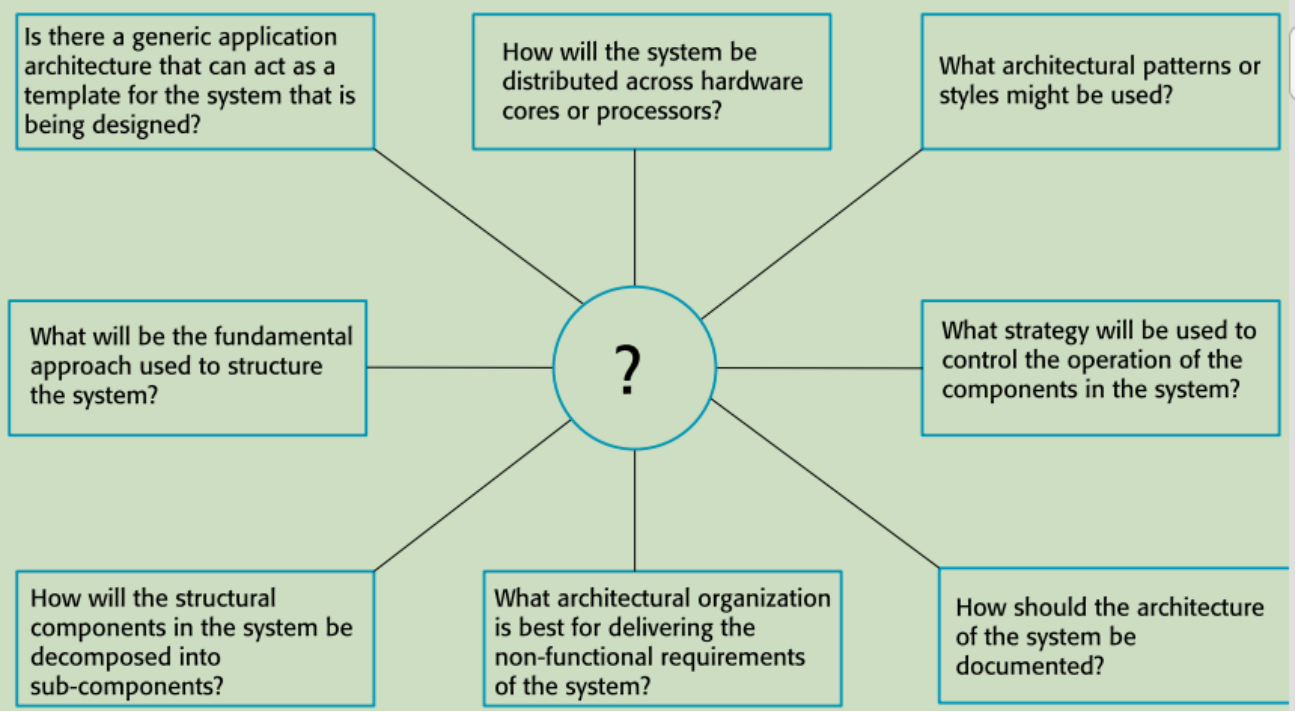
L7 Architectural Design

- Architectura designs is concerned with understanding how a software system should be organized and designing the overall structure of that system. It is the critical link between design and requirements engineering.
- Architecutre in the small is concerned with the architecture of individual programs. We concerned with the way that an individual program is decomposed into componets.
- Architecture in the large is concerned with the architecture of complex enterprise systems that include other systems, programs and program components.

Architectural design decisions



Architecutre and system characteristics

- Performance; Security; Safety; Availability; Maintainability

Architectural views

- Each architecutral model only shows one view or perspective of the system.

4+1 view model of software architecture

- **Logical view**
  - *It shows the key abstractions in the system as objects or object classes*
- **Process view**

- *it shows hot, at run-time, the system is composed of interacting* processes

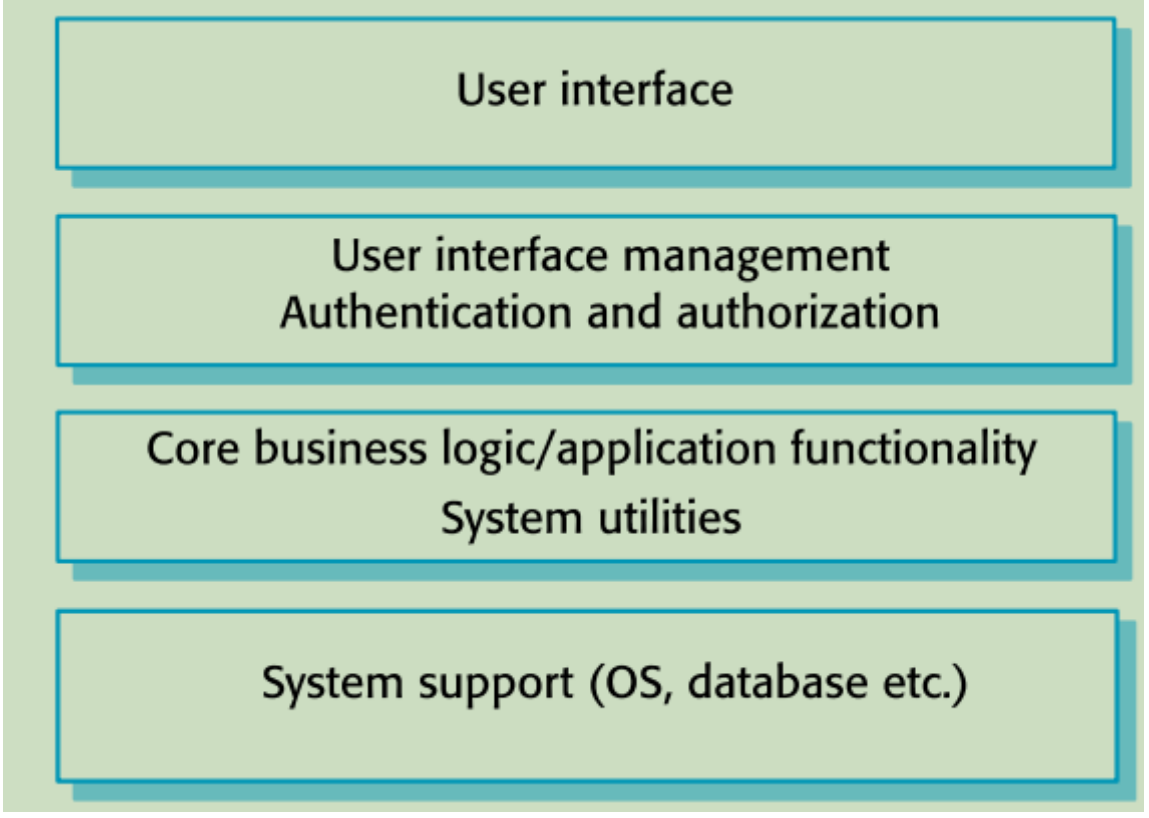
- **Development view**
  - *it shows how the software is decomposed for development*

- **Physical view**
  - *it shows the system hardware and how software components are distributed across the processors in the system*

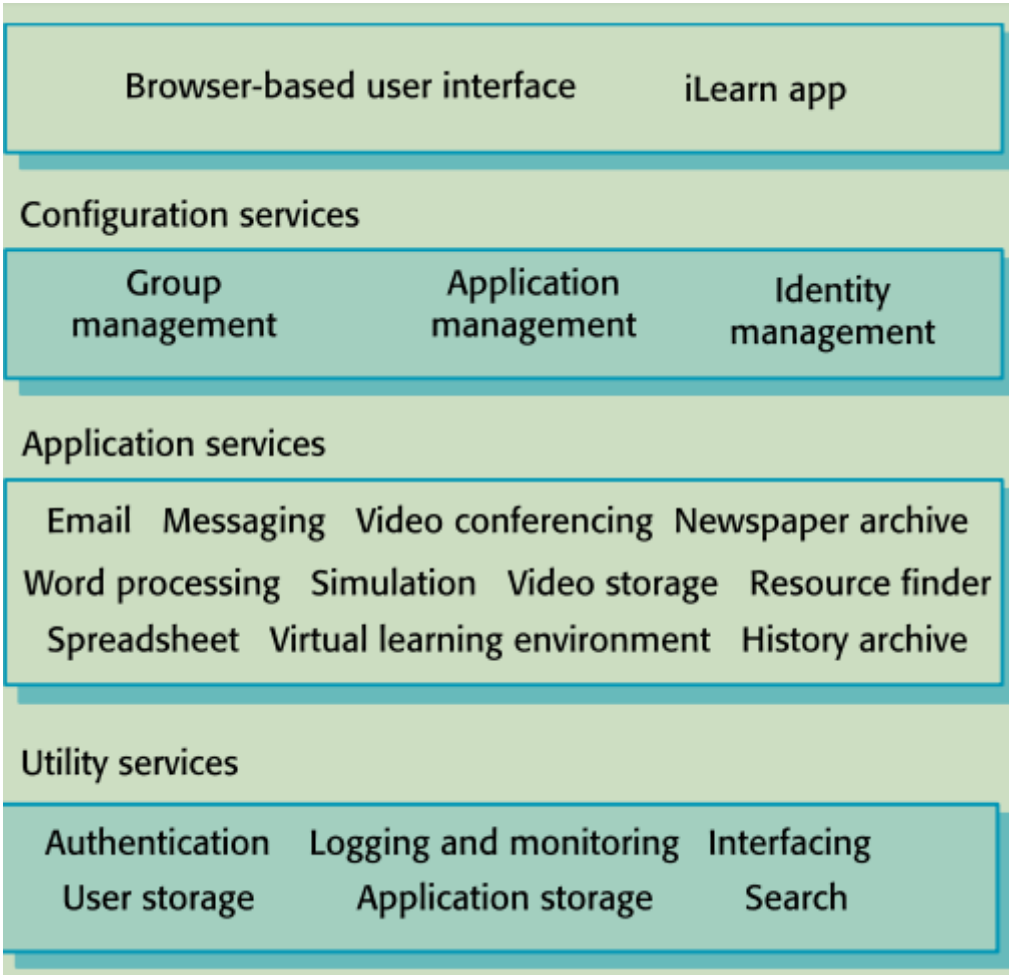
- **Related using use cases or scenarios (+1)**

Architectural patterns

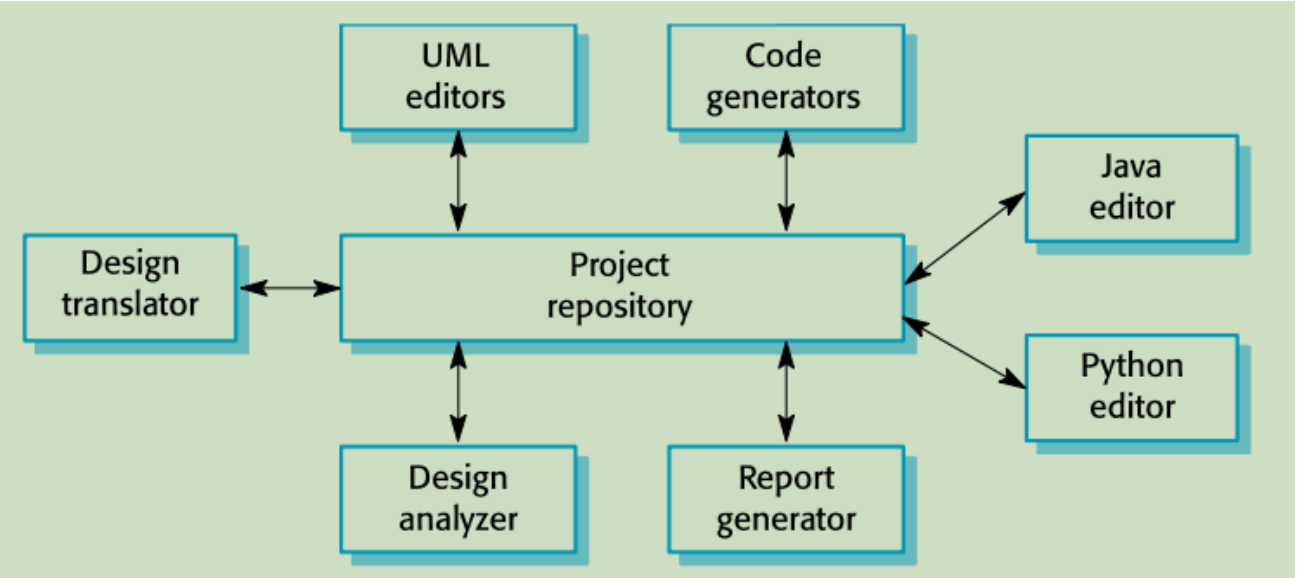
- An architectural pattern is a stylized description of good design practice, which has been tried and tested in different environments
- A generic layered architecture



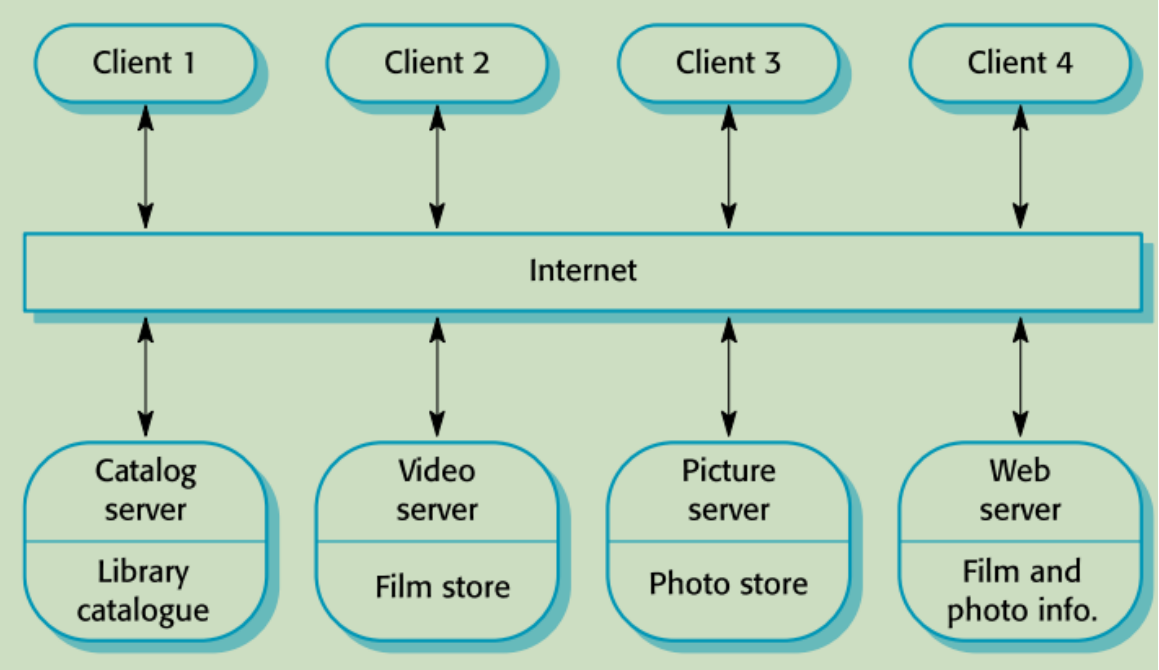
  - e.g. iLearn system



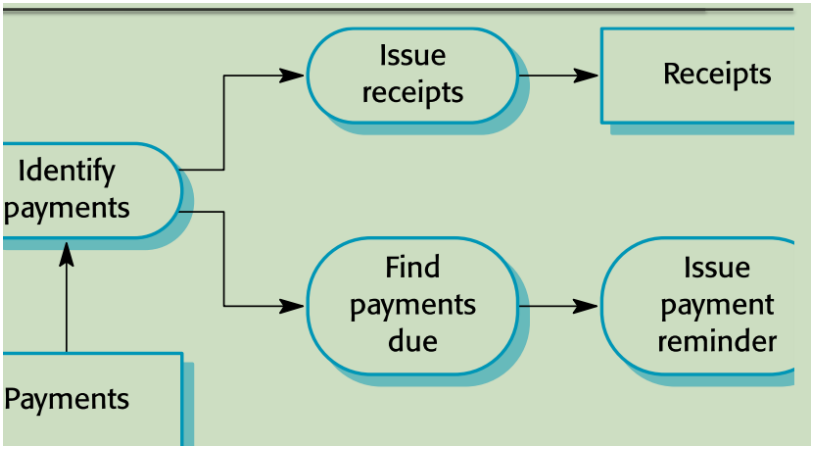
- Repository architecutre
  - Sub-systems must exchange data. This may be done in two ways:
    - shared data is held in central database or repository and may be accessed by all sub-systems
    - each sub-system maintains its own database and passed data explicitly to other sub-system
  - when large amounts of data are to be shared, the repository model of sharing is most commonly used.
  - e.g. IDE



- Client-server architecutre
  - Distributed system model which shows how data and processing is distributed across a range of components
  - e.g. film library

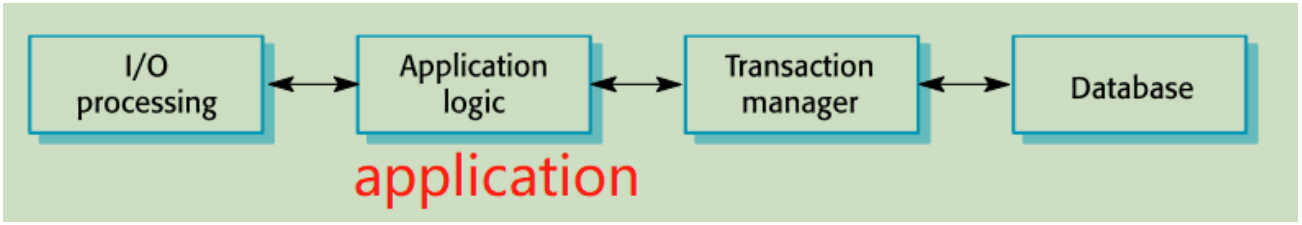


- Pipe and filter architecutre
  - Functional transformations process their inputs to produce outputs
  - e.g. payment system

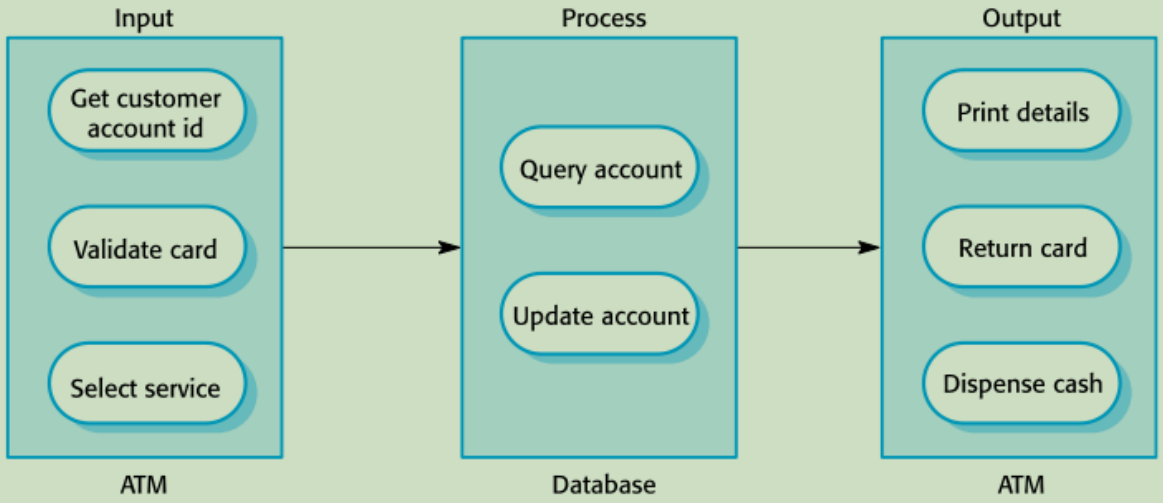


Application architectures

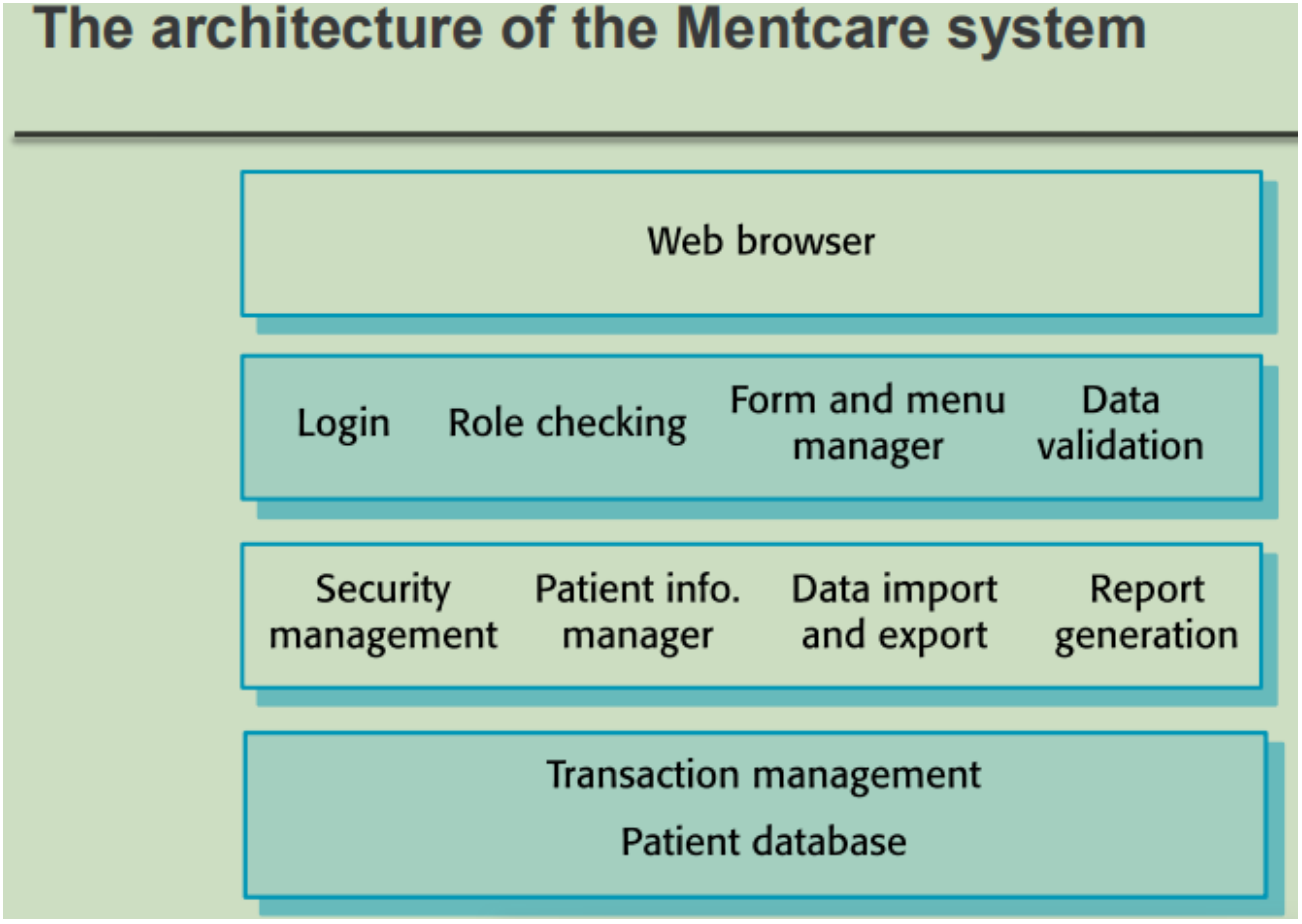
- Application systems are designed to meet an organisational need.
- A generic application architecture is an architecture for a type of software system that may be configured and adapted to create a system that meets specific requirements.
- Two very widely used generic application architectures are transaction processing systems and language processing systems
    - Transaction processing systems
        - Process user requests for information from a database or requests to update the database
        - e.g. E-commerce systems; Reservtion systems



The software architecture of an ATM system



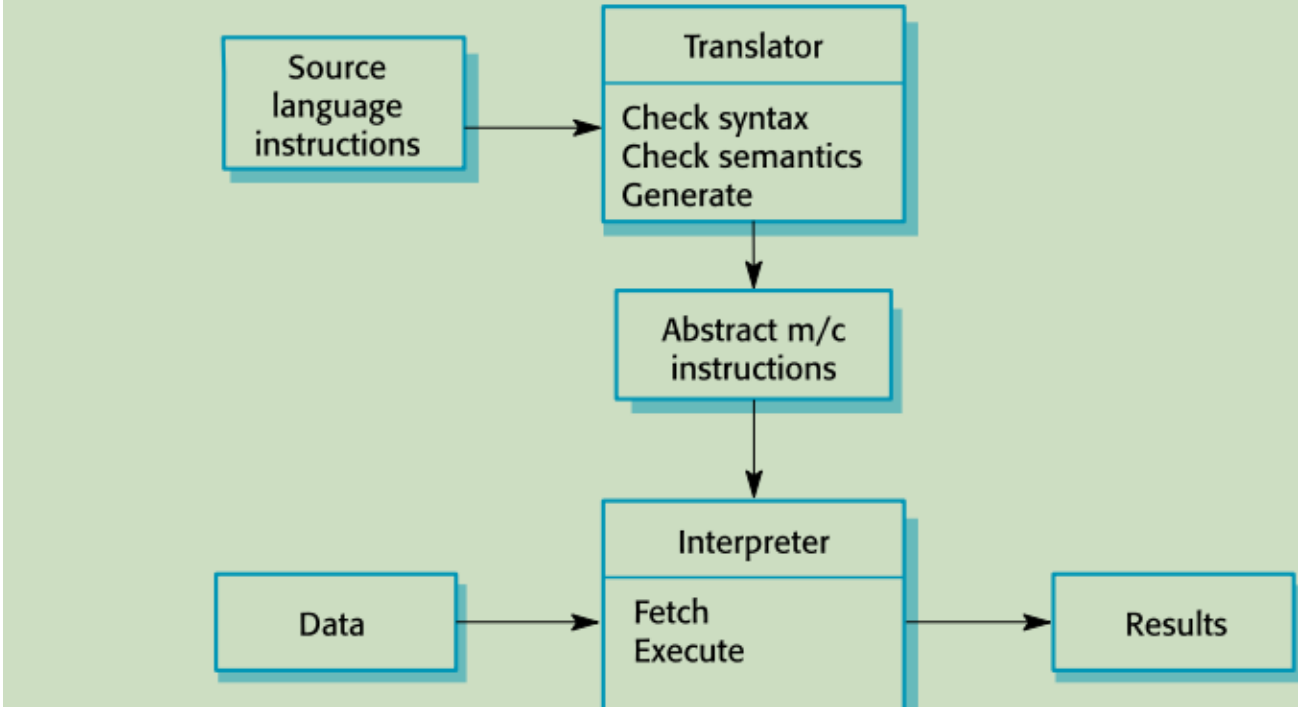
The architecture of the Mentcare system



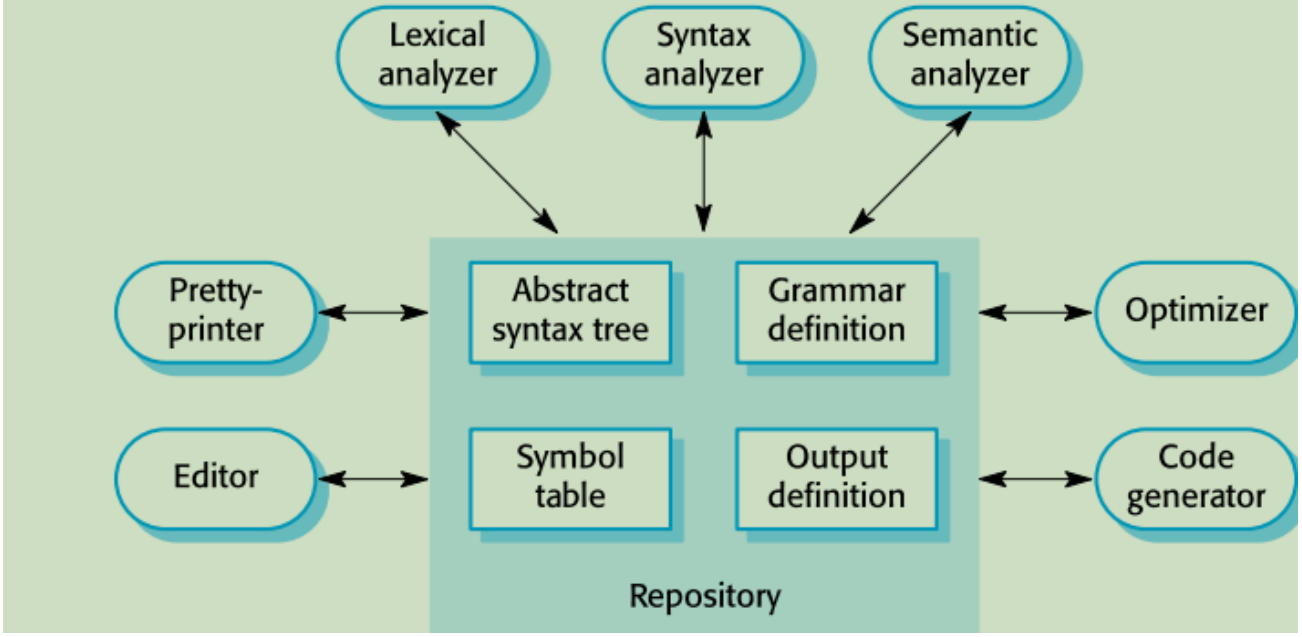
* language processing systems

```
    * Accept a natural or artifical language as input and generate
some other representation of that language

    * Used in situations where the easiest way to solve a problem is
to describe an algorithm or desribe the system data

    * e.g. Compilers, Command interpreters
```

The architecture of a language processing system



A repository architecture for a language processing system



Key Points

- a software architecture is a description of how a software system is organized

- architectural design decisions include decisions on the type of application, the distribution of the system, the architectural styles to be used
- architectural may be documented from several different perspectives or views such as conceptual view, a logical view, a process view, and development view
- architectural patterns are a means of reusing knowledge about generic system architecutres. They describe the architecture, explain when it may be used and describe its advantage and disadvantages
- transaction processing systems are interactive systems that allow information in a databae to be remotely accessed and modified by a number of users
- language processing systems are used to translate texts from one language into another and carry out the instructions specified in the input language. They include a translator and an abstract machine that executes the generated language

L8 : Design and Implementation

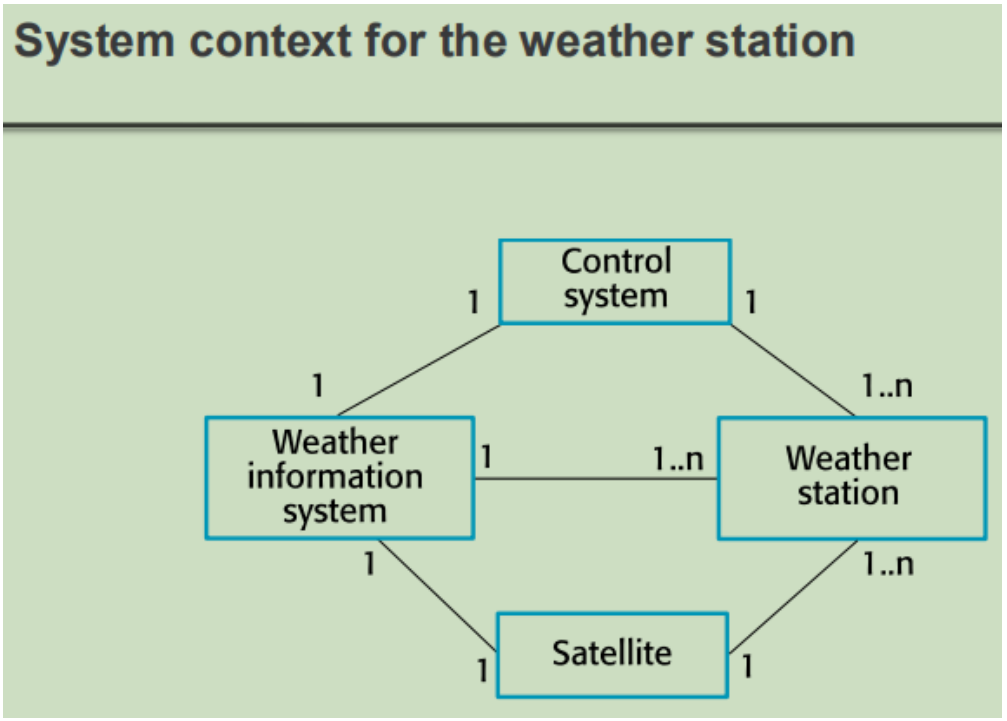
- Software design and implementation is the stage in the software engineering process at which an executable software system is develope
- Software design and implementation activities are invariably inter-leaved
    - software design is a creative activity in which you identify software components and their relationships, based on a customer's requirements
    - implementation is the process of realizing the design as a program

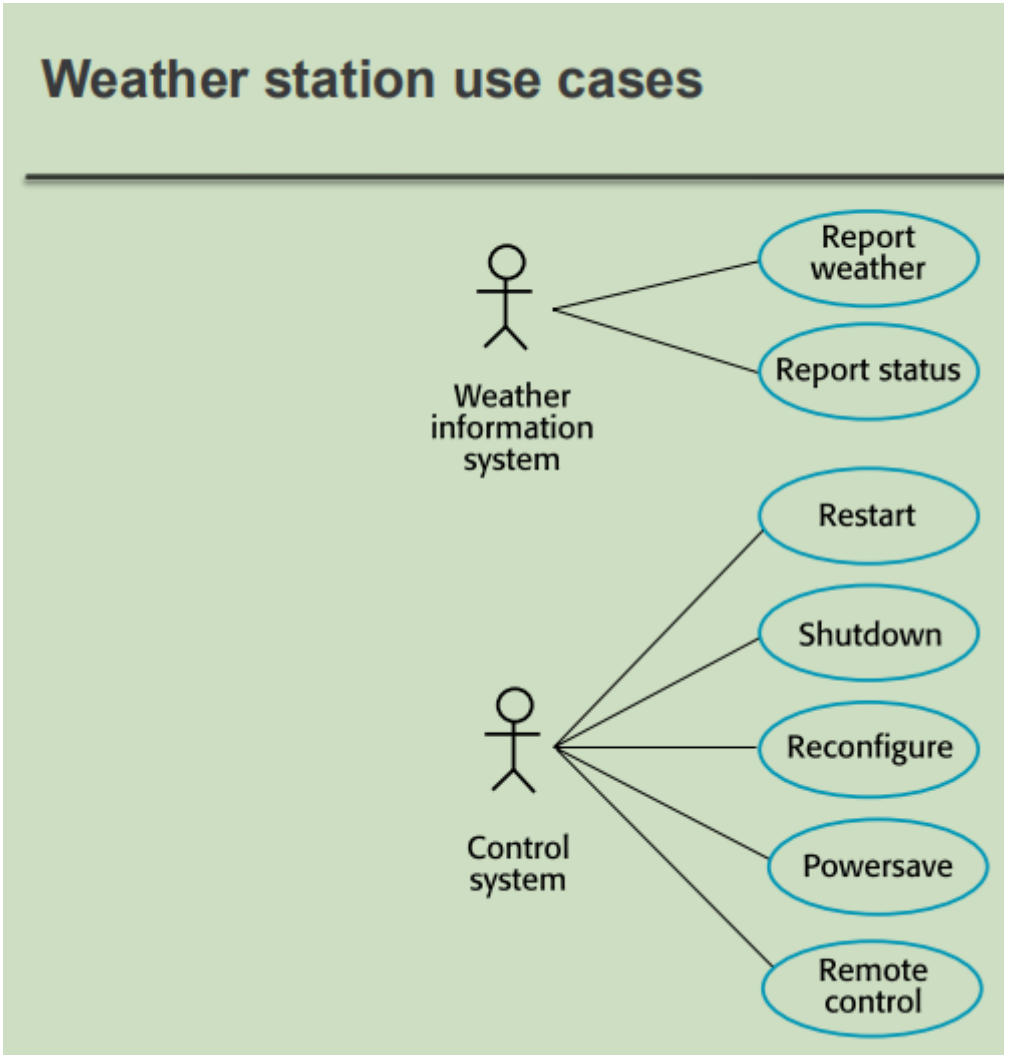
Object-oriented design using the UML

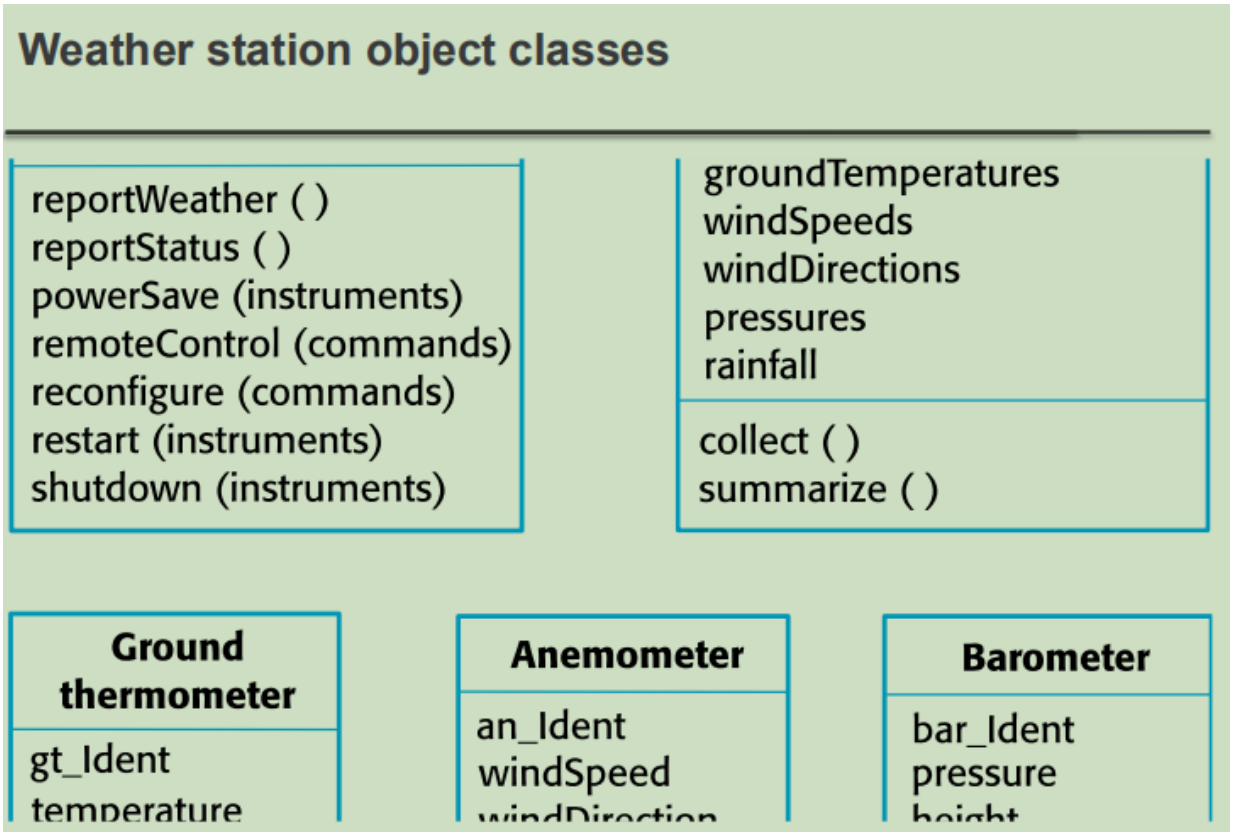
- Structured object-oriented design process involve developing a number of different system models.
- They require a lot of effort for development and maintenance of these models and, for small systems, this may not be cost-effective
- Common activities in these processes include:
    - define the context and models of use of the system
    - design the system architecture
    - identify the principal system objects
    - develop design models
    - specify object interfaces

Context and models

- A system context model is a structural model that demonstrates the other systems in the environment of the system being developed.
- An interaction model is a dynamic model that shows how the system interacts with its environment as it is used

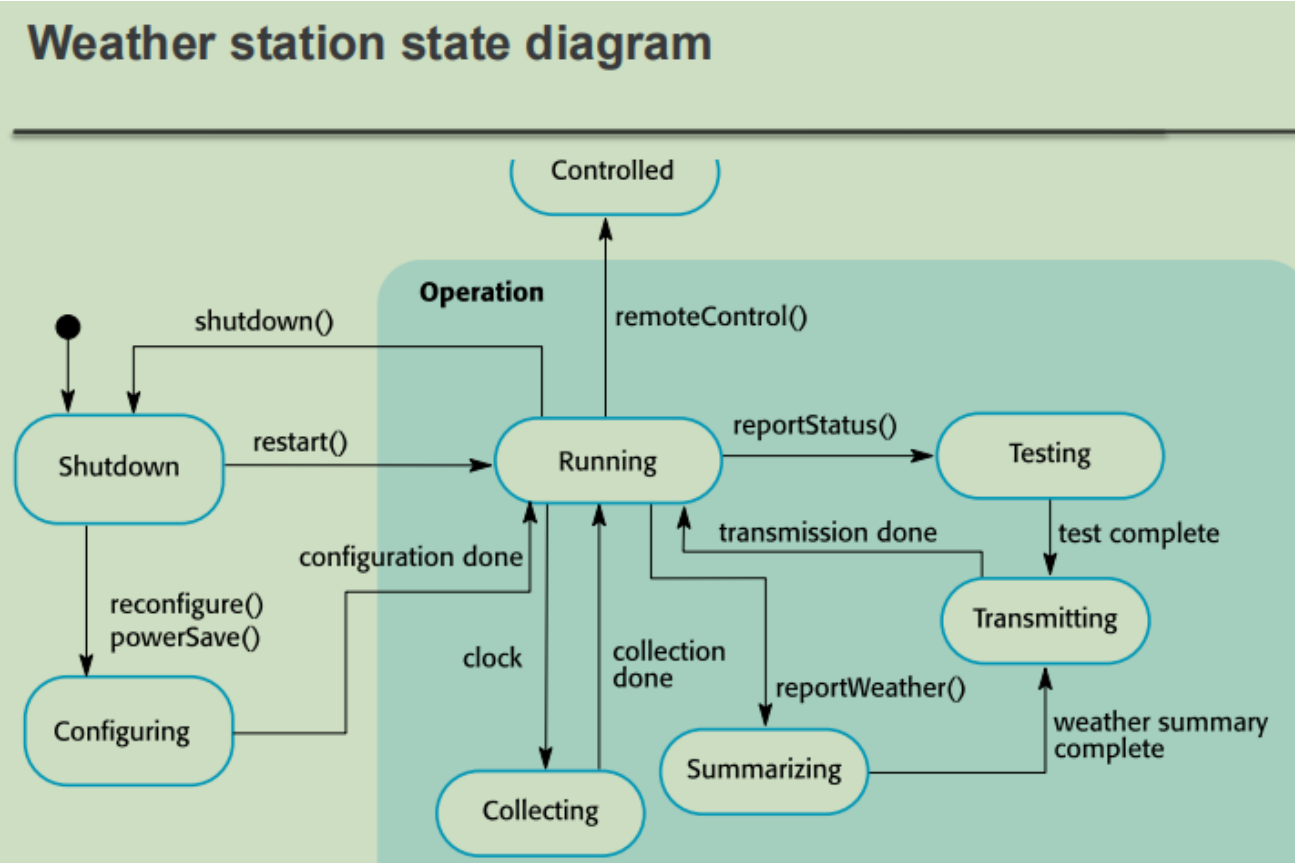






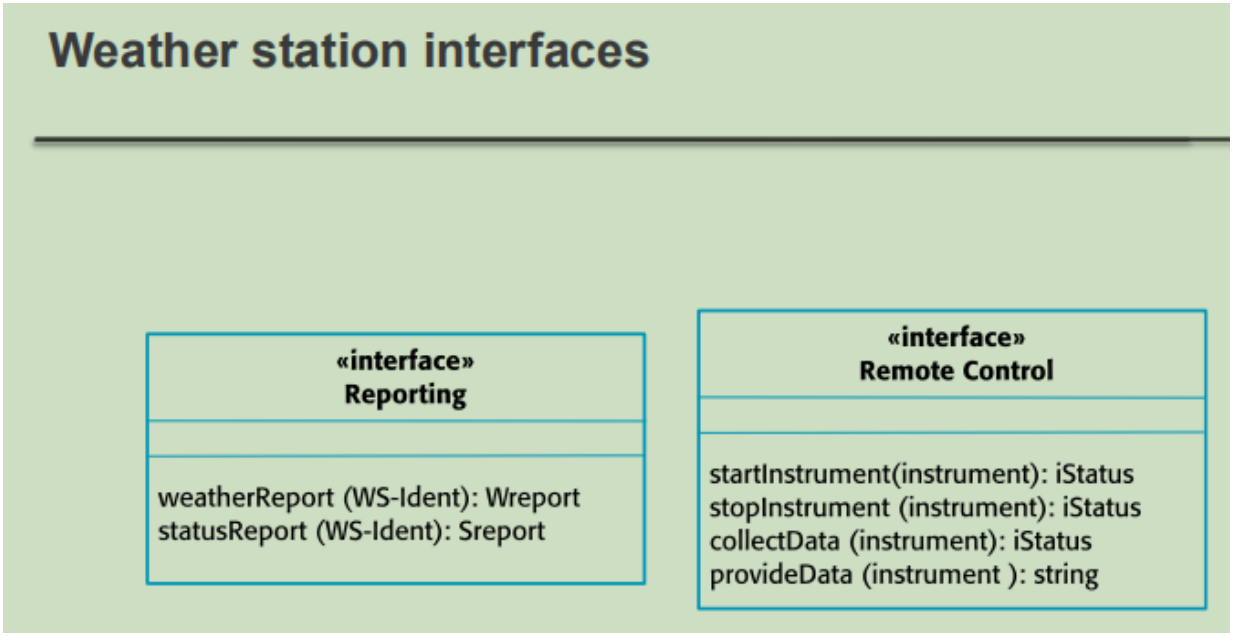
Design models

- Design models show the objects and object classes and relationships between these entities
- There are two kinds of design models
  - Structural models descibe the static structure of the system in terms of object classes and relationships
  - Dynamic models describe the dynamic interactions between objects
- Examples of design models
  - Subsystem models that show logical groupings of objects into coherent subsystems
  - Sequence models that show the sequence of object interactions
  - State machine models that show how individual objects change their state in response to events
  - Other models include use-case models, aggregation models, generalisation models, etc.



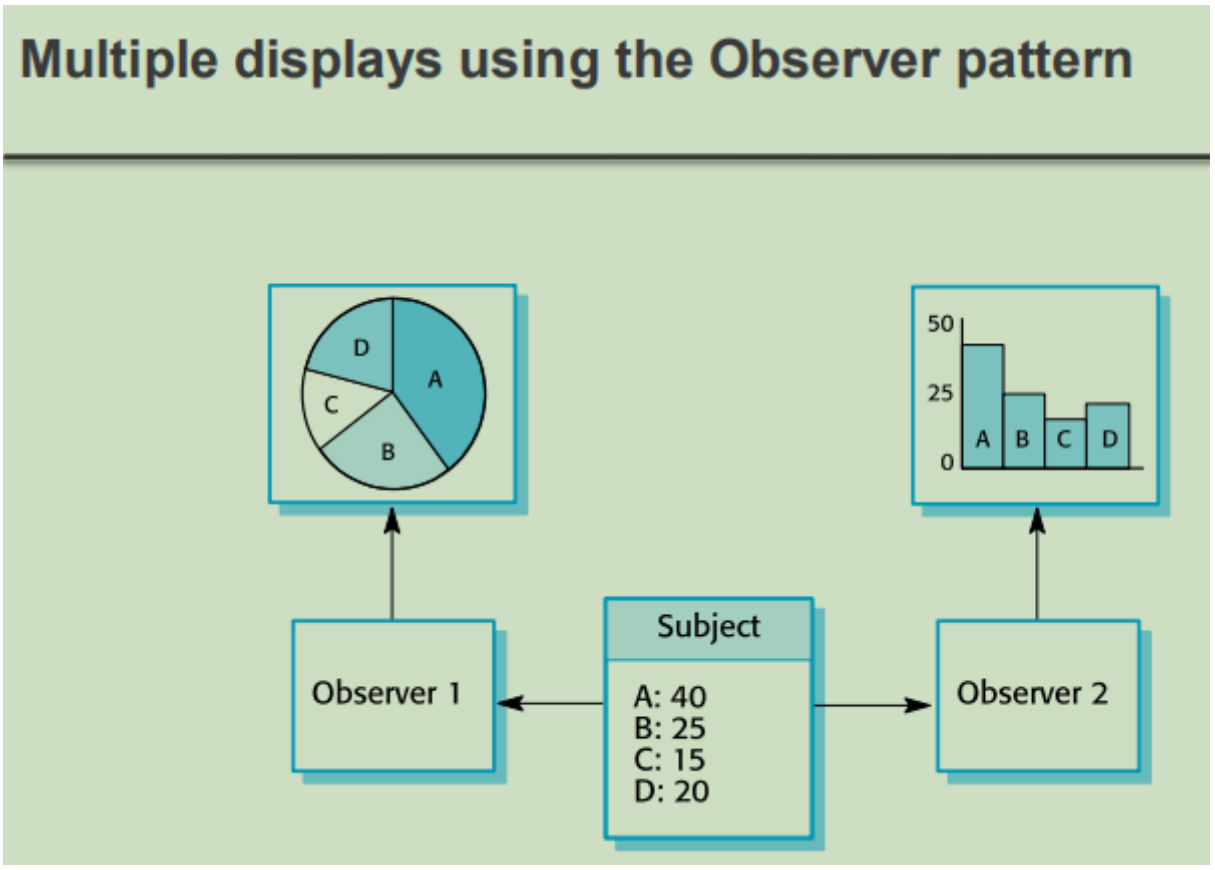
Interface specification

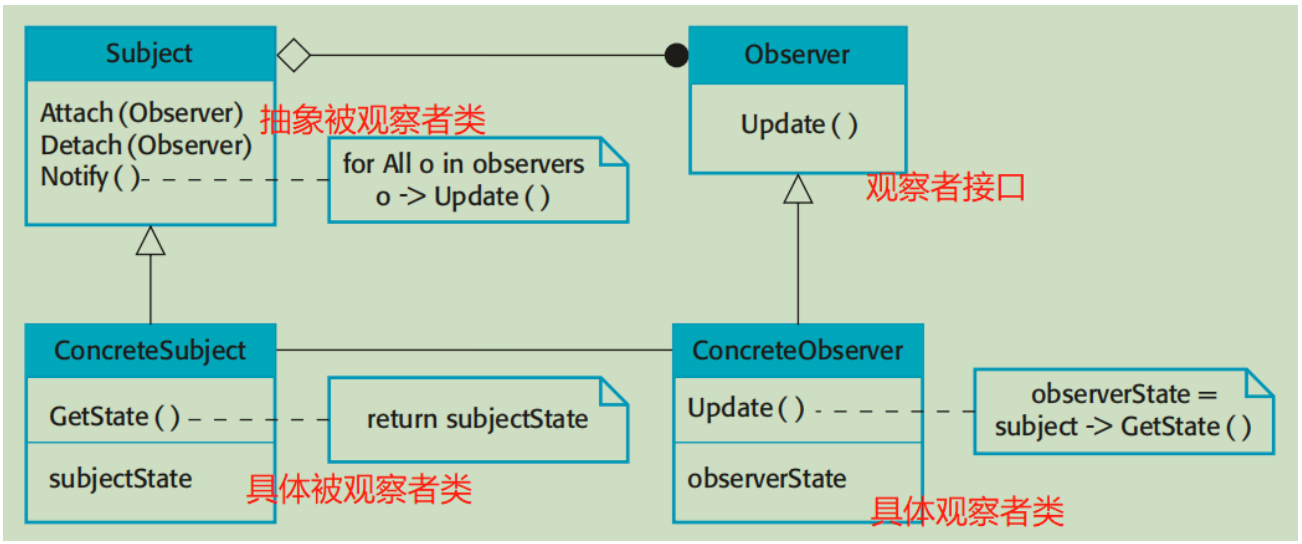
- Object interfaces have to be specified so that the objects and other components can be designed in parallel



Design patterns

- A design pattern is a way of reusing abstract knowledge about a problem and its solution. It is a description of the problem and the essence of its solution.
- Pattern elements (e.g. Observer pattern)
  - Name: a meaningful pattern identifier (Observer)
  - Problem description (Used when multiple displays of state are needed)
  - Solution descriptison
    - Not a concrete design but a template for a design solution that can eb instantiated in different ways (See slide with UML description)
  - Consequences:
    - the results and trade-offs of applying the pattern (Optimisations to enhance display performance are impractical)



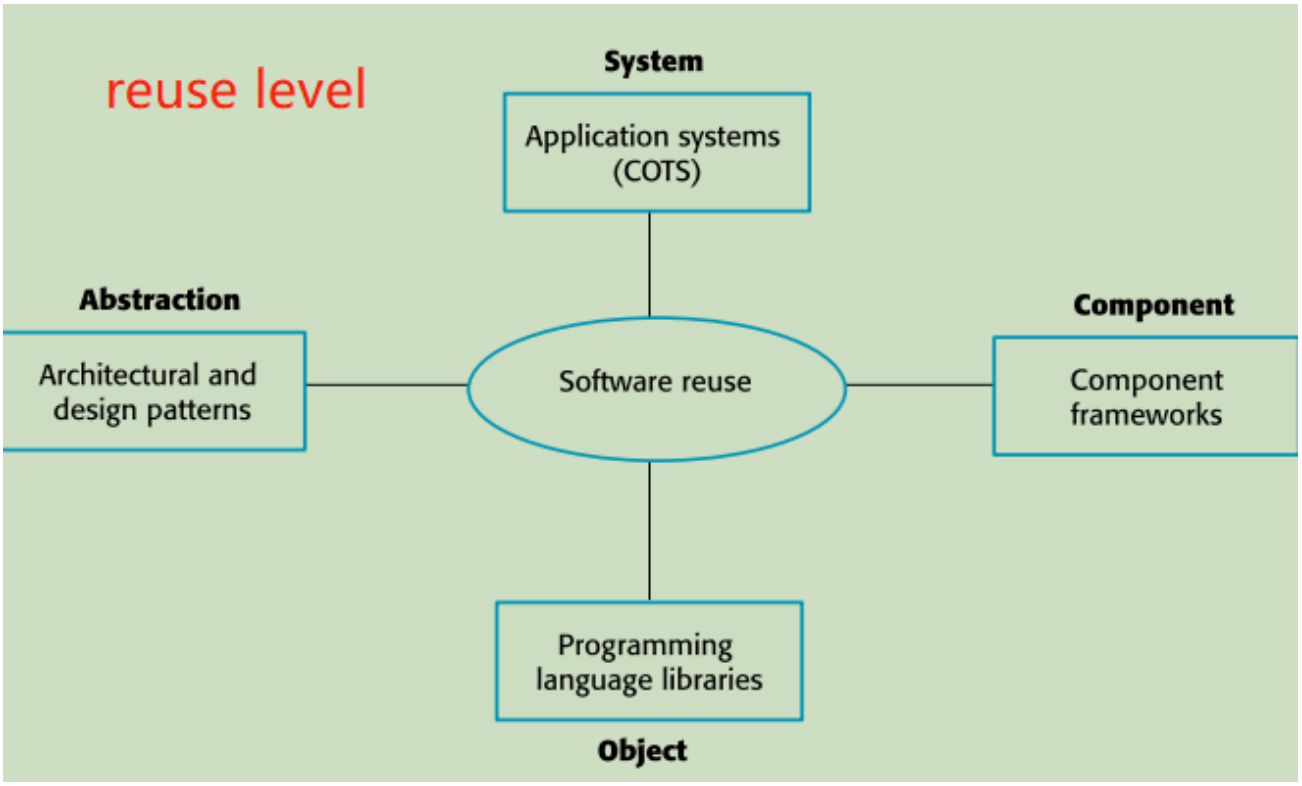


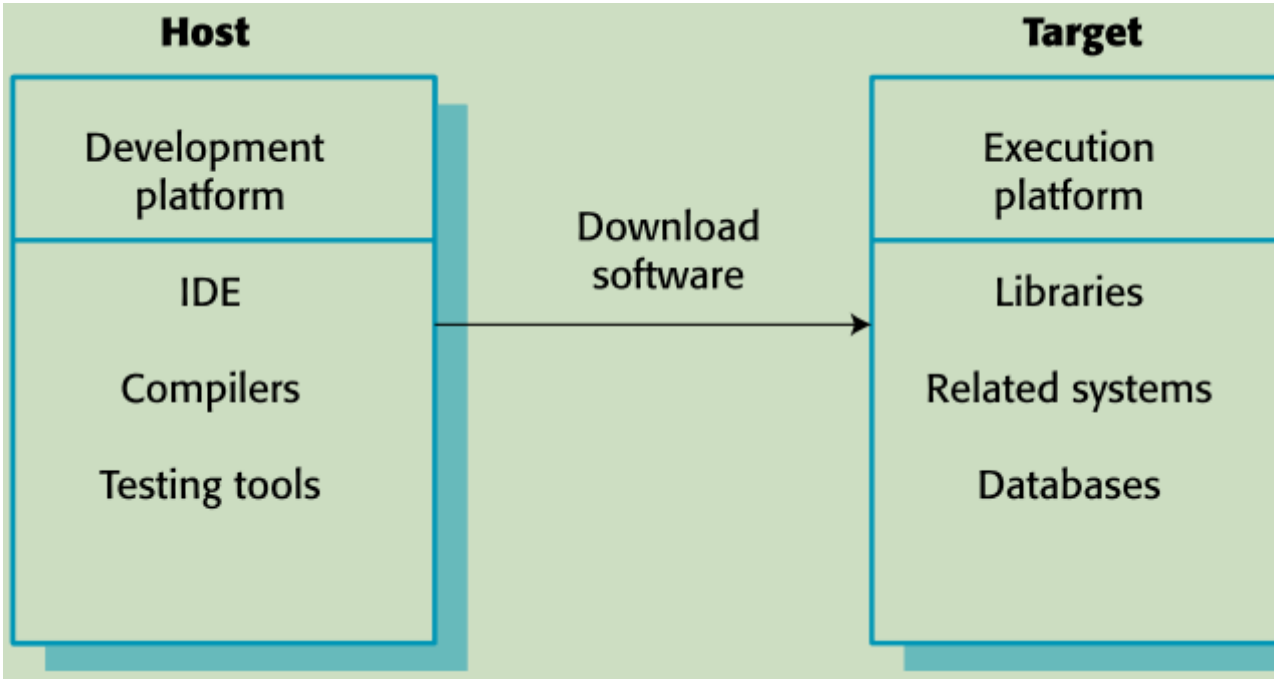
例如烧水和报警器

Implementation issues

- Implementation issues
  - **Reuse**: 尽量使用已有代码
    - Reuse level
  - **Configuration management**: Keep track of the many different versions of each software component in a configuration management system. Configuration management activities:
    - Version management: 版本管理，跟踪软件组件的不同版本
    - System integration: 系统集成， 提供（开发人员选取每个版本组件）的帮助

- **Host-target development**: production software doesn't execute on the same computer as the software development environment.





- Integrated development environments (IDEs)
  - A software tools that supports different aspects of software development, within some common framework and user interface

Open source development

- open source development is an approach to software development in which the source code of a software system is published and volunteers are invited to participate in the development process

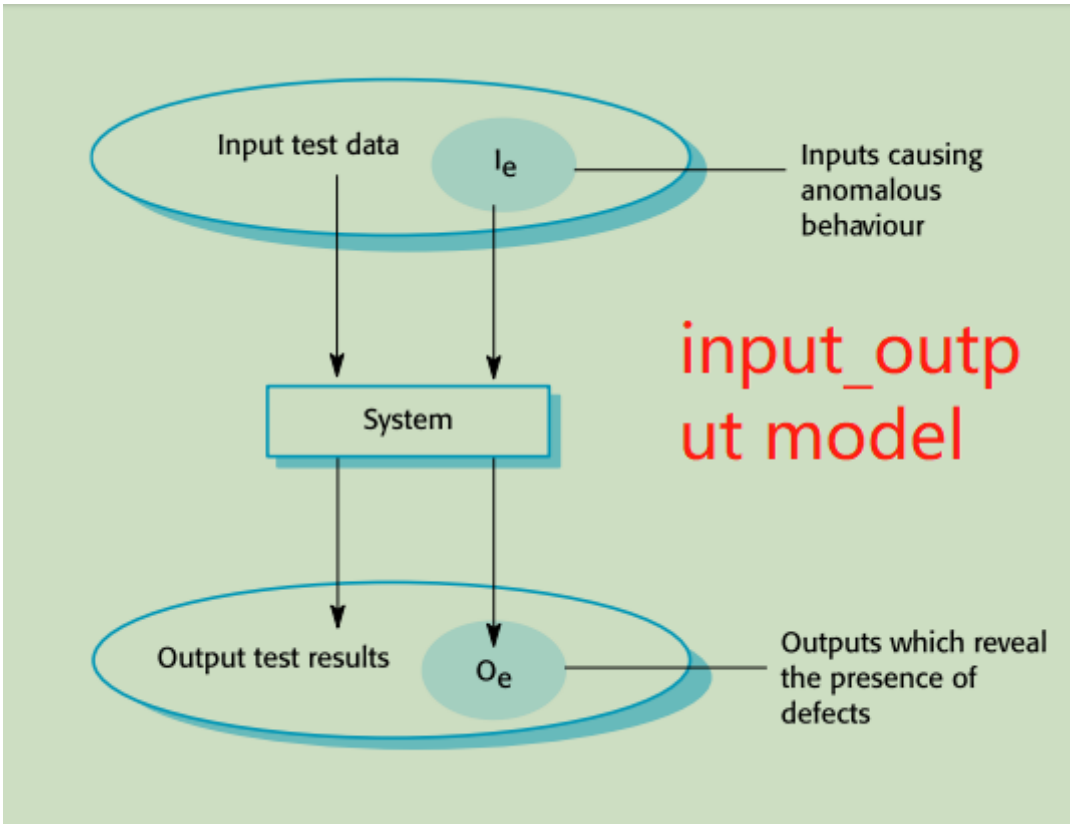
- Their business model is not reliant on selling a software product but not on selling support for that product

Conclusion

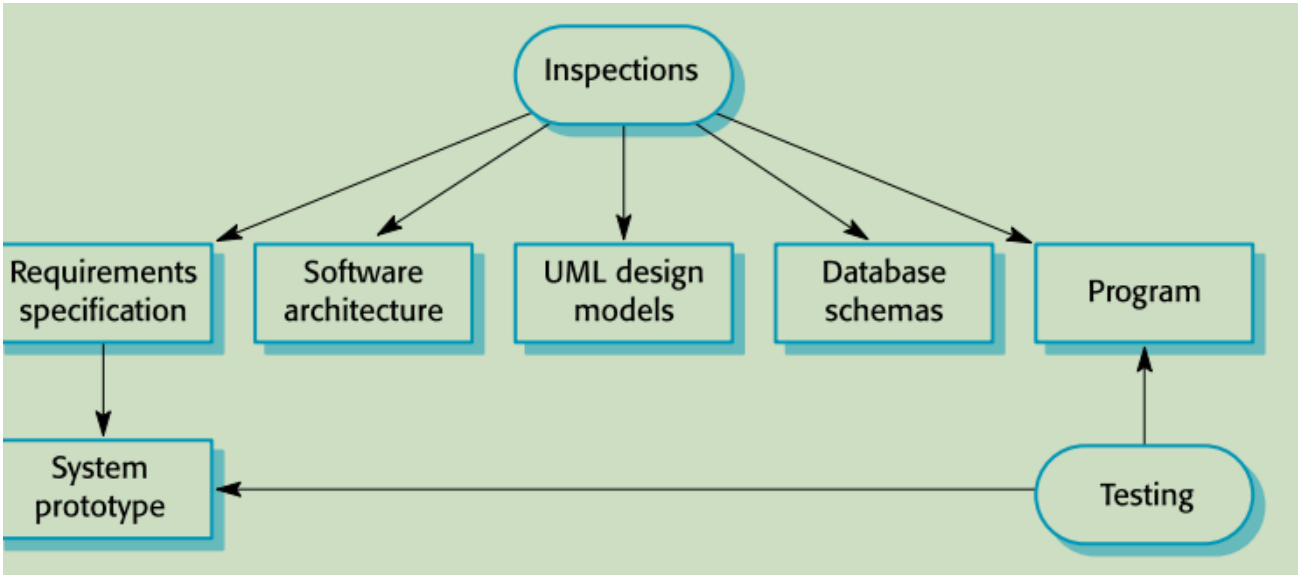
- software design and implementation are inter-leaved activities. The level of detail in the design dependes on the type of system and whether you are using a plan-driven or agile approach
- the process of object-oriented design includes activities to desin the system architecutre, indentify objects in the system, describe the design using different object models and document the component interfaces
- a range of different models may be produced during an object-oriented design process. These include static models (class models, generalization models, association models) and dynamic models (swquence models, statemachine models)
- component interfaces must be defined precisely so that other objects can use them. A UML interface may be used to define interfaces
- When developing software, you should always consider the possibility of reusing existing software, either as components, services or complete systems
- Configuration management is the process of managing changes to an evolving software system. It is essential when a team of people are cooperating to develop software
- Open source development involves making the source code of a system publicly available. This means that many people can propose changes and improvements to the software.

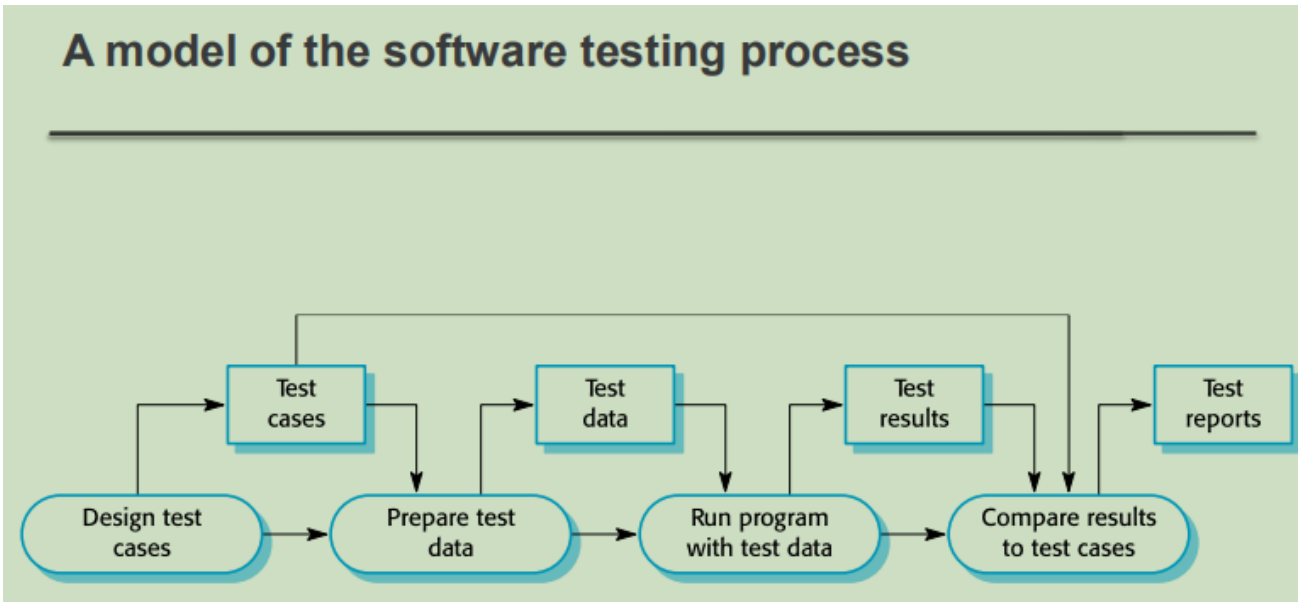
L9 Software Testing

- 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.
- The first goal leads to validation testing
    - Test cases reflect the system's expected use.
- The second goal leads to defect testing
    - Test cases in defecting can be deliberately obscure and need not reflect how the system is normally used.



- Verification vs Validation
  - verification: "Are we building the product right"
  - validation: "Are we building the right product"
- Inspections and testing:
  - **software inspections** concerned with analysis of the static system representation to discover problems (static verification). 不需要execute system
  - **software testing**: Concerned with exercising and observing product behavior (dynamic verification)





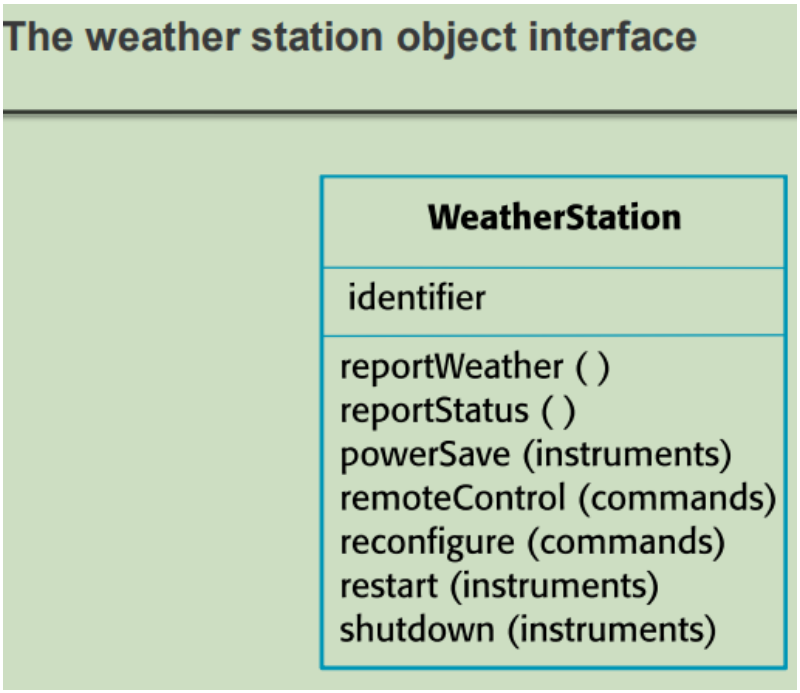
- Stages of testing
  - development testing: where the system is tested during development to discover bus and defects
  - release testing: where a separate testing team test a complete version of the system before it is released to users
  - user testing: where users or potential users of a system test the system in their own environment

Development testing

- Development testing includes all testing activities that are carried out by the team developing the system.
  - Unit testing: individual program units or object classes are tested. Unit testing should focus on testing the funcitonality of objects or methods. **defect testing process**

    - 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 system is tested as a whole.

- e.g. Weather station testing. Define test cases for reporWeather, calibrate, test, startup, shutdown.



- 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.
- JUnit framework
- Three parts:
    - setup part: initialize with test case
    - call part: call object or method to be tested
    - assertion part: compare result with expected
- validation and defects cause two types unit test case.

....

- Testing atrategies
  - partition testing: groups of inputs (common characteristics)
  - guideline-based testing: previous errors
- General testing guidelines
  - 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: focus on showing that the component interface behaves according to its specification
- Interface testing: 检测错误. Four types:
  - parameter interfaces: data passed from one method or procedure to another
  - shared memory interfaces:....
  - procedural interfaces: ...
  - message passing interface:....
- interface errors
  - interface misuse: in its use
  - interface misunderstnding: a calling component embeds assumptions about the behavior of the called component which are incorrect
  - timing errors: different speeds, out-of-date information is accessed

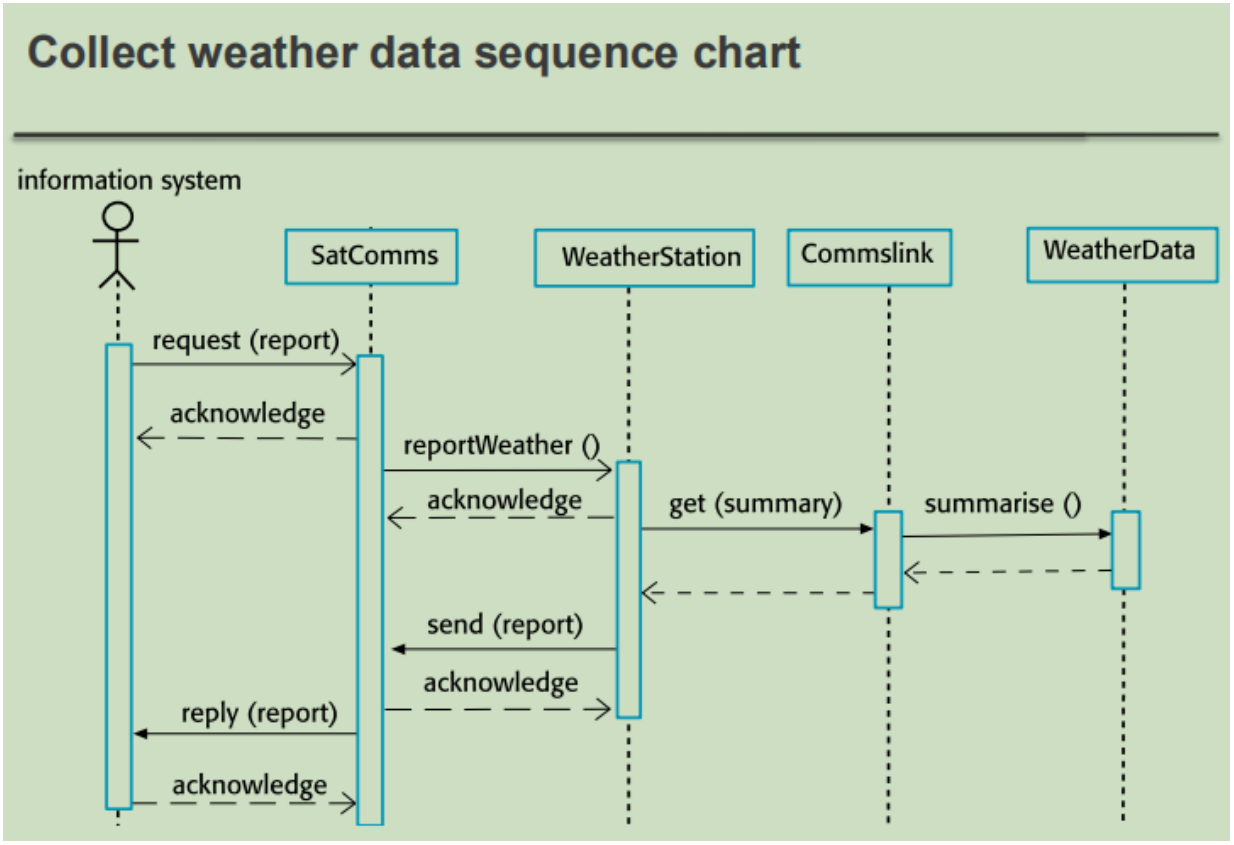
System testing

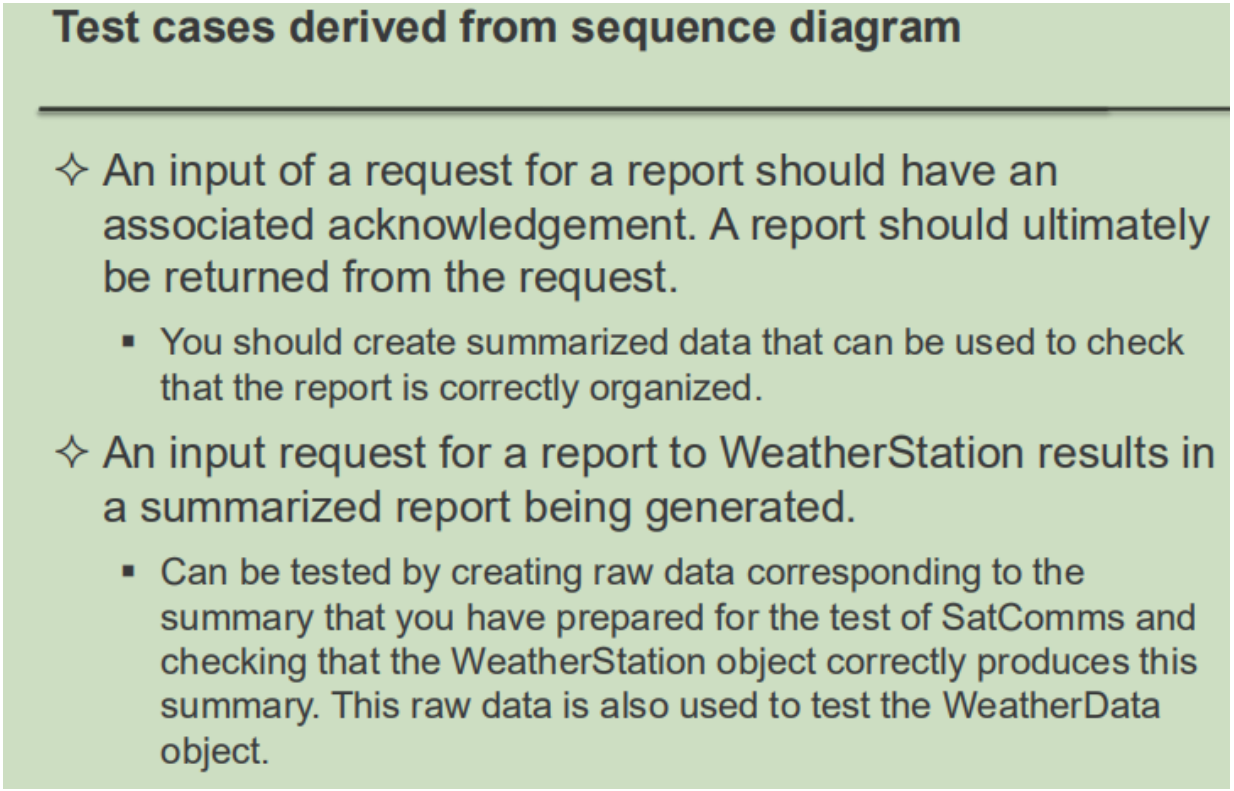
- It during development involves integrating components to create a version of the system and then testing the integrated system
- **interactions between components**

Use-case testing

- It is developed to indentify system interactions and can be used as a basis for system testing

- e.g.





Test-driven development

- TDD is an approach to program development in which you inter-leave testing and code development
- Before code. Develop code incrementally, along with a test for that increment
- as part of agile methods. also can be plan-driven development processes.

- Benefits
  - code coverage
  - regression testing: check that changes have not "broekn" previously working code; simple and straighforward in automated testing
  - simplified debugging
  - system documentation

Release testing

- It is the process of 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 is good enough for use
- black-box
- A form of system testing, but
  - separate team should be responsible for release testing
  - objectives are different

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
- 必不可少：工作环境不可复制
- Types
  - alpha testing: work with development team at developer's site
  - beta testing: release version, allow experiment, raise problems
  - acceptance testing: custom systems. whether or not accept.

Conclusion

- testing can only show the presence of errors in a program. It can't 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 customores
- development testing includes unit testing, in which you test individual objects and methods component testing inwhich you test related groups of objects and system testing, in which you test partial or complete systems

- when testing software, try to break software by using experience and guidelines to choose types of test case that have been effective in discovering defects in other systems
- automated tests are good. run every time a change is made to a system
- test-first development : tests are written before the code to be tested
- scenario testing involves inventing a typical usage scenario and using this to derive test cases
- accepatance testing is a user testing process where the aim is to decide if the software is good enough to be deployed and used in is operational environment