Software Testing

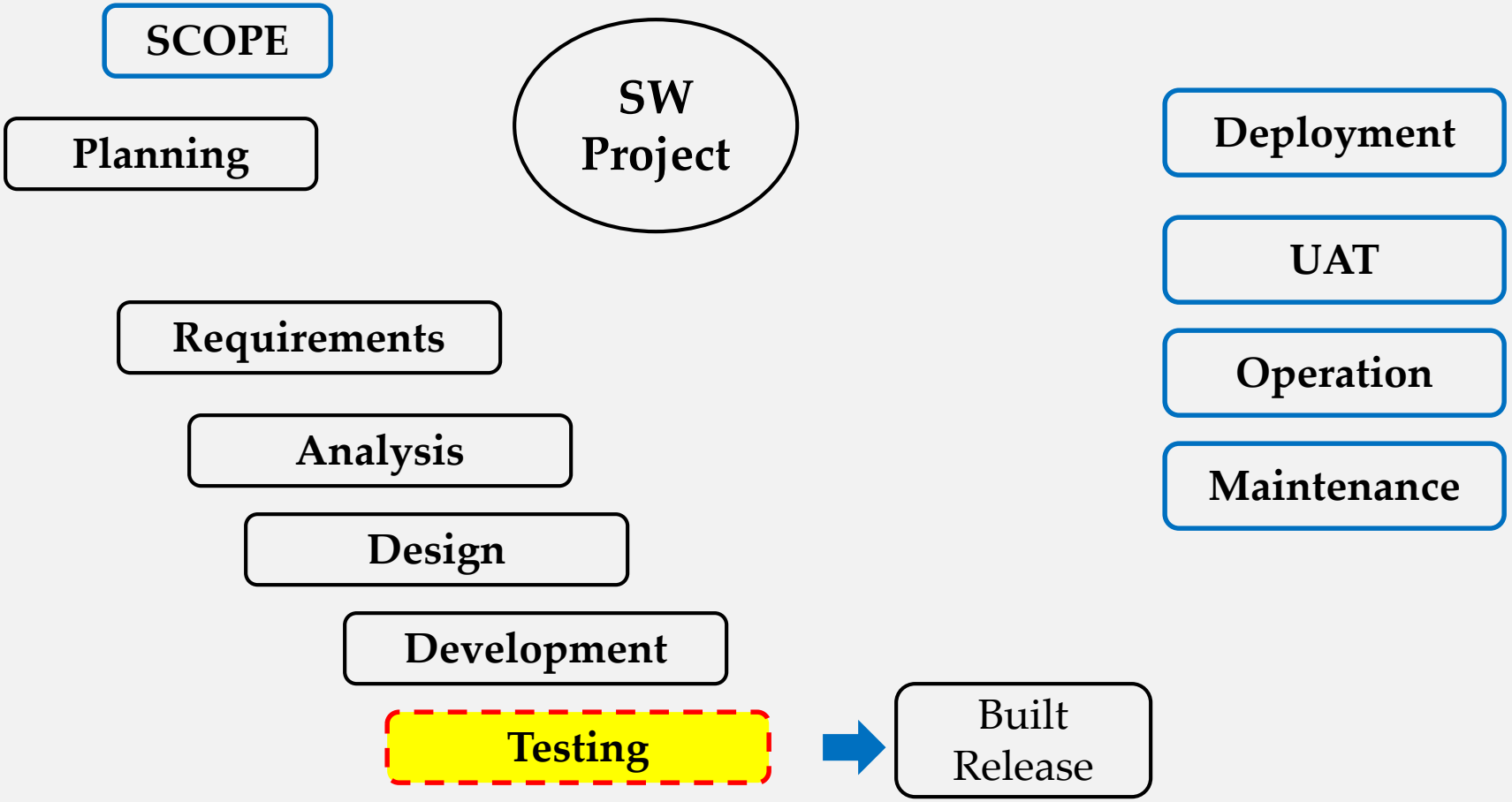
Lecture # 34, 35,
36,37,38,39
16,21,22,23,28,29 April

Rubab Jaffar
rubab.jaffar@nu.edu.pk

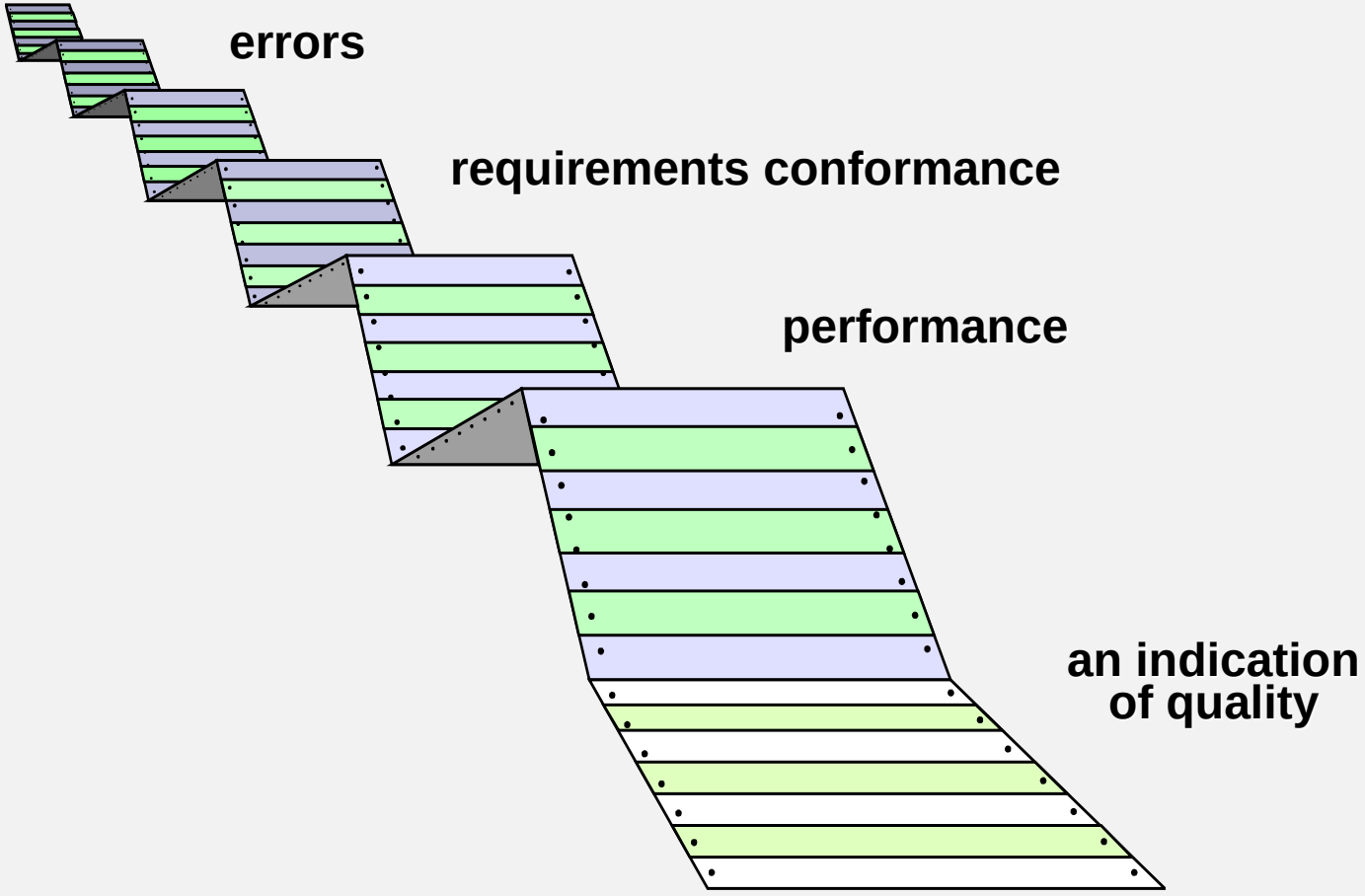
Software Engineering
CS-303

Topics covered

- Development testing
- Test-driven development
- Release testing
- User testing


SCOPE
Planning
SW Project
Deployment
UAT
Operation
Maintenance
Requirements
Analysis
Design
Development
Testing
Built Release

What Testing Shows



Software testing- Definition

- **Testing is the process of evaluating a system or its component(s) with the intent to find whether it satisfies the specified requirements or not.**
- **Software testing is a process of analyzing or operating software for the purpose of finding bugs.**
- **Software testing is the process of testing the *functionality* and correctness of software by running it.**
- **Software testing is usually performed for one of two reasons:**
    1) **Defect detection**
    2) **Reliability estimation**

Why Software Testing?

- **An investigation conducted to provide stakeholders with information about the quality of the software under test.**

- **To detect failures so that defects may be discovered and corrected.**

- **Testing cannot establish that a product functions properly under all conditions; but can only establish that it does not function properly under specific conditions**

Manual Testing

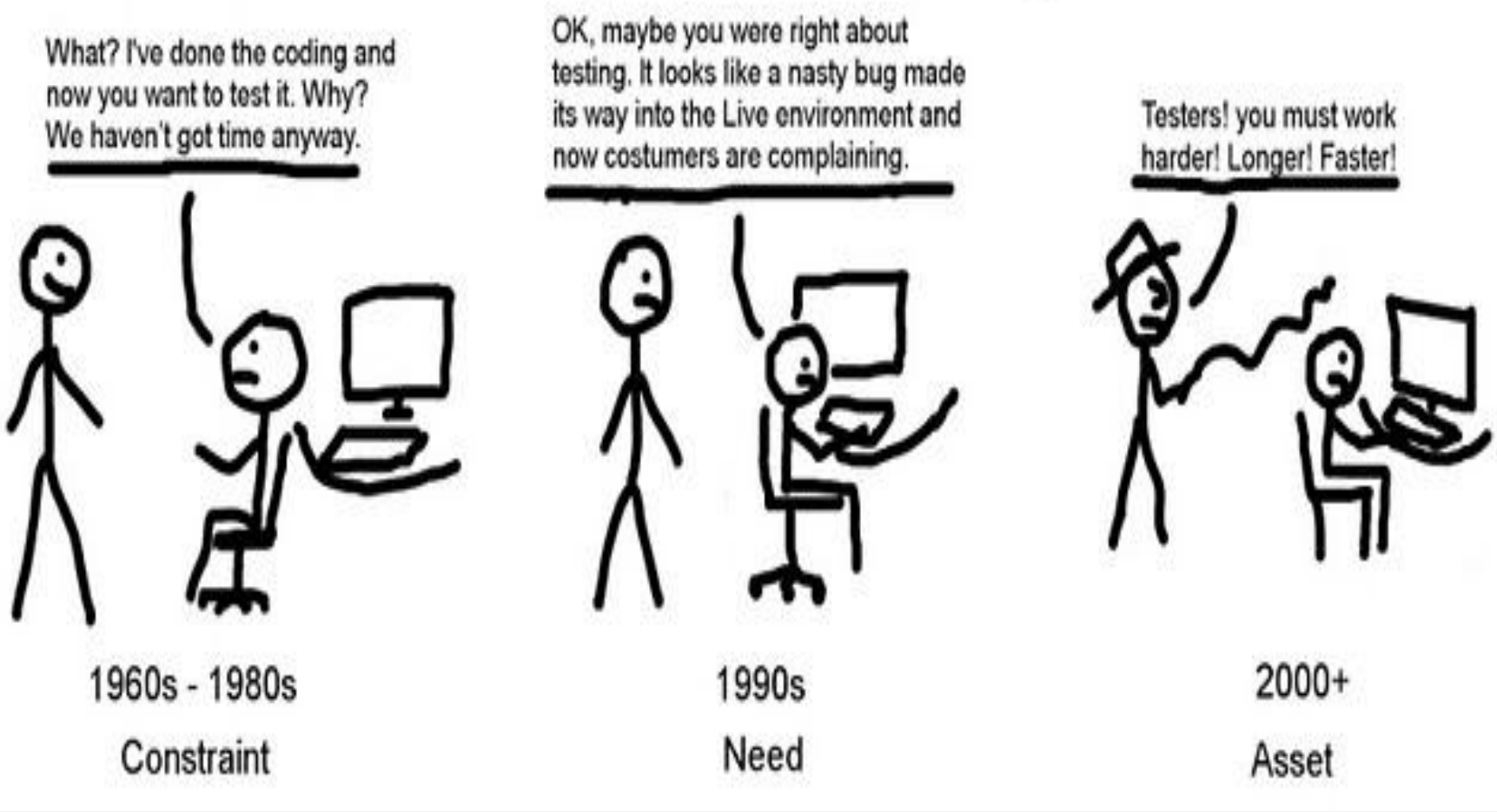
- **Manual testing includes testing a software manually, i.e., without using any automated tool or any script. In this type, the tester takes over the role of an end-user and tests the software to identify any unexpected behavior or bug.**
- **Testers use test plans, test cases, or test scenarios to test a software to ensure the completeness of testing.**

Automation Testing

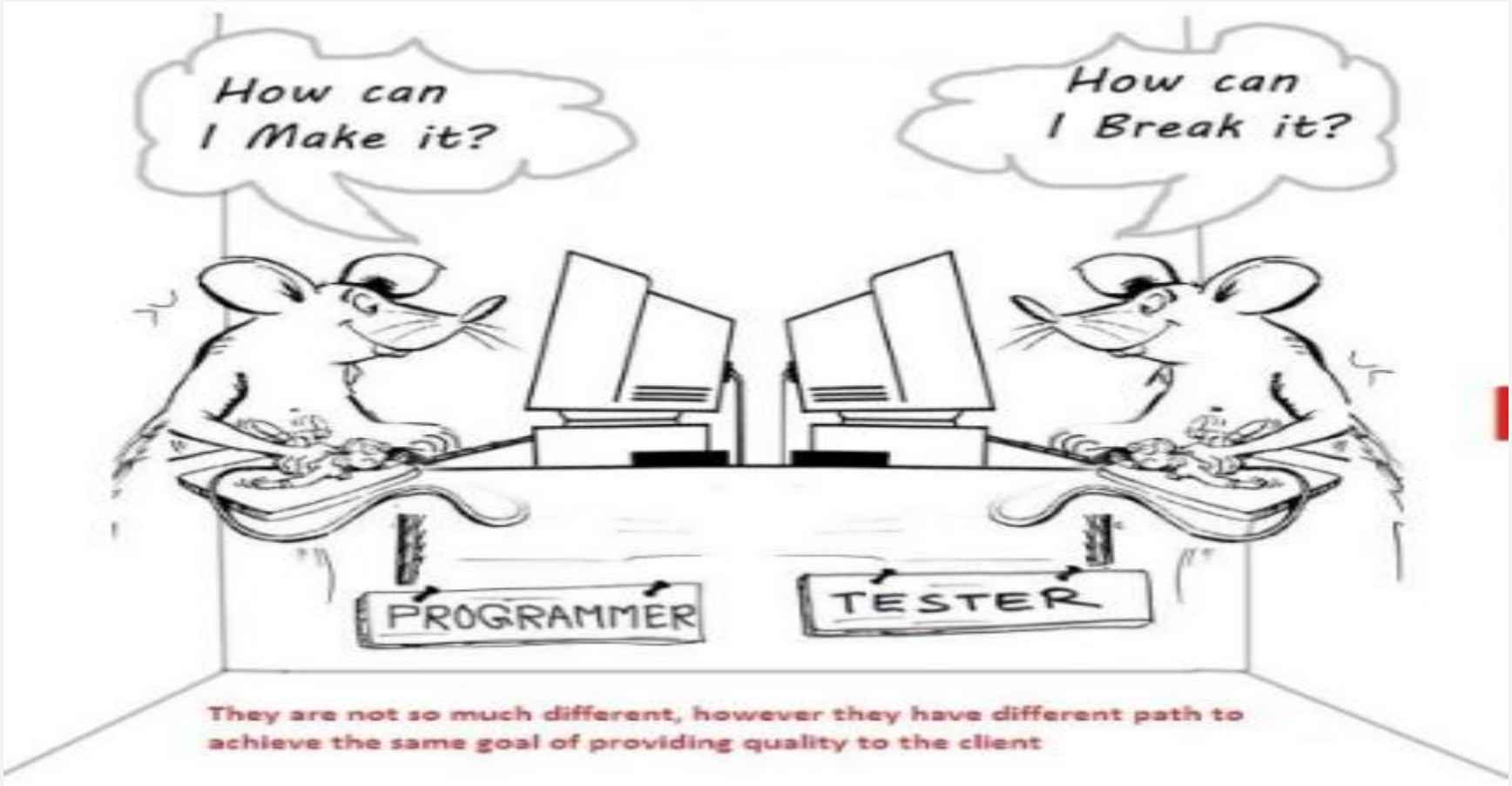
- **When the tester writes scripts and uses it to test the product. This process involves automation of a manual process.**
- **Automation Testing is used to re-run the test scenarios that were performed manually, quickly, and repeatedly.**

If a manual test costs $X to run the first time, it will cost $X to run every time thereafter. An automated test can cost 3 to 30 times $X the first time, but will cost about $0 after that.

History of Software Testing



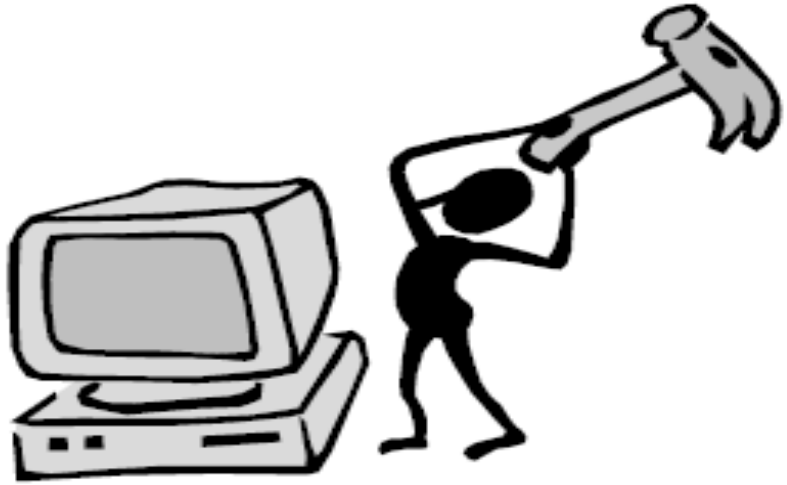
Development Vs Testing



When to Start / Stop Testing?

Who Should Test?

- Developer
  - Understands the system
  - But, will test gently
  - And, is driven by deadlines



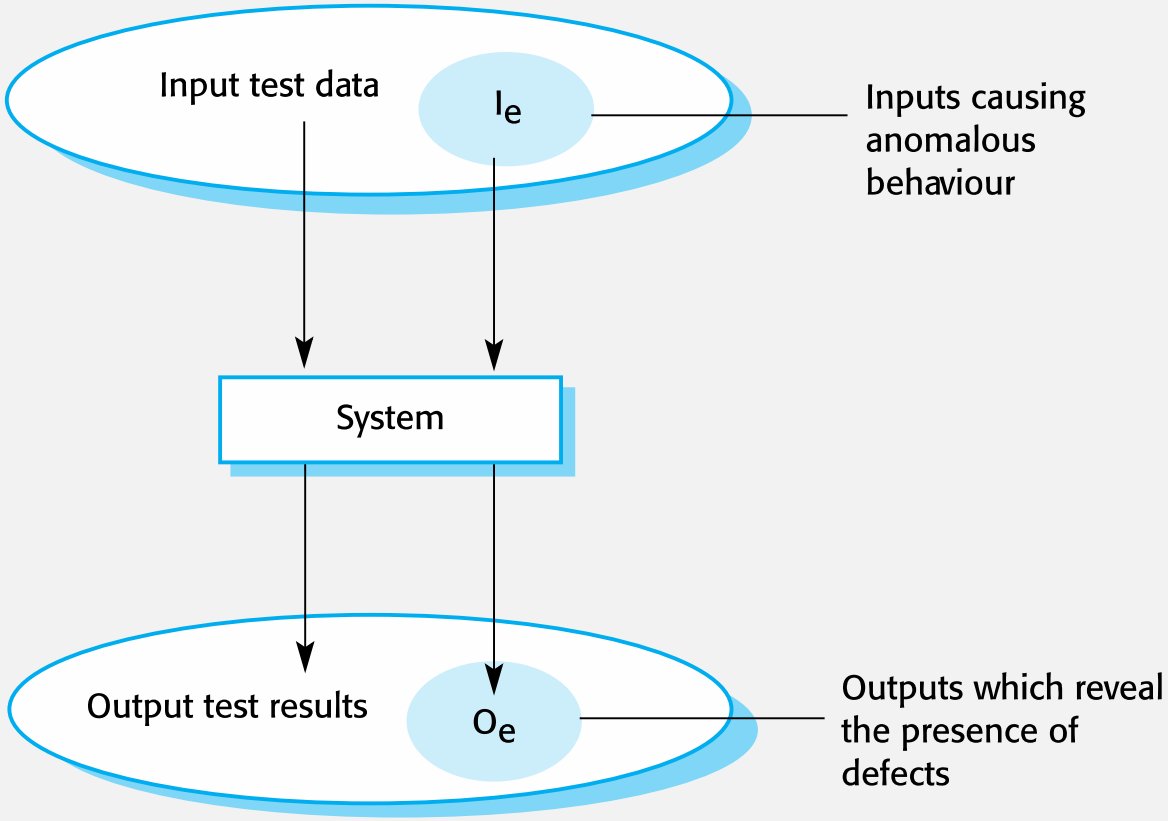
- Independent tester
  - Must learn system
  - But, will attempt to break it
  - And, is driven by "quality"

The role of the independent tester is to remove the conflict of interest inherent when the builder is testing his or her own product.

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.

An Input-output Model of Program Testing



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.

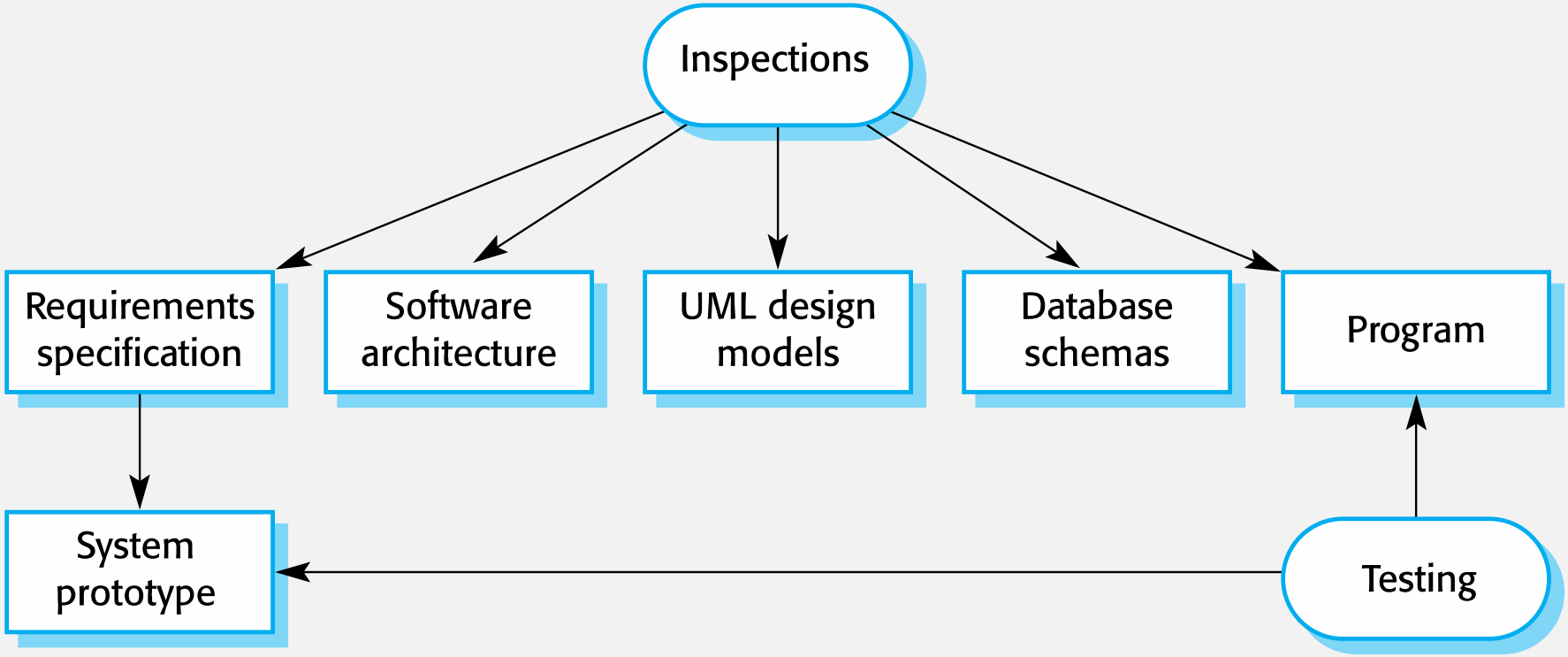
V & V Confidence

- 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 supplement 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



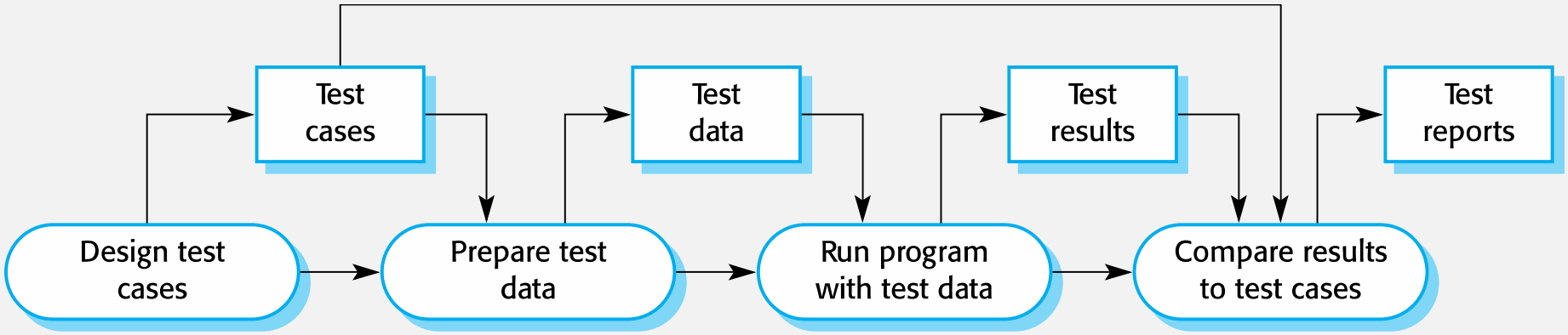
Software Inspections

- These involve people examining the source representation with the aim of discovering anomalies and defects.
- Inspections 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 have been shown to be an effective technique for discovering program errors.

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 own 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.

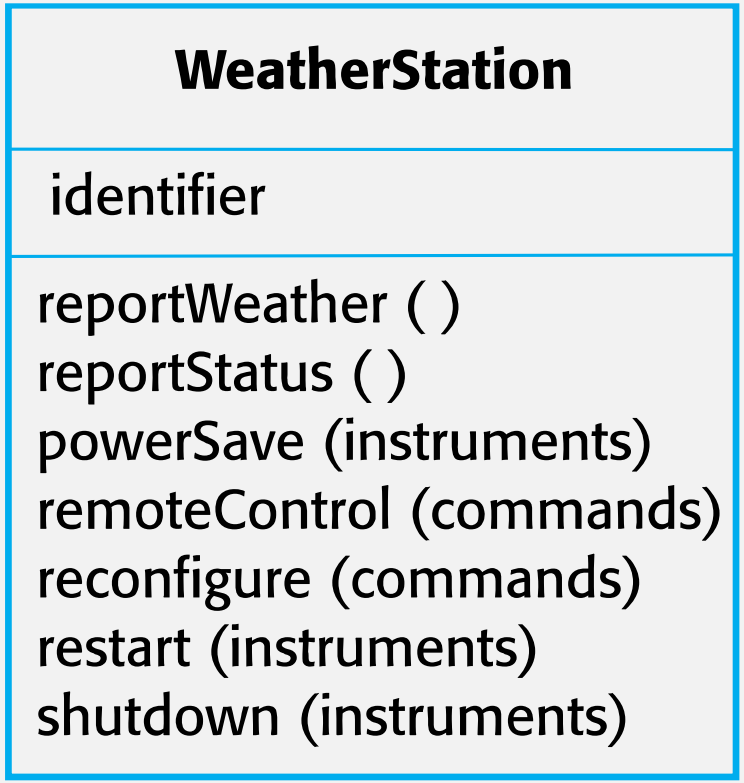
Unit Testing

- 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



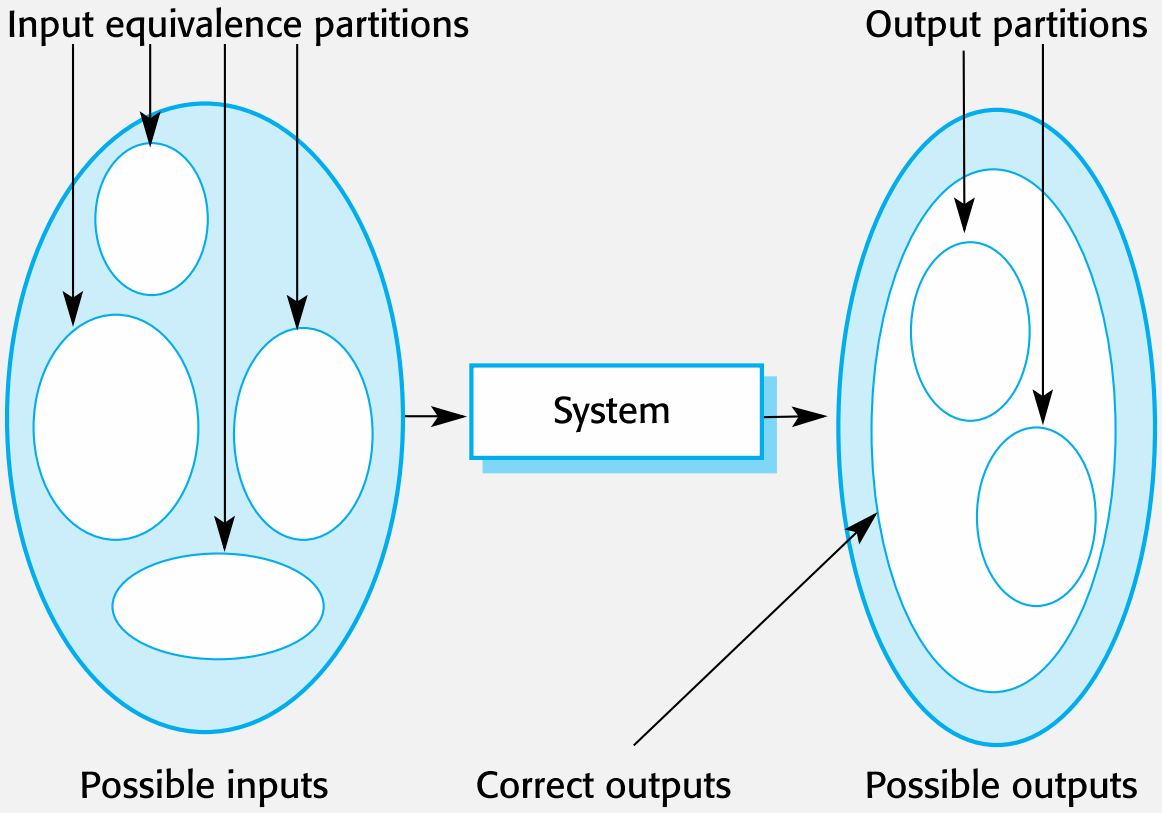
Testing Strategies

- Partition testing, where you identify groups of inputs that have common characteristics and should be processed in the same way.
  - You should choose tests from within each of these groups.
- Guideline-based testing, where you use testing guidelines to choose test cases.
  - These guidelines reflect previous experience of the kinds of errors that programmers often make when developing components.

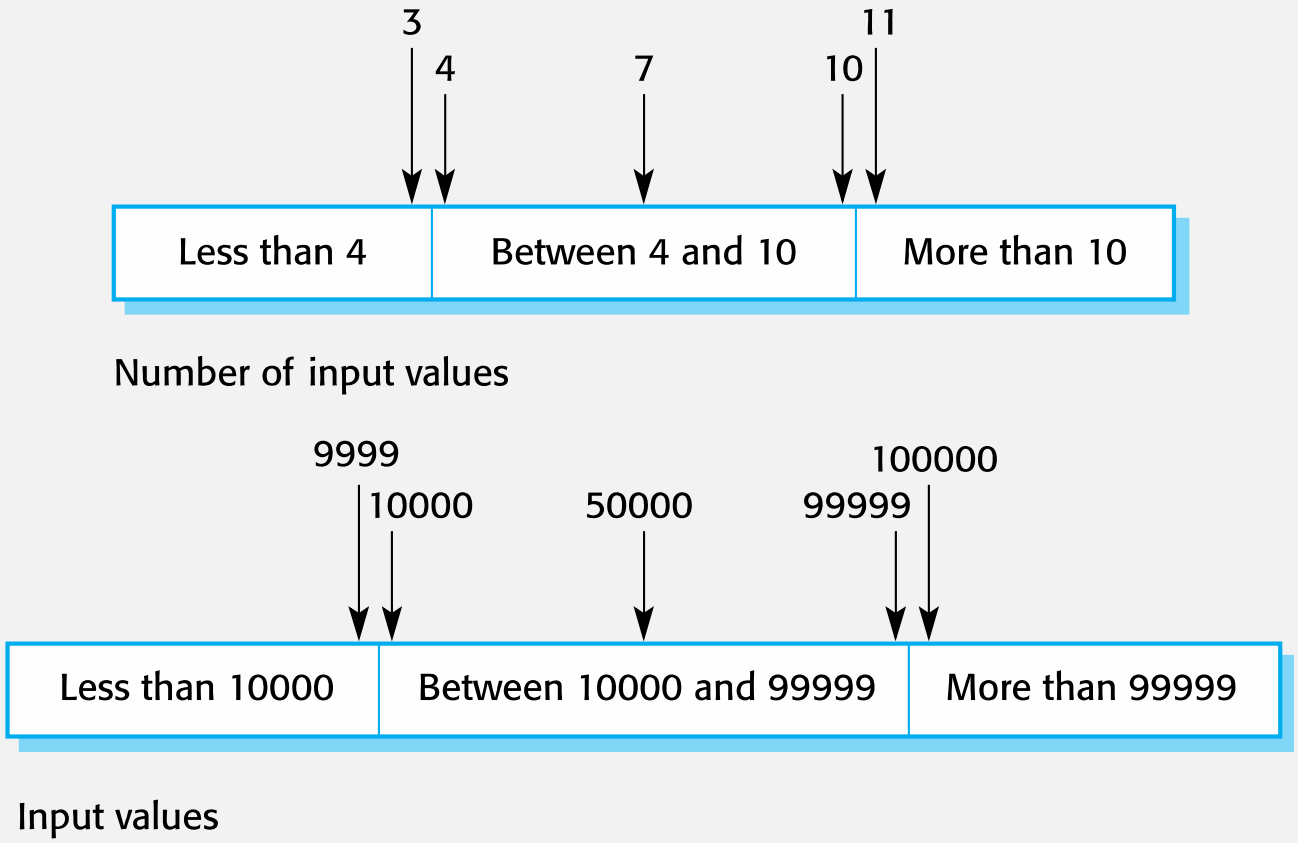
Partition Testing

- Input data and output results often fall into different classes where all members of a class are related.
- Each of these classes is an equivalence partition or domain where the program behaves in an equivalent way for each class member.
- Test cases should be chosen from each partition.

Equivalence Partitioning



Equivalence Partitions



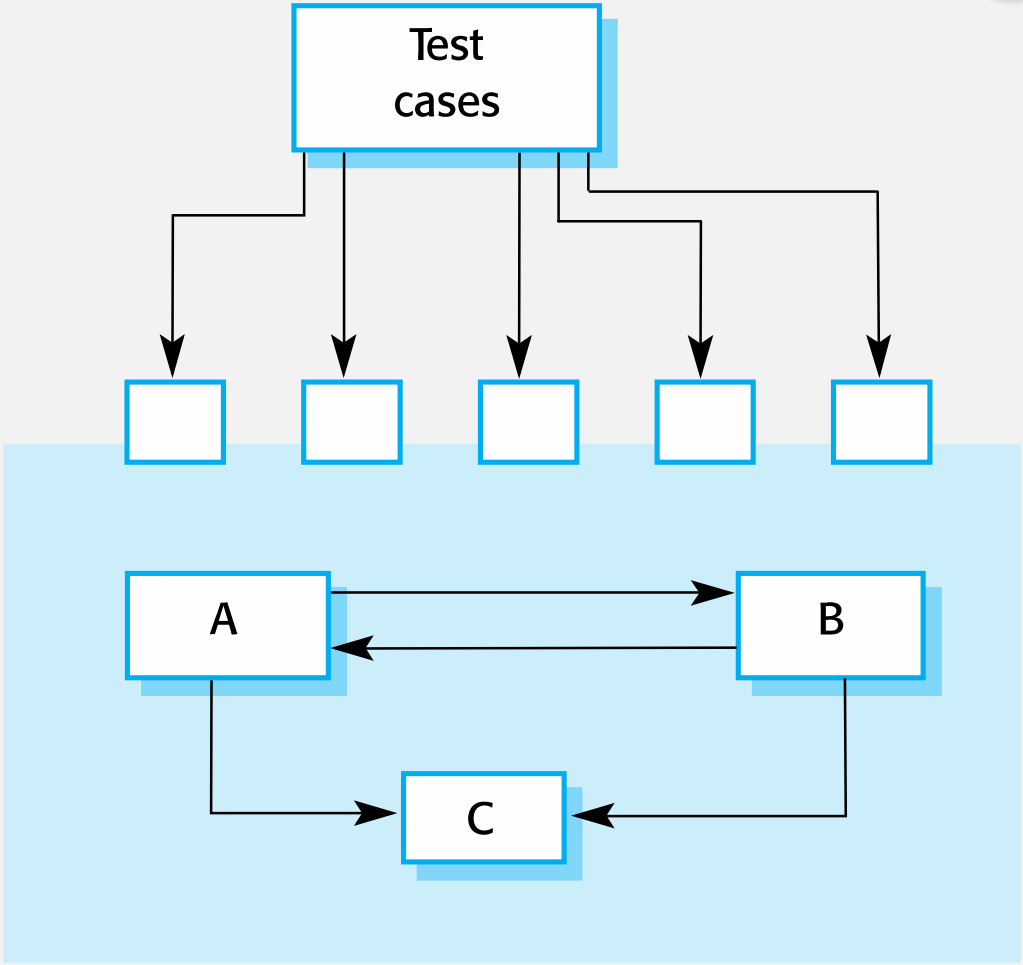
General Guidelines Based Testing

- 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.

Component Testing

- Software components are often composite components that are made up of several interacting objects. .
- 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.

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.

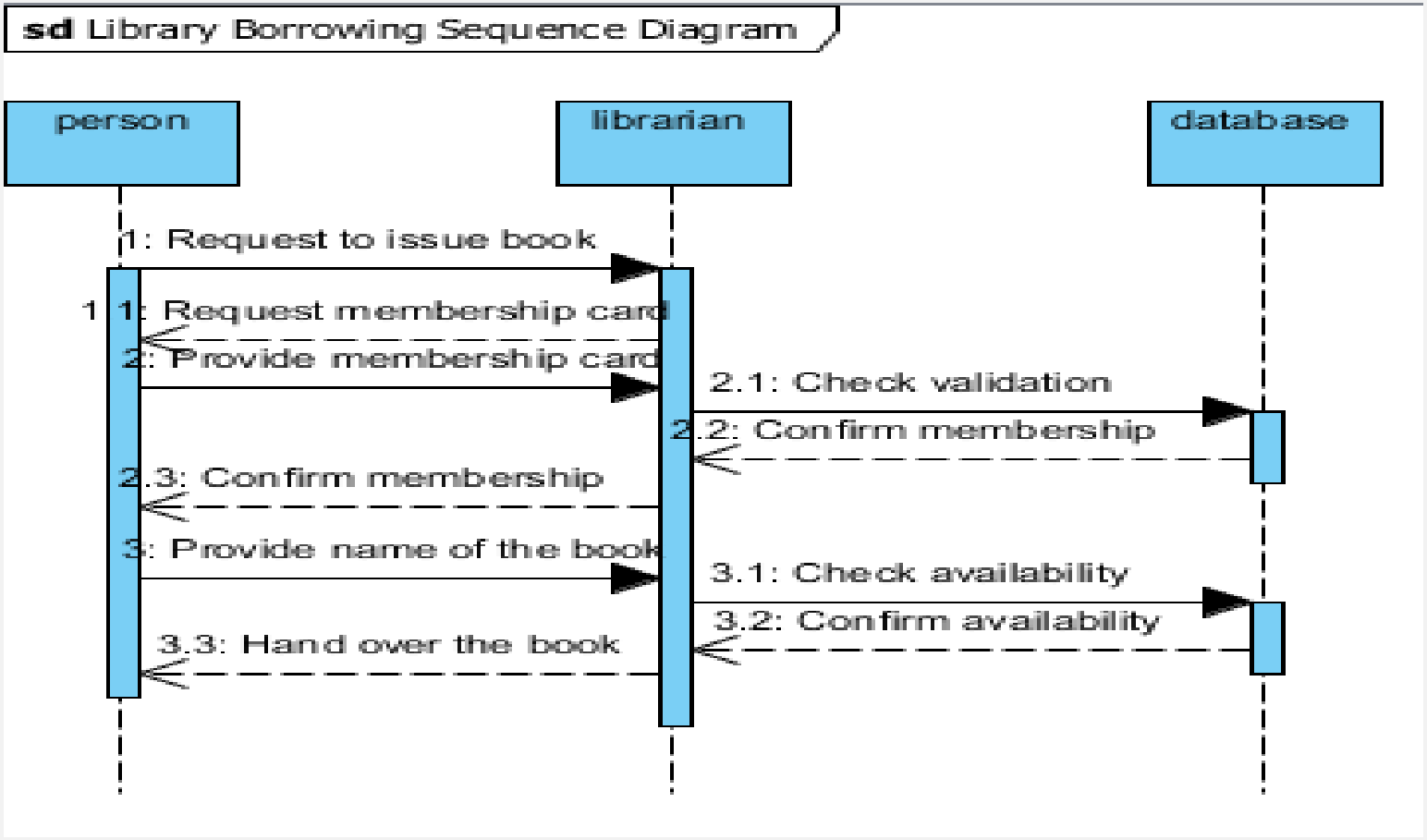
System Testing

- 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.

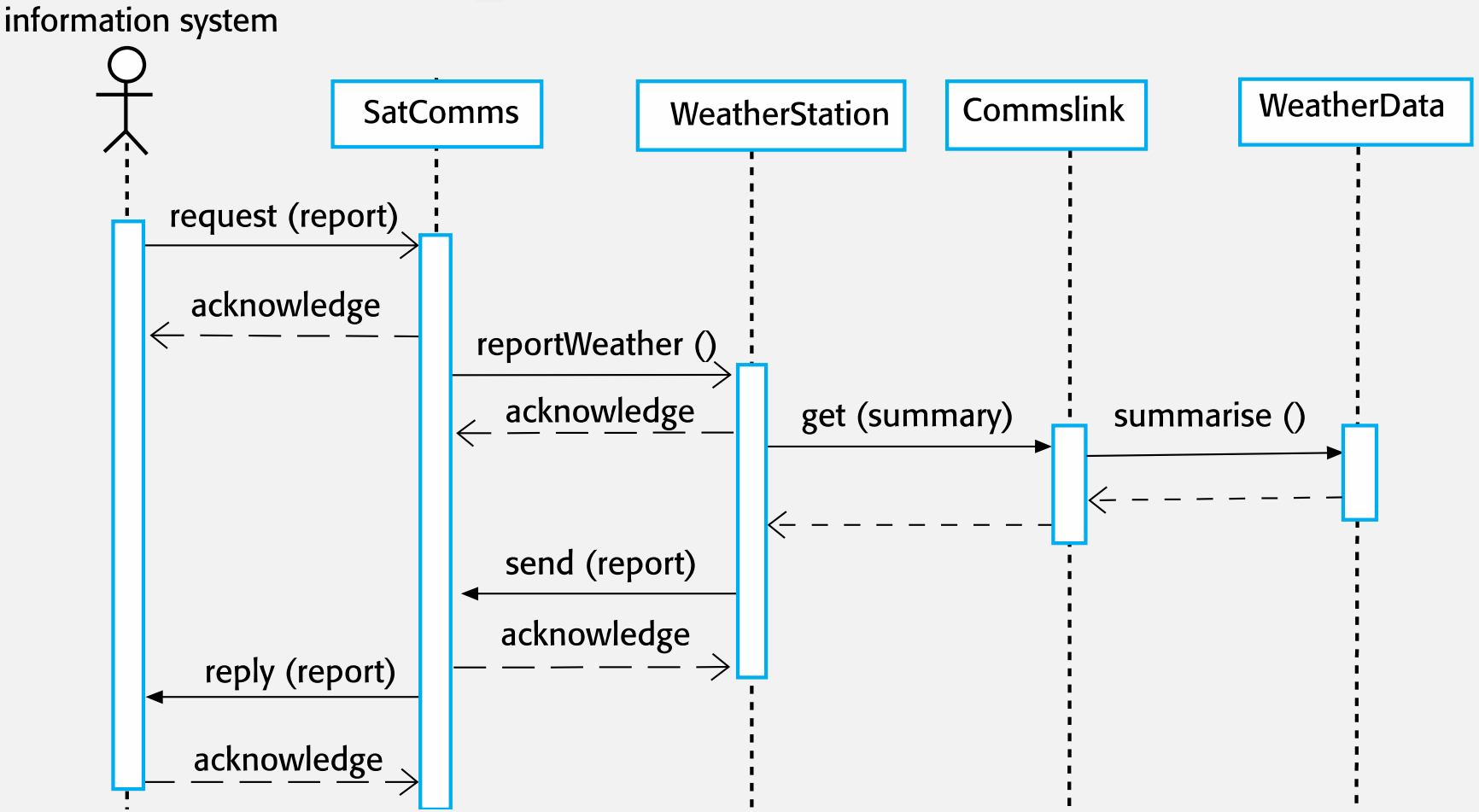
Use-case Testing

- 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.

Sequence Diagram



Collect Weather Data Sequence Chart



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.

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.

Path Testing

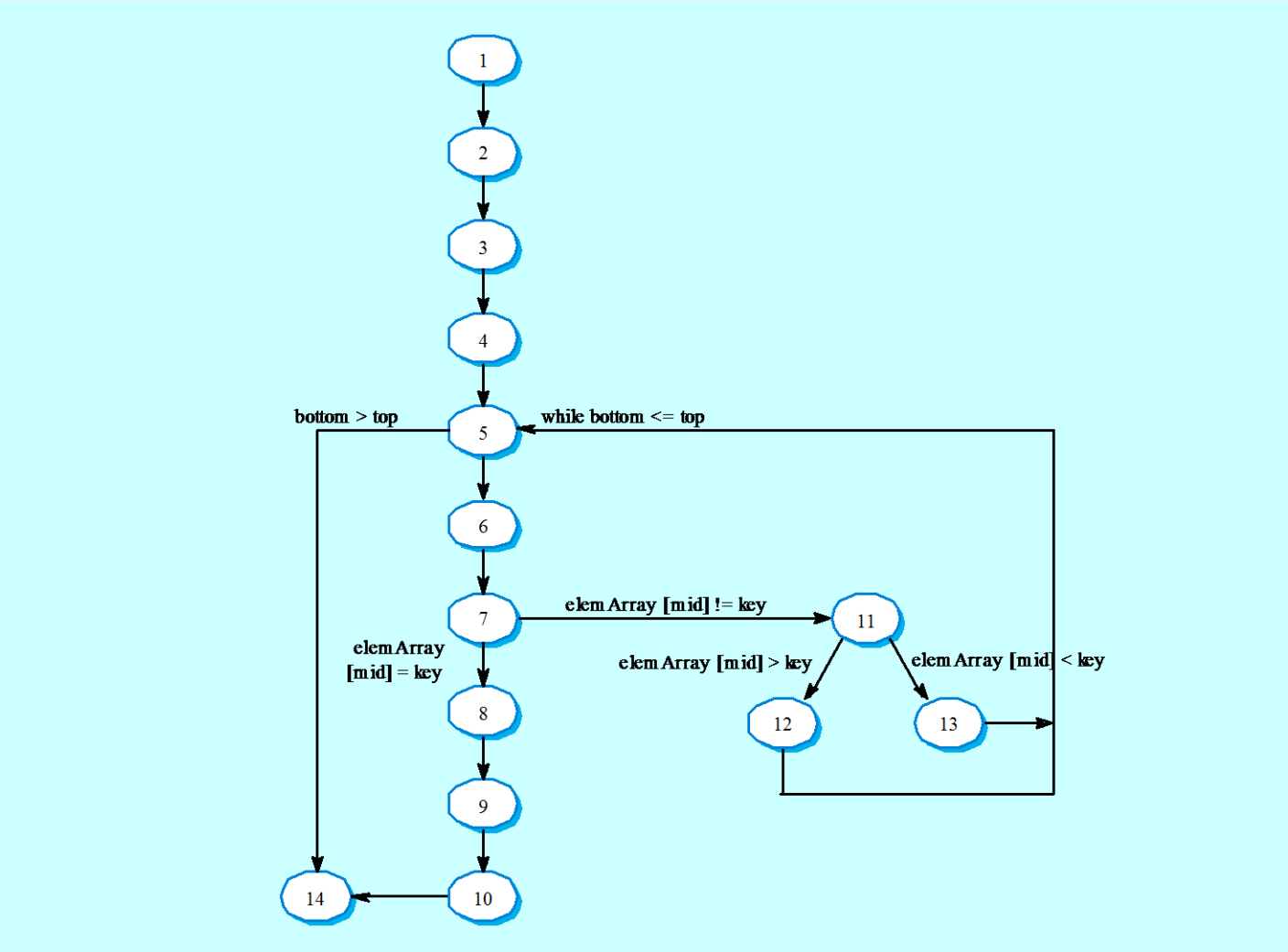
- The objective of path testing is to ensure that the set of test cases is such that each path through the program is executed at least once.
- The starting point for path testing is a program flow graph that shows nodes representing program decisions and arcs representing the flow of control.
- Statements with conditions are therefore nodes in the flow graph.

Steps for Basis Path testing

- The basic steps involved in basis path testing include
- Draw a control graph (to determine different program paths)
- Find a basis set of paths
- Generate test cases to exercise each path

```
public static void search ( int key, int 0elemArray, Result r )
{
1. int bottom =0 ;
2. int top =elemArray.length - 1 ; int mid;
3. r.found =false ;
4. r.index =-1 ;
5. while ( bottom <= top )
{
6 mid =(top + bottom) / 2 ;
7 if (elemArray [mid] = key)
{
8 rindex = mid;
9 r.found =true ;
10 return ; } // if part
else
{
11 if (elemArray [mid] < key)
12 bottom = mid + 1 ;
else
13 top = mid - 1 ; }
} //while loop
14. } //search
```

Binary Search Flow Graph



Independent Paths

- 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 14
- 1, 2, 3, 4, 5, 14
- 1, 2, 3, 4, 5, 6, 7, 11, 12, 5, …
- 1, 2, 3, 4, 5, 6, 7, 11, 13, 5, …
- Test cases should be derived so that all of these paths are executed
- A dynamic program analyser may be used to check that paths have been executed

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.

Requirements Tests

- 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.
- Set up a patient record in which allergies to two or more drugs are recorded. Prescribe both of these drugs separately and check that the correct warning for each drug is issued.
- Prescribe a drug that issues a warning and overrule that warning. Check that the system requires the user to provide information explaining why the warning was overruled.

A Usage Scenario for the Mentcare System

George is a nurse who specializes in mental healthcare. One of his responsibilities is to visit patients at home to check that their treatment is effective and that they are not suffering from medication side effects.

On a day for home visits, George logs into the Mentcare system and uses it to print his schedule of home visits for that day, along with summary information about the patients to be visited. He requests that the records for these patients be downloaded to his laptop. He is prompted for his key phrase to encrypt the records on the laptop.

One of the patients that he visits is Jim, who is being treated with medication for depression. Jim feels that the medication is helping him but believes that it has the side effect of keeping him awake at night. George looks up Jim's record and is prompted for his key phrase to decrypt the record. He checks the drug prescribed and queries its side effects. Sleeplessness is a known side effect so he notes the problem in Jim's record and suggests that he visits the clinic to have his medication changed. Jim agrees so George enters a prompt to call him when he gets back to the clinic to make an appointment with a physician. George ends the consultation and the system re-encrypts Jim's record.

After, finishing his consultations, George returns to the clinic and uploads the records of patients visited to the database. The system generates a call list for George of those patients who He has to contact for follow-up information and make clinic appointments.

Features Tested by Scenario

- Authentication by logging on to the system.
- Downloading and uploading of specified patient records to a laptop.
- Home visit scheduling.
- Encryption and decryption of patient records on a mobile device.
- Record retrieval and modification.
- Links with the drugs database that maintains side-effect information.
- The system for call prompting.

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.

Load Testing

- It is a process of testing the behavior of a software by applying maximum load in terms of software accessing and manipulating large input data. It can be done at both normal and peak load conditions. This type of testing identifies the maximum capacity of software and its behavior at peak time.

Stress Testing

- Stress testing includes testing the behavior of a software under abnormal conditions. For example, it may include taking away some resources or applying a load beyond the actual load limit.
- The aim of stress testing is to identify the breaking point.
- It tests the failure behaviour of the system.
- This testing can be performed by testing different scenarios such as:
  - Shutdown or restart of network ports randomly
  - Turning the database on or off
  - Running different processes that consume resources such as CPU, memory, server, etc.

User testing

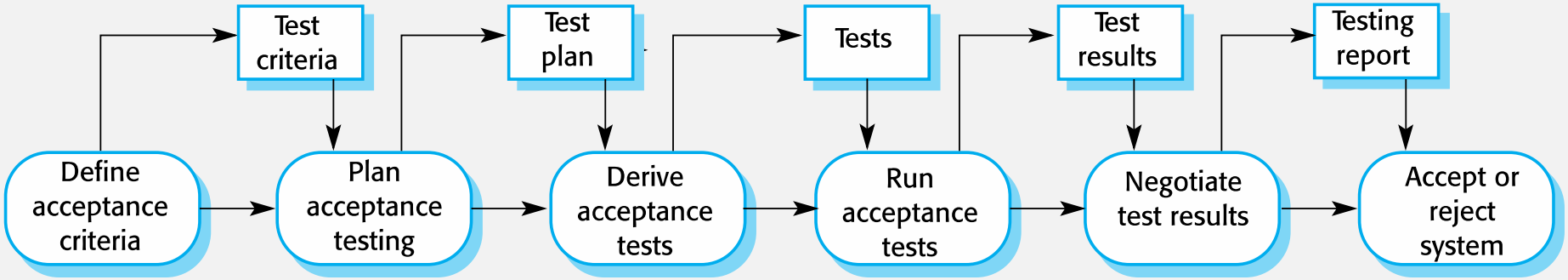
User Testing

- 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.

The Acceptance Testing Process



Stages in the Acceptance Testing Process

- Define acceptance criteria
- Plan acceptance testing
- Derive acceptance tests
- Run acceptance tests
- Negotiate test results
- Reject/accept system

Agile methods and acceptance testing

- In agile methods, the user/customer is part of the development team and is responsible for making decisions on the acceptability of the system.
- Tests are defined by the user/customer and are integrated with other tests in that they are run automatically when changes are made.
- There is no separate acceptance testing process.
- Main problem here is whether or not the embedded user is 'typical' and can represent the interests of all system stakeholders.

That is all