Software Design

Rakibul Hassan
Lecturer
Dept. of ECE, RUET

Software Design

Software design is a mechanism to transform user requirements into some suitable form, which helps the programmer in software coding and implementation. It deals with representing the client's requirement, as described in SRS (Software Requirement Specification) document, into a form, i.e., easily implementable using programming language.

Software Design Process

The software design process can be divided into the following three levels of phases of design:

1. Interface Design
2. Architectural Design
3. Detailed Design

Software Design Process

Interface design

- Interface design is the specification of the interaction between a system and its environment.
- During interface design, the internal of the systems are completely ignored and the system is treated as a black box. Attention is focused on the dialogue between the target system and the users, devices, and other systems with which it interacts.
- The design problem statement produced during the problem analysis step should identify the people, other systems, and devices which are collectively called agents.

Software Design Process

Interface design

Interface design should include the following details:

- Precise description of events in the environment, or messages from agents to which the system must respond.
- Precise description of the events or messages that the system must produce.
- Specification on the data, and the formats of the data coming into and going out of the system.
- Specification of the ordering and timing relationships between incoming events or messages, and outgoing events or outputs.

Software Design Process

Architectural design

Architectural design is the specification of the major components of a system, their responsibilities, properties, interfaces, and the relationships and interactions between them. In architectural design, the overall structure of the system is chosen, but the internal details of major components are ignored.

Software Design Process

Architectural design

Issues in architectural design includes:

- Gross decomposition of the systems into major components.
- Allocation of functional responsibilities to components.
- Component Interfaces Component scaling and performance properties, resource consumption properties, reliability properties, and so forth.
- Communication and interaction between components.

Software Design Process

Detailed Design

Design is the specification of the internal elements of all major system components, their properties, relationships, processing, and often their algorithms and the data structures.

Software Design Process

Detailed Design

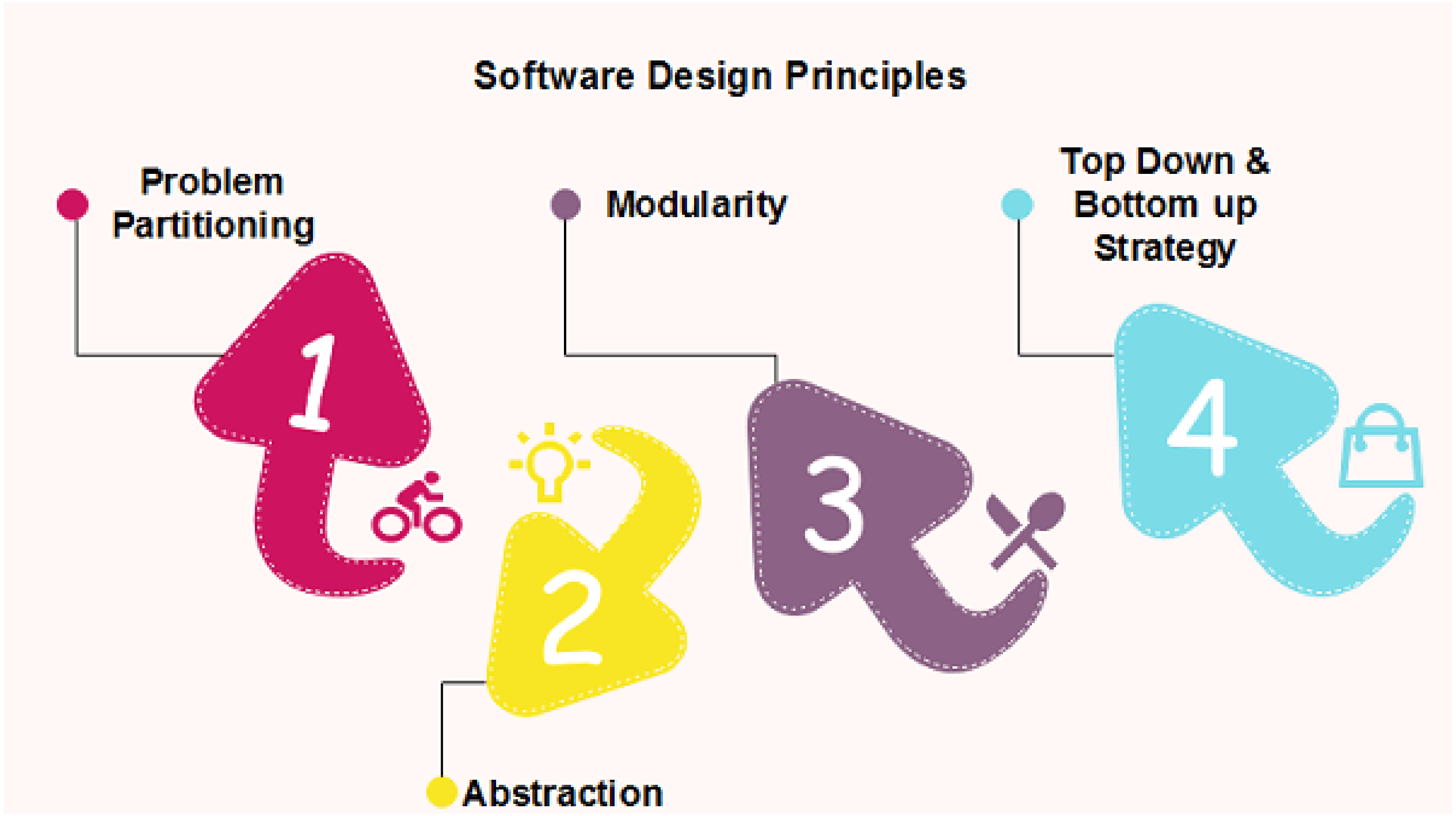
The detailed design may include:

- Decomposition of major system components into program units.
- Allocation of functional responsibilities to units.
- User interfaces
- Unit states and state changes
- Data and control interaction between units
- Data packaging and implementation, including issues of scope and visibility of program elements
- Algorithms and data structures

Software Design Principles

Software design principles are concerned with providing means to handle the complexity of the design process effectively. Effectively managing the complexity will not only reduce the effort needed for design but can also reduce the scope of introducing errors during design.

Software Design Principles



Software Design Principles

Problem Partitioning

For small problem, we can handle the entire problem at once but for the significant problem, divide the problems and conquer the problem it means to divide the problem into smaller pieces so that each piece can be captured separately. For software design, the goal is to divide the problem into manageable pieces.

Software Design Principles

Problem Partitioning

Benefits of Problem Partitioning

- Software is easy to understand
- Software becomes simple
- Software is easy to test
- Software is easy to modify
- Software is easy to maintain
- Software is easy to expand

Software Design Principles

Problem Partitioning

These pieces cannot be entirely independent of each other as they together form the system. They have to cooperate and communicate to solve the problem. This communication adds complexity.

Software Design Principles

Abstraction

An abstraction is a tool that enables a designer to consider a component at an abstract level without bothering about the internal details of the implementation. Abstraction can be used for existing element as well as the component being designed. Here, there are two common abstraction mechanisms-

- Functional Abstraction
- Data Abstraction

Software Design Principles

Abstraction

Functional Abstraction

- A module is specified by the method it performs.
- The details of the algorithm to accomplish the functions are not visible to the user of the function

Data Abstraction

Details of the data elements are not visible to the users of data. Data Abstraction forms the basis for **Object Oriented design**

.

Software Design Principles

Modularity

Modularity specifies to the division of software into separate modules which are differently named and addressed and are integrated later on in to obtain the completely functional software. It is the only property that allows a program to be intellectually manageable. Single large programs are difficult to understand and read due to a large number of reference variables, control paths, global variables, etc.

Software Design Principles

Modularity

The desirable properties of a modular system are:

- Each module is a well-defined system that can be used with other applications.
- Each module has single specified objectives.
- Modules can be separately compiled and saved in the library.
- Modules should be easier to use than to build.
- Modules are simpler from outside than inside

Software Design Principles

Modularity

There are several **advantages** of Modularity

- It allows large programs to be written by several or different people
- It encourages the creation of commonly used routines to be placed in the library and used by other programs.
- It simplifies the overlay procedure of loading a large program into main storage.
- It provides more checkpoints to measure progress.
- It provides a framework for complete testing, more accessible to test
- It produced the well designed and more readable program.

Software Design Principles

Modularity

There are several **disadvantages** of Modularity

- Execution time maybe, but not certainly, longer
- Storage size perhaps, but is not certainly, increased
- Compilation and loading time may be longer
- Inter-module communication problems may be increased
- More linkage required, run-time may be longer, more source lines must be written, and more documentation has to be done

Software Design Principles

Modular Design

Functional Independence

Functional independence is achieved by developing functions that perform only one kind of task and do not excessively interact with other modules. Independence is important because it makes implementation more accessible and faster. The independent modules are easier to maintain, test, and reduce error propagation and can be reused in other programs as well. Thus, functional independence is a good design feature which ensures software quality.

Software Design Principles

Modular Design

Functional Independence

It is measured using two criteria:

- Cohesion
- Coupling

Software Design Principles

Modular Design

Functional Independence

Cohesion: It measures the relative function strength of a module. Cohesion is a measure that defines the degree of intra-dependability within elements of a module. The greater the cohesion, the better is the program design.

* Types of Cohesion (Self study)

Software Design Principles

Modular Design

Functional Independence

Coupling: It measures the relative interdependence among modules. Coupling is a measure that defines the level of inter-dependability among modules of a program. It tells at what level the modules interfere and interact with each other. The lower the coupling, the better the program.

* Types of coupling (Self study)

Software Design Principles

Modular Design

Information hiding: The fundamental of Information hiding suggests that modules can be characterized by the design decisions that protect from the others, in other words, modules should be specified that data include within a module is inaccessible to other modules that do not need for such information.

The use of information hiding as design criteria for modular system provides the most significant benefits when modifications are required during testing's and later during software maintenance. Inadvertent errors introduced during modifications are less likely to propagate to different locations within the software.

Software Design Principles

Strategy of Design

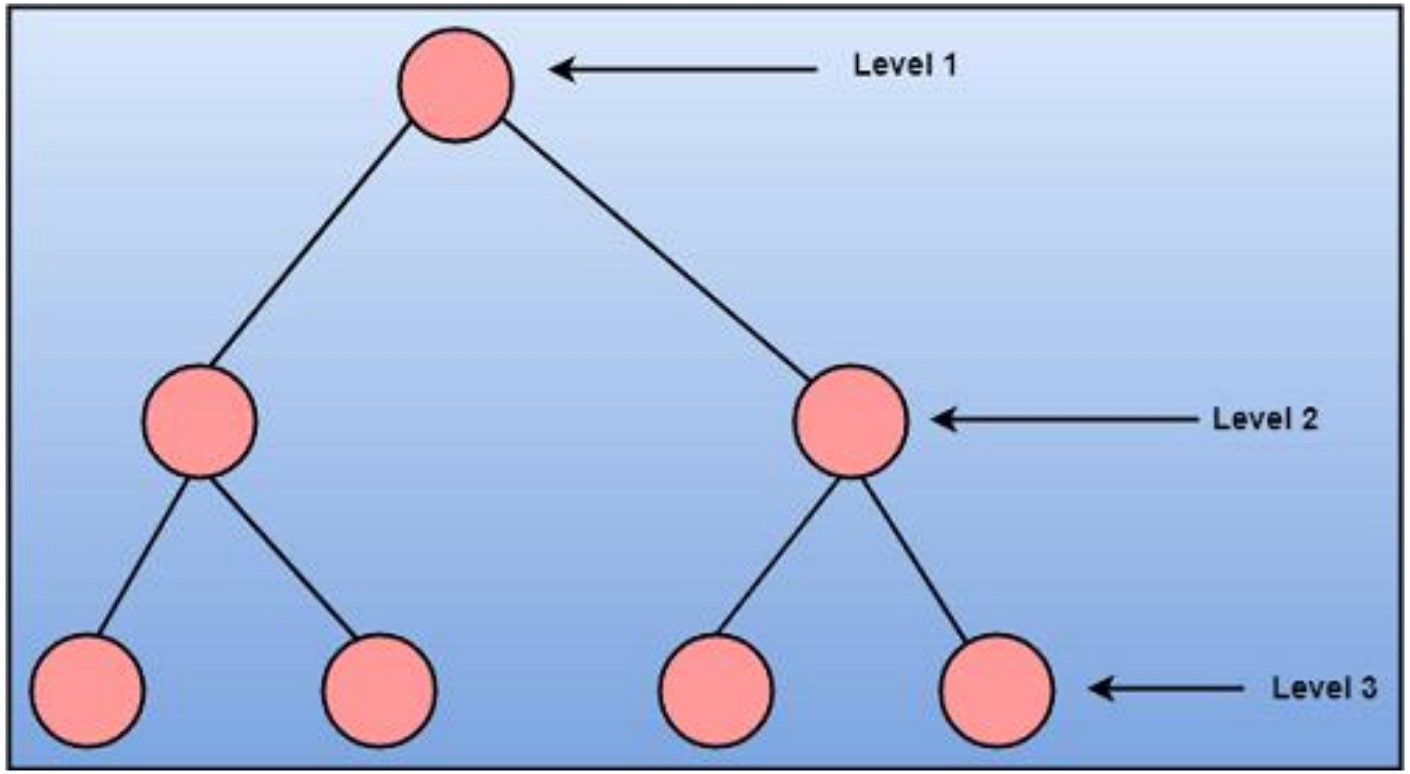
A good system design strategy is to organize the program modules in such a method that are easy to develop and latter too, change. Structured design methods help developers to deal with the size and complexity of programs. Analysts generate instructions for the developers about how code should be composed and how pieces of code should fit together to form a program. To design a system, there are two possible approaches:

- Top-down Approach
- Bottom-up Approach

Software Design Principles

Strategy of Design

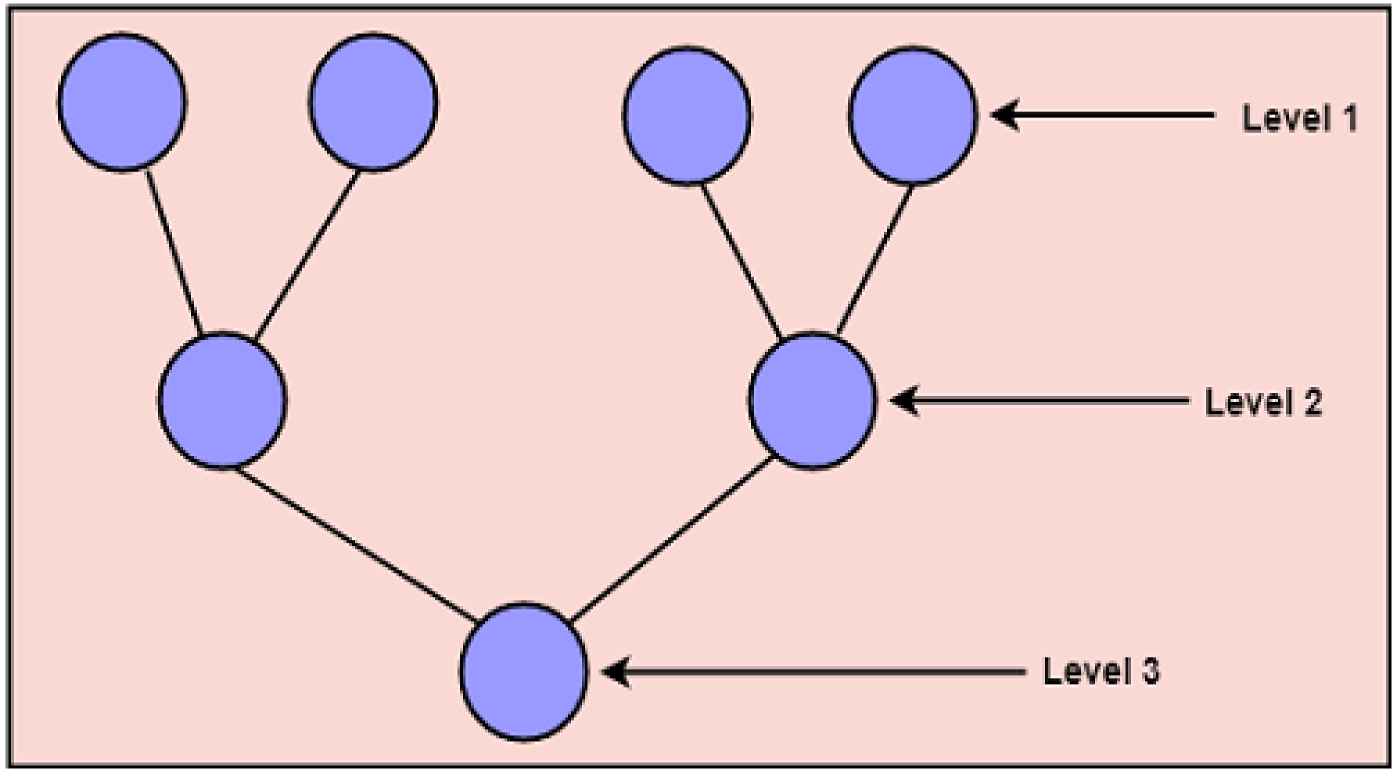
Top down approach starts with the identification of the main components and then decomposing them into their more detailed sub-components.



Software Design Principles

Strategy of Design

A **bottom-up** approach begins with the lower details and moves towards up the hierarchy, as shown in fig. This approach is suitable in case of an existing system.



Thank you