Chapter 8 – Software Testing

Topics covered

- ✓ **Development testing**
  - √ **Unite**
  - √ **Component**
  - System
- ✧ Test-driven development
- ✧ Release testing
- ✧ User 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.

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.

wilderness weather station example

- In the wilderness weather station example, the system software reports summarized weather data to a remote computer as described in Figure 7.3.

System	Weather station
Use case	Report weather
Actors	Weather information system, Weather station
Data	The weather station sends a summary of the weather data that has been collected from the instruments in the collection period to the weather information system. The data sent are the maximum, minimum, and average ground and air temperatures; the maximum, minimum, and average air pressures; the maximum, minimum and average wind speeds; the total rainfall; and the wind direction as sampled at 5-minute intervals.
Stimulus	The weather information system establishes a satellite communication link with the weather station and requests transmission of the data.
Response	The summarized data is sent to the weather information system.
Comments	Weather stations are usually asked to report once per hour, but this frequency may differ from one station to another and may be modified in future.

Figure 7.3 Use case description—Report weather

Figure 8.8 shows the **sequence of operations** in the weather station when it responds to a request to collect data for the mapping system.

You can use this diagram to **identify operations that will be tested and to help design the test cases to execute the tests.**

Collect weather data sequence chart

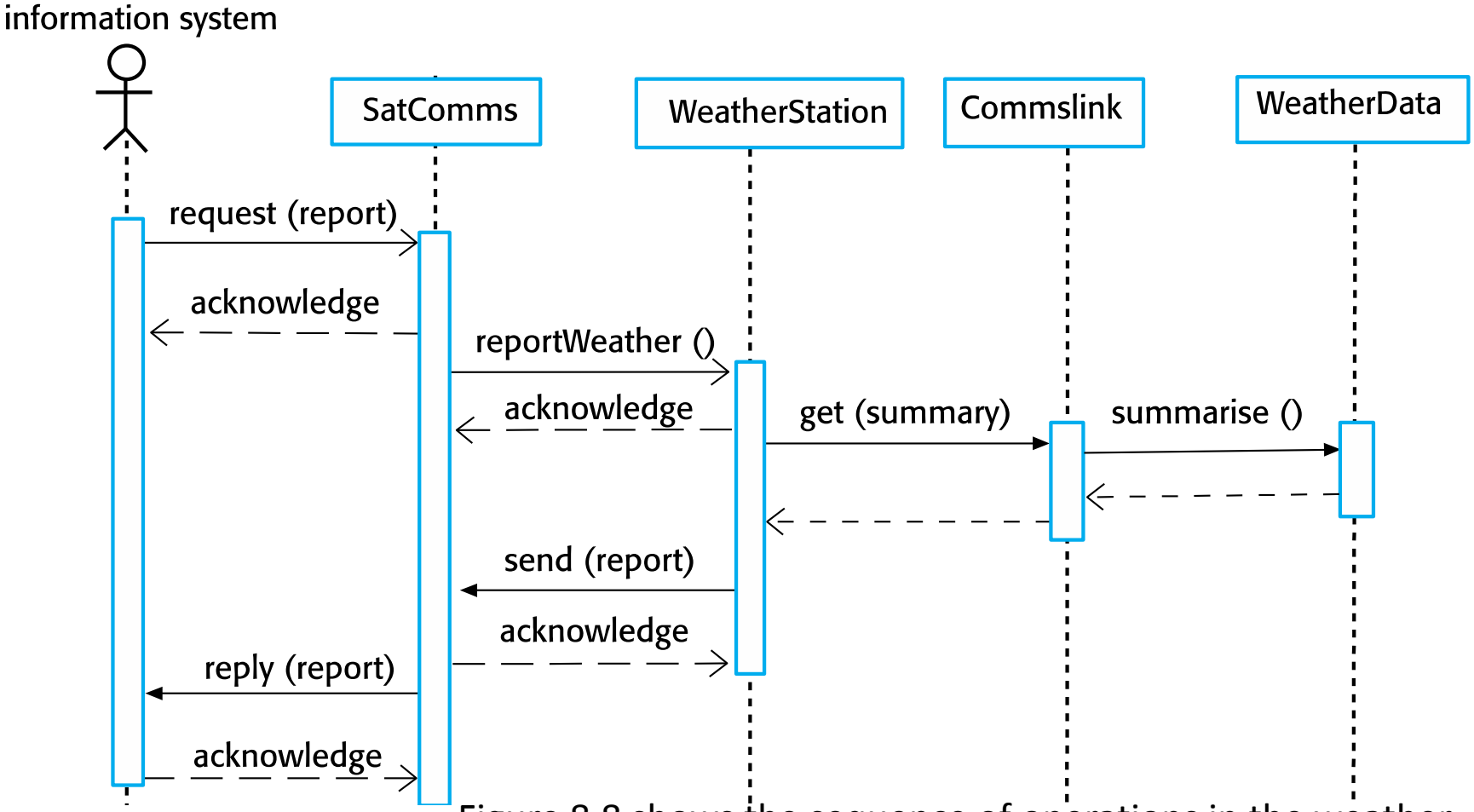


Figure 8.8 shows the sequence of operations in the weather station when it responds to a request to collect data for the mapping system.

Therefore issuing a **request for a report** will result in the execution of the following thread of methods:

SatComms:request → WeatherStation:reportWeather → Commslink:Get(summary) → WeatherData:summarize

Test cases derived from sequence diagram

The sequence diagram helps you design the specific test cases that you need, as it shows what inputs are required and what outputs are created:

1. An input of a request for a report should have an associated acknowledgment. A report should ultimately be returned from the request. During testing, you should create summarized data that can be used to check that the report is correctly organized.

2. An input request for a report to **WeatherStation** results in a summarized report being generated. You can test this in isolation 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

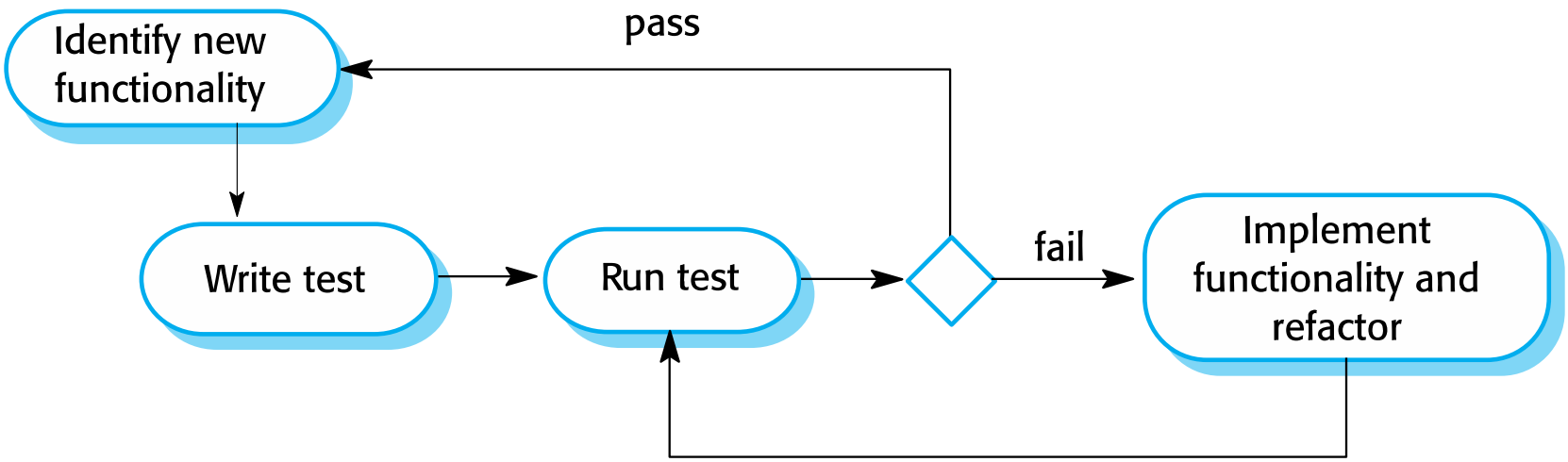
- Exhaustive system testing **(where every possible program execution sequence is tested)** is impossible so testing policies which define the required system **test coverage** may be developed.
- Examples of testing policies:
  - All system functions that are accessed through menus should be tested.
  - 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.

Test-driven development

Test-driven development

- Test-driven development (TDD) is an approach to program development in **which you inter-leave testing and code development.**
- Tests are written before code **and '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.

Test-driven development



TDD process activities

- Start by identifying the increment of functionality that is required. **This should normally be small and implementable in a few lines of code.**
- Write a test for this functionality and implement this as an automated test.
- Run the test, along with all other tests that have been implemented. Initially, you have not implemented the functionality so the new test will fail.
- Implement the functionality and re-run the test.
- Once all tests run successfully, you move on to implementing the next chunk of functionality.

Benefits of test-driven development

- Code coverage
  - Every code segment that you write has at least one associated test so all code written has at least one test.
- Regression testing
  - A regression test suite is developed incrementally as a program is developed.
- Simplified debugging
  - When a test fails, it should be obvious where the problem lies. The newly written code needs to be checked and modified.
- System documentation
  - The tests themselves are a form of documentation that describe what the code should be doing.

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**).

1- 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 two drugs that the patient is allergic to. **Check** that two warnings are correctly 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.

2- Scenario testing

Scenario testing is an approach to release testing whereby you devise typical scenarios of use and use these scenarios to **develop test cases** for the system.

A scenario is **a story that describes one way** in which the system might be used.

Scenarios **should be realistic**, and real system users should be able to relate to them.

If you have used scenarios or user stories as part of the **requirements engineering process**, then you may be able to reuse them as testing scenarios.

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.

This scenario tests a number of features of the Mentcare system:

- ✧ 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.

3- 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

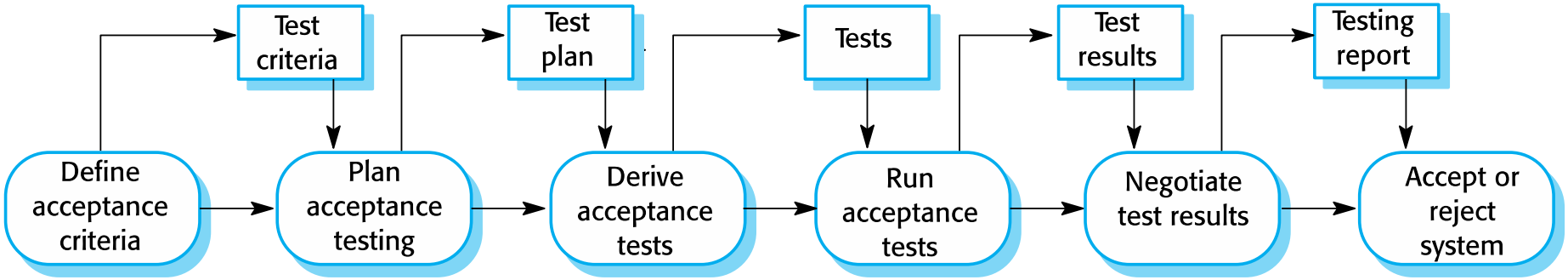
User testing

- 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.**

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.**

EXERCISES

The Software Engineering Competence Center (**SECC**) is an Egyptian leading ICT organization
aiming at bridging the gap between the technologies needed to overcome the economical-social-environmental challenges and the current existing technologies.

https://www.secc.org.eg/main.asp

https://junit.org