0.1 Introduction

0.1.1 Software design and implementation

- It is the stage in the software engineering process at which an executable software system is developed.
- Its activities are invariably inter-leaved.

0.1.2 Design levels

- **High-level design or architectural design:** partition the system into components.
- **Detailed design:** partition each component (or small program) into classes.
- Design of algorithms and data structures

0.2 Object-oriented design using the UML

Structured object-oriented design processes involve developing a number of different system models.
Since they require a lot of effort for development and maintenance, this models are only cost-effective for large systems.

0.2.1 Process stages

- Define the system context and use cases;
- Design the system architecture;
- Identify the principal object classes in the system;
- Develop design models;
- Specify object interfaces (API).

0.2.2 Design models and design views

An object-oriented design model may in general cover 4 inter-related design views.

refinment

		Static (or Structural)	Dynamic (or Behavioral)
External		Class diagram (API, interfaces)	Sequence diagram (external interactions)
Internal	Inter-class	Class diagram (all except private features)	Sequence diagram (inter-object interactions)
	Intra-class	Class diagram (private features per class)	State machine diagram (internal behavior per class)

0.2.3 Sequence diagrams (SD)

Sequence diagrams show the sequence of object interactions that take place. (usually in the context of a system use case).

0.2.4 State machine diagrams (SMD)

- State machine diagrams are used to show how objects respond to different service requests and the state transitions triggered by these requests;
- State machine diagrams are useful high-level models of a system or an object's run-time behavior;
- You don't usually need a state machine diagram for all of the objects in the system because most objects in a system are simple.

0.2.5 Interface specification

- Object interfaces have to be specified;
- Objects may have several interfaces which are viewpoints on the methods provide;
- The UML uses class diagrams for interface specification;

0.3 Design patterns

- Way of reusing abstract knowledge about a problem and its solution;
- A pattern is a description of the problem and the essence of its solution;
- It should be sufficiently abstract to be reused in other cases;
- Pattern descriptions usually make use of object-oriented characteristics,
- Any design problem may have an associated pattern that can be applied.

0.3.1 Pattern elements

- Name;
- Problem description;
- Solution description. (template for a design solution);
- Consequences (results).

0.4 Implementation issues

0.4.1 Reuse

Most modern software is constructed by reusing existing components or systems.

Reuse levels:

- **The abstraction level:** Don't reuse software directly but use knowledge of successful abstractions;
- **The object level:** Directly reuse objects from a library;
- **The component level:** Components are collections of objects and object classes that you reuse in application systems;
- **The system level:** reuse entire application systems.

Reuse costs:

- The costs of the time spent in looking for software to reuse and assessing whether or not it meets your needs;
- The costs of buying the reusable software;
- The costs of adapting and configuring the reusable software components or systems;
- The costs of integrating reusable software elements with each other and with the new code developed.

0.4.2 Configuration management

Configuration management is the name given to the general process of managing a changing software system.
Its aim is to support the system integration process.

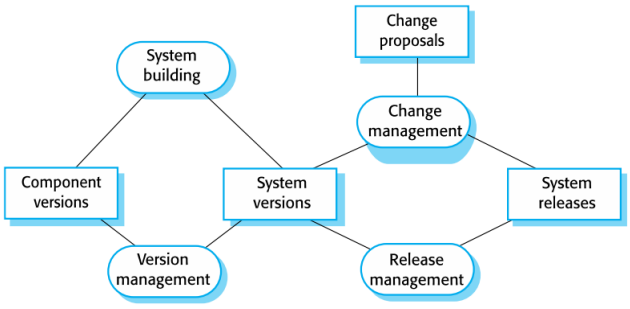
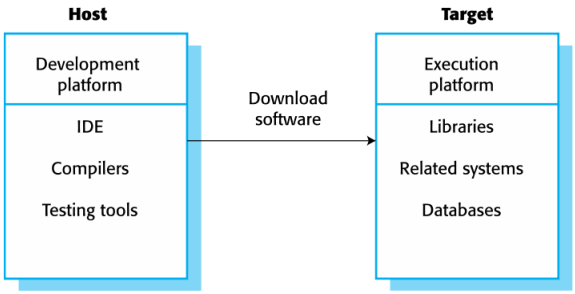


Figure 1: Configuration management tool interaction

0.4.3 Host-target development

Most software is developed on one computer(the host), but runs on a separate machine (the target).



0.4.4 Development platform tools

- Integrated compiler and syntax-directed editing system that allows you to create, edit and compile code;
- Language debugging system;
- Graphical editing tools;
- Testing tools;
- Project support tools that help you organize the code for different development projects.

0.4.5 Integrated development environments (IDEs)

- Software development tools are often grouped to create an integrated development environment (IDE);

- Is a set of tools that supports different aspects of software development;
- IDEs are created to support development in a specific programming language. The language IDE may be developed specially, or may be an instantiation of a general-purpose IDE.

0.5 Key points (Design)

- Software design and implementation are inter-leaved activities. The level of detail in the design depends 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 design the system architecture, identify 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 and dynamic models.
- Component interfaces must be defined precisely so that other objects can use them.

0.6 Key points (Implementation)

- When developing software, you should reuse existing software.
- Configuration management is the process of managing changes to an evolving software system.
- Most software development is host-target development. You use an IDE on a host machine to develop the software, which is transferred to a target machine for execution.