 - You should create summarized data that can be used to check that the report is correctly organized.
- ✧ An input request for a report to WeatherStation results in a summarized report being generated.
 - Can be tested by creating raw data corresponding to the summary that you have prepared for the test of SatComms and checking that the WeatherStation object correctly produces this summary. This raw data is also used to test the WeatherData object.

Testing policies



- ✧ Exhaustive system testing is impossible so testing policies which define the required system test coverage may be developed.
- ✧ 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.

Regression testing



- ✧ **Regression testing** is testing the system to check that changes have not 'broken' previously working code.
- ✧ In a manual testing process, regression testing is expensive but, with automated testing, it is simple and straightforward. All tests are rerun every time a change is made to the program.
- ✧ Tests must run 'successfully' before the change is committed.

Release testing

Release testing



- ✧ **Release testing** is the process of testing a particular release of a system that is intended for use outside of the development team.
- ✧ The primary goal of the release testing process is to convince the supplier of the system that it is good enough for use.
 - Release testing, therefore, has to show that the system delivers its specified functionality, performance and dependability, and that it does not fail during normal use.
- ✧ Release testing is usually a black-box testing process where tests are only derived from the system specification.

Release testing and system testing



- ✧ Release testing is a form of system testing.
- ✧ Important 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).

Requirements based testing



- ✧ Requirements-based testing involves examining each requirement and developing a test or tests for it.
- ✧ Mentcare system requirements:
 - If a patient is known to be allergic to any particular medication, then prescription of that medication shall result in a warning message being issued to the system user.
 - If a prescriber chooses to ignore an allergy warning, they shall provide a reason why this has been ignored.

Performance testing



- ✧ Part of release testing may involve testing the emergent properties of a system, such as performance and reliability.
- ✧ 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



- ✧ User or customer testing is a stage in the testing process in which users or customers provide input and advice on system testing.
- ✧ User testing is essential, even when comprehensive system and release testing have been carried out.
 - The reason for this is that 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.

Types of 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.

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



- ✧ 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.
- ✧ Scenario testing involves inventing a typical usage scenario and using this to derive test cases.
- ✧ 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.