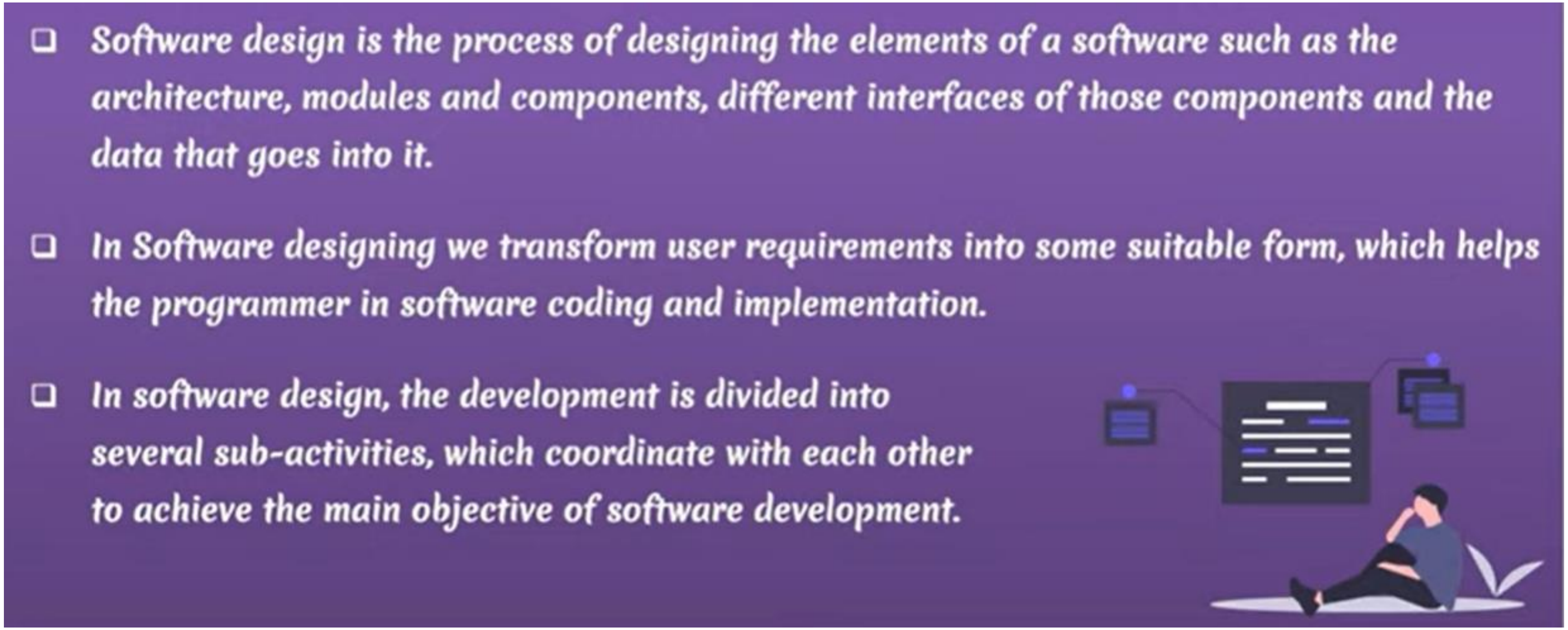
Design Principles

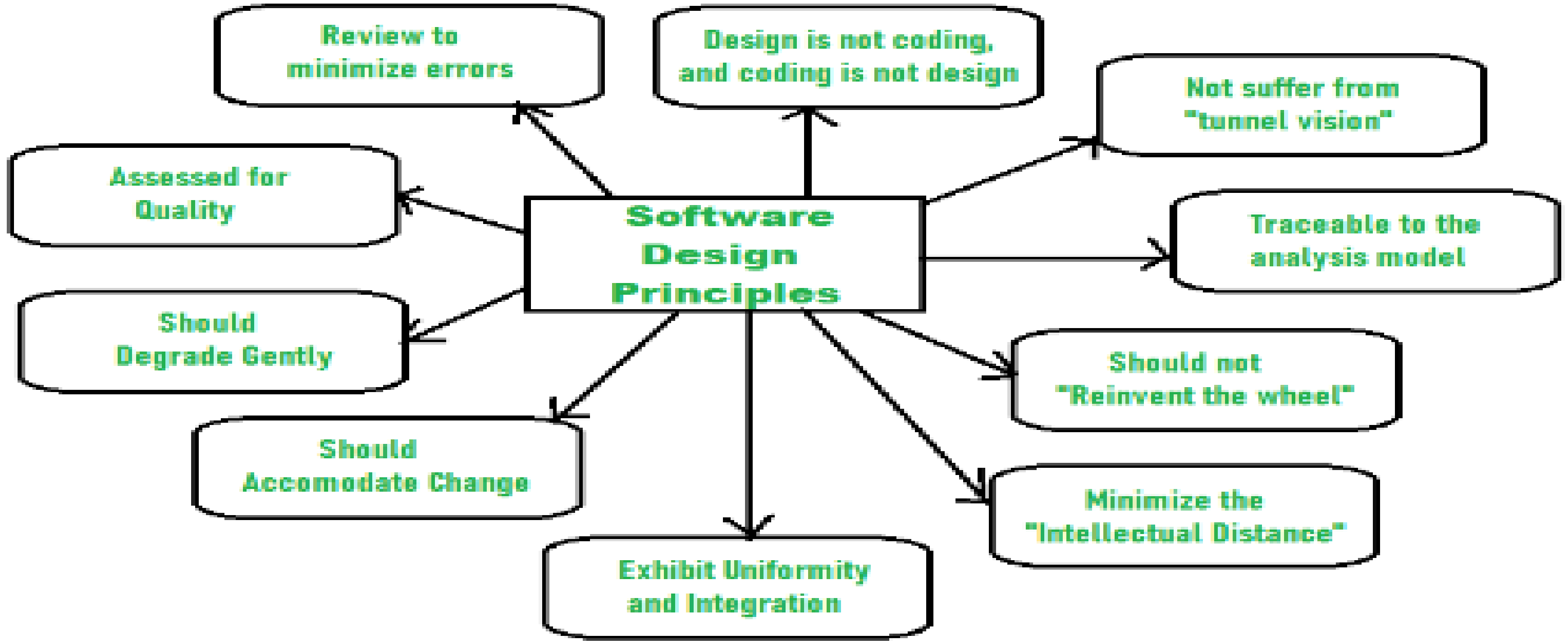
Software Design



- ❑ *Software design is the process of designing the elements of a software such as the architecture, modules and components, different interfaces of those components and the data that goes into it.*
- ❑ *In Software designing we transform user requirements into some suitable form, which helps the programmer in software coding and implementation.*
- ❑ *In software design, the development is divided into several sub-activities, which coordinate with each other to achieve the main objective of software development.*

Software Design is also a process to plan or convert the software requirements into a step that are needed to be carried out to develop a software system. There are several principles that are used to organize and arrange the structural components of Software design. Software Designs in which these principles are applied affect the content and the working process of the software from the beginning.

These principles are stated below :


Software Design Principles
Review to minimize errors
Design is not coding, and coding is not design
Not suffer from "tunnel vision"
Traceable to the analysis model
Should not "Reinvent the wheel"
Minimize the "Intellectual Distance"
Exhibit Uniformity and Integration
Should Accomodate Change
Should Degrade Gently
Assessed for Quality

Principles Of Software Design :

1. **Should not suffer from "Tunnel Vision"** –
   While designing the process, it should not suffer from "tunnel vision" which means that is should not only focus on completing or achieving the aim but on other effects also.

2. **Traceable to analysis model** –
   The design process should be traceable to the analysis model which means it should satisfy all the requirements that software requires to develop a high-quality product.

3. **Should not "Reinvent The Wheel"** –
   The design process should not reinvent the wheel that means it should not waste time or effort in creating things that already exist. Due to this, the overall development will get increased.

4. **Minimize Intellectual distance** –
   The design process should reduce the gap between real-world problems and software solutions for that problem meaning it should simply minimize intellectual distance.

5. **Exhibit uniformity and integration** –
   The design should display uniformity which means it should be uniform throughout the process without any change. Integration means it should mix or combine all parts of software i.e. subsystems into one system.

6. **Accommodate change** –
   The software should be designed in such a way that it accommodates the change implying that the software should adjust to the change that is required to be done as per the user's need.

7. **Degrade gently** –
   The software should be designed in such a way that it degrades gracefully which means it should work properly even if an error occurs during the execution.

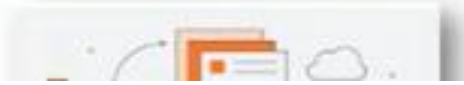
8. **Assessed or quality** –
   The design should be assessed or evaluated for the quality meaning that during the evaluation, the quality of the design needs to be checked and focused on.

9. **Review to discover errors** –
   The design should be reviewed which means that the overall evaluation should be done to check if there is any error present or if it can be minimized.

10. **Design is not coding and coding is not design** –
    Design means describing the logic of the program to solve any problem and coding is a type of language that is used for the implementation of a design.

About Software Design Concepts

- The software design concept simply means the idea or principle behind the design.
- It describes how you plan to solve the problem of designing software.
- It also shows the logic or thinking behind how you will design software.
- The software design concept for developing the right software provides a supporting and essential structure or model.



Software Design Concepts

1. **Abstraction**
2. **Architecture**
3. **Design Patterns**
4. **Modularity**
5. **Information Hiding**
6. **Functional Independence**
7. **Refinement**
8. **Refactoring**
9. **Object-Oriented Design Concept**

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

1. Functional Abstraction
2. Data Abstraction

Functional Abstraction

i. A module is specified by the method it performs.

ii. The details of the algorithm to accomplish the functions are not visible to the user of the function.

Functional abstraction forms the basis for **Function oriented design approaches**.

Data Abstraction

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

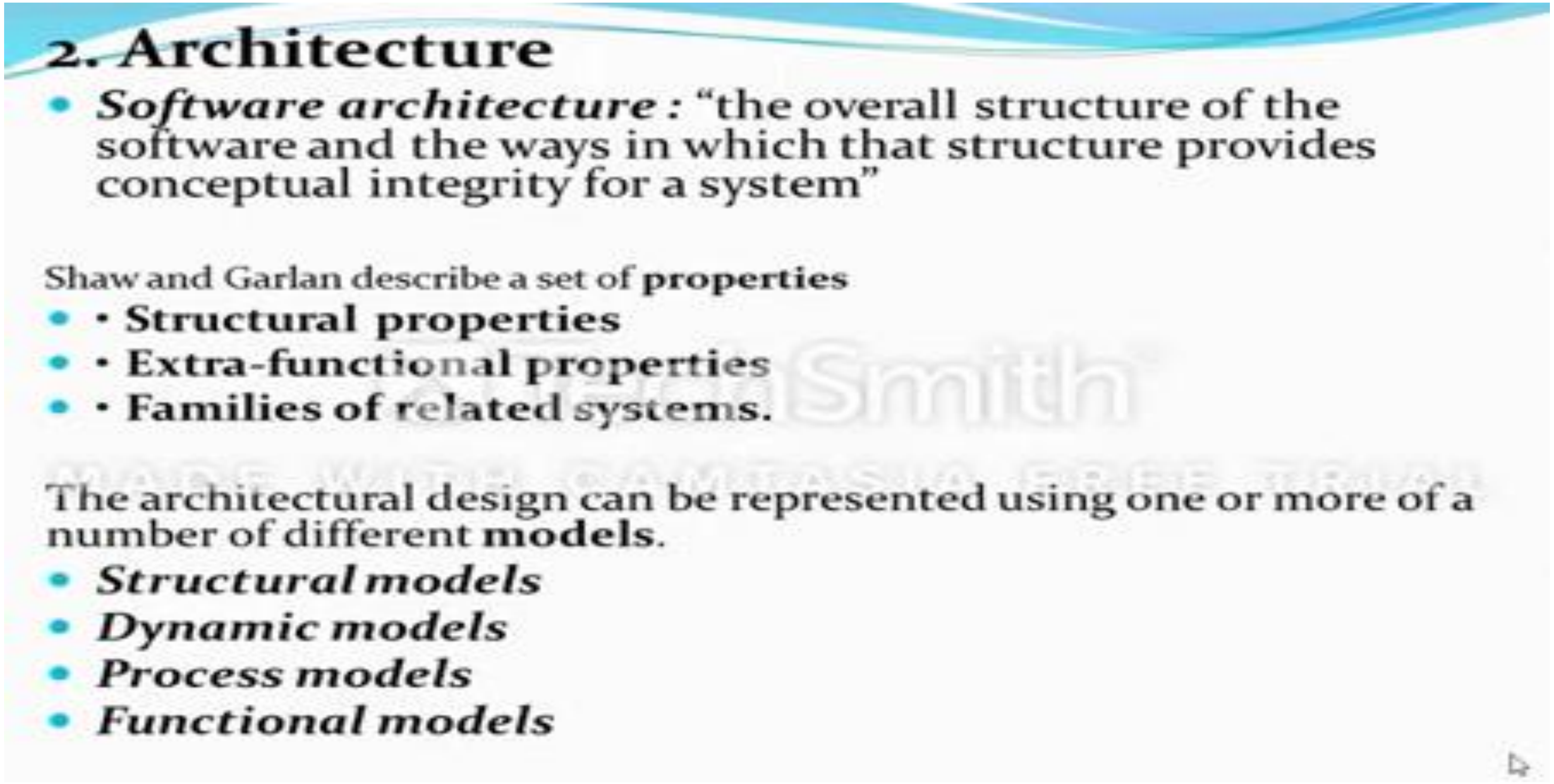
2. Architecture

- The architecture is the structure of program modules where they interact with each other in a specialized way.

➢ **Structural Properties:** Architectural design represent different types of components, modules, objects & relationship between these.

➢ **Extra-Functional Properties:** How design architecture achieve requirements of Performance, Capacity, Reliability, Security, Adaptability & other System Characteristics.

➢ **Families of related systems:** The architectural design should draw repeatable patterns. They have ability to reuse repeatable blocks.



2. Architecture

- ***Software architecture :*** "the overall structure of the software and the ways in which that structure provides conceptual integrity for a system"

Shaw and Garlan describe a set of **properties**

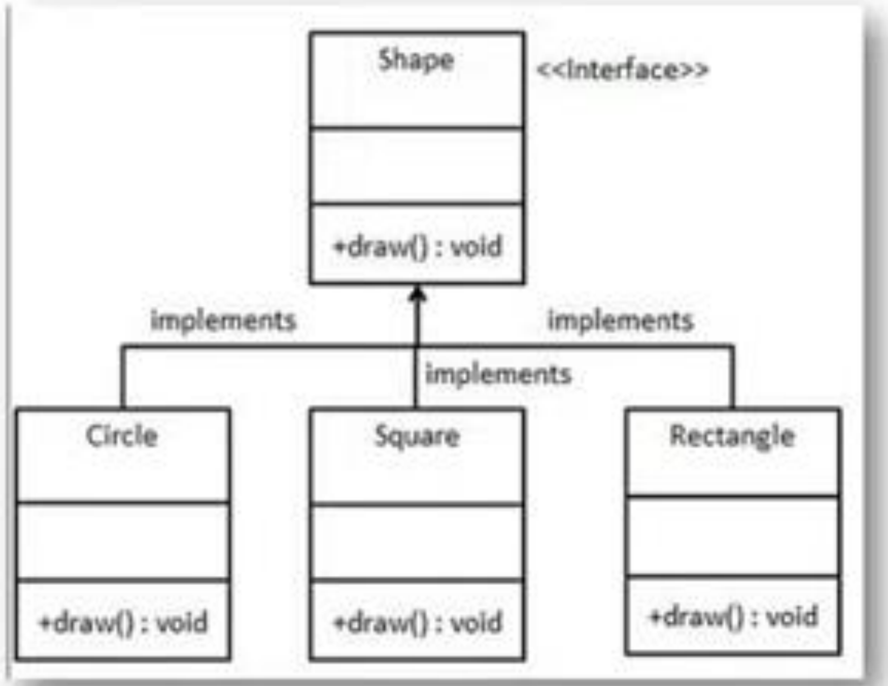
- **• Structural properties**
- **• Extra-functional properties**
- **• Families of related systems.**

The architectural design can be represented using one or more of a number of different **models**.

- ***Structural models***
- ***Dynamic models***
- ***Process models***
- ***Functional models***

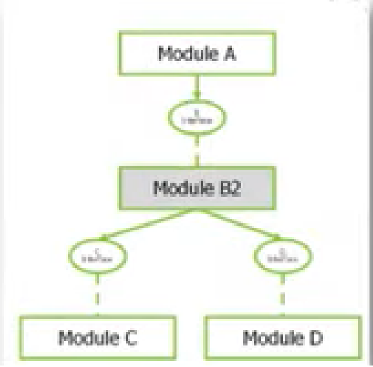
3. Design Patterns

- The pattern simply means a repeated form or design in which the same shape is repeated several times to form a pattern.
- **Example:**



4. Modularity

- Modularity simply means dividing the system or project into smaller parts to reduce the complexity of the system or project.
- After developing the modules, they are integrated together to meet the software requirements.
- Modularizing a design helps to effective development, accommodate changes easily, conduct testing, debugging efficiently and conduct maintenance work easily.



5. Modularity

- Software is divided into separately named and addressable components, sometimes called ***module***.

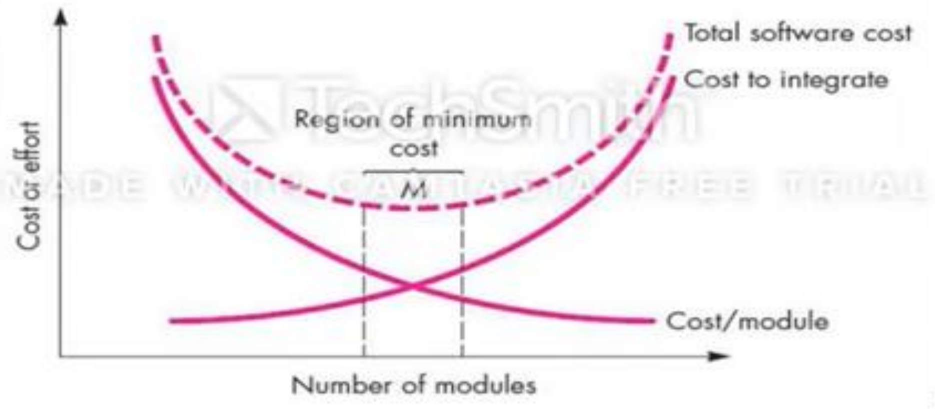


Fig : Modularity and software cost

- ❑ ***<u>Modularity:</u> It means the division of software into separate modules which are differently addressed and are integrated later on in to obtain the complete functional software.***

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.

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.

Advantages and Disadvantages of Modularity

Advantages

- 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.

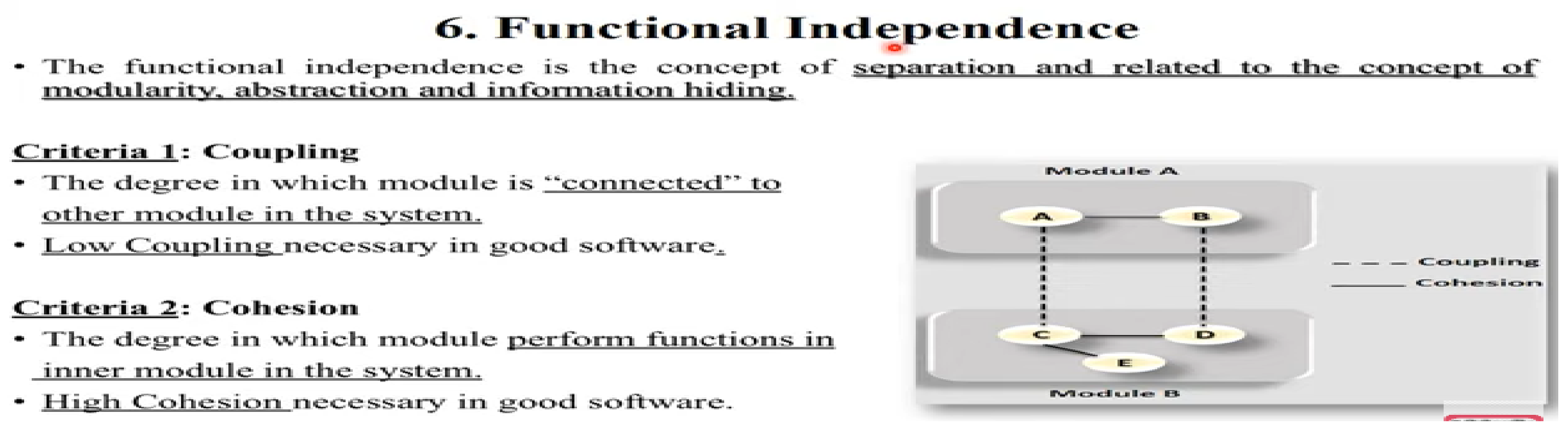
Disadvantages of Modularity

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

Modular design reduces the design complexity and results in easier and faster implementation by allowing parallel development of various parts of a system. We discuss a different section of modular design in detail in this section:

1. 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.



6. Functional Independence

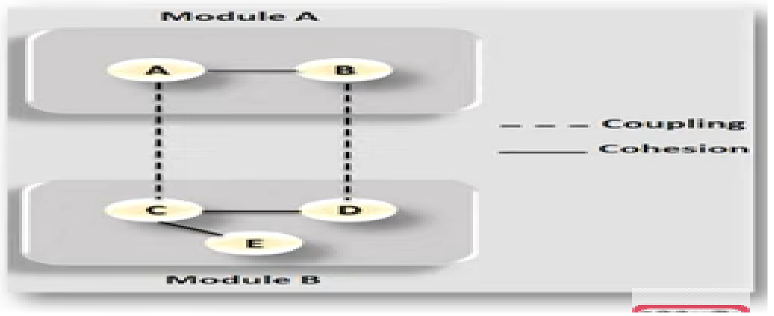
- The functional independence is the concept of separation and related to the concept of modularity, abstraction and information hiding.

Criteria 1: Coupling

- The degree in which module is "connected" to other module in the system.
- Low Coupling necessary in good software.

Criteria 2: Cohesion

- The degree in which module perform functions in inner module in the system.
- High Cohesion necessary in good software.



Information Hiding

2. Information hiding: The fundamental of Information hiding suggests that modules can be characterized by the design decisions that protect from the others, i.e., 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. This is because as most data and procedures are hidden from other parts of the software, inadvertent errors introduced during modifications are less likely to propagate to different locations within the software.

5. Information Hiding

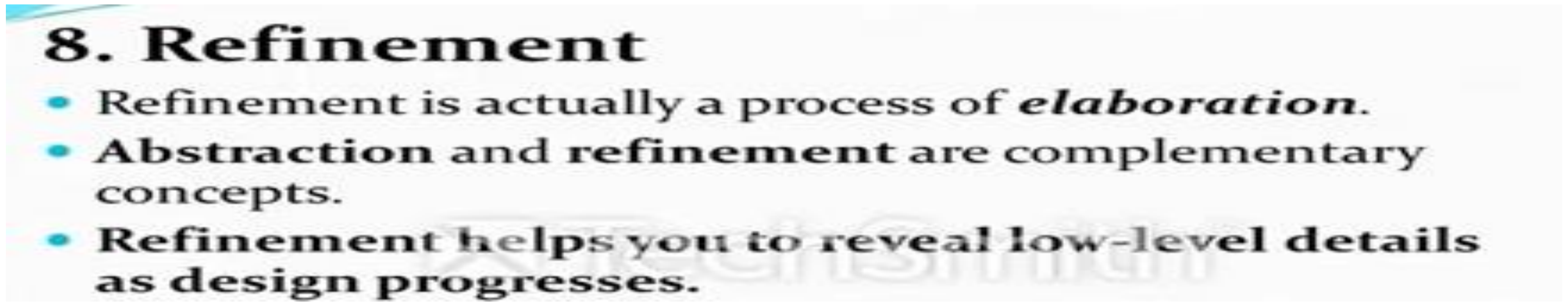
- Modules should be specified and designed in such a way that the data structures and algorithm details of one module are not accessible to other modules.
- They pass only that much information to each other, which is required to accomplish the software functions.
- The way of hiding unnecessary details in modules is referred to as information hiding.

7. Refinement

- Refinement is a <u>top-down design approach.</u>
- It is a <u>process of elaboration.</u>
- A program is established for <u>refining levels of procedural details</u>.
- A hierarchy is established by decomposing a statement of function in a <u>stepwise manner till the programming language statement are reached.</u>

Example:

INPUT	INPUT
Get number 1 (Integer)	Get number 1 (Integer)
Get number 2 (Integer)	Get number 2 (Integer)
PROCESS	While (Invalid Number)
OUTPUT	EXIT

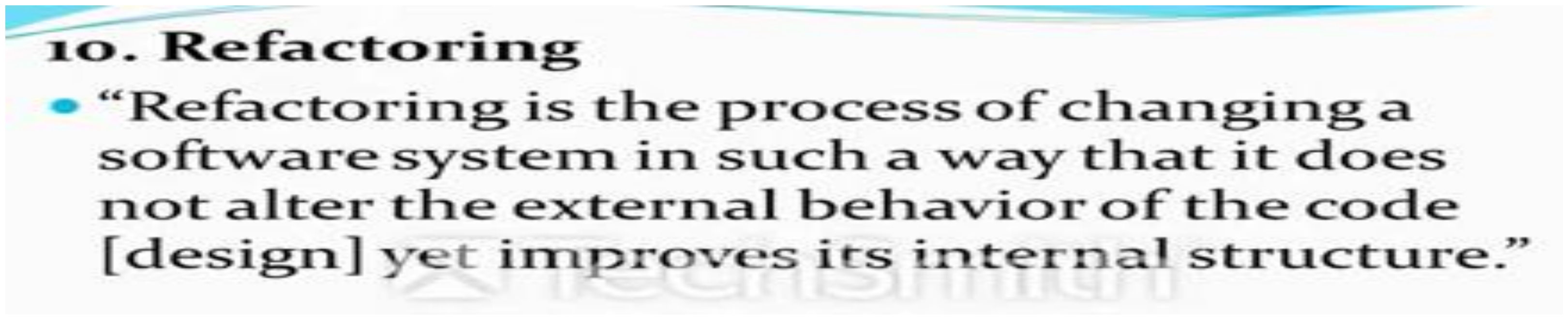


8. Refinement

- Refinement is actually a process of ***elaboration***.
- **Abstraction** and **refinement** are complementary concepts.
- **Refinement helps you to reveal low-level details as design progresses.**

8. Refactoring

- Refactoring is the process of changing the internal software system in a way that it does not change the external behavior of the code still improves its internal structure.
- When software is refactored the existing design is examined for redundancy, unused design elements, unnecessary design algorithms, poorly constructed data, inappropriate data structure or any other design failure that can be corrected for better design.

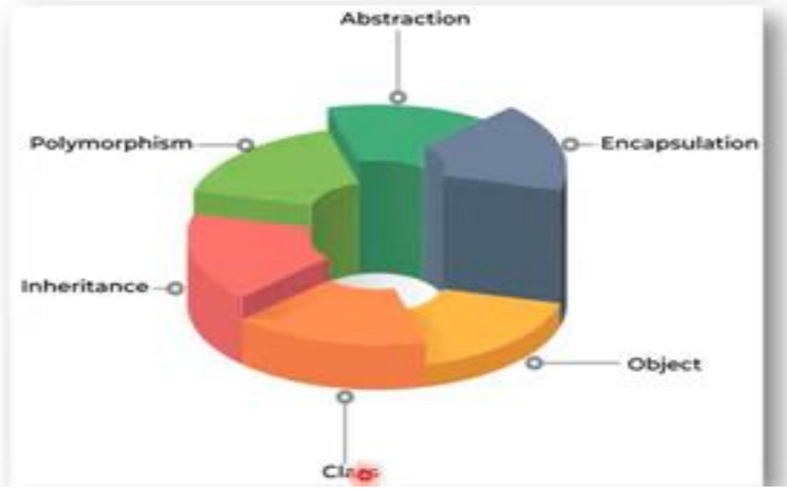


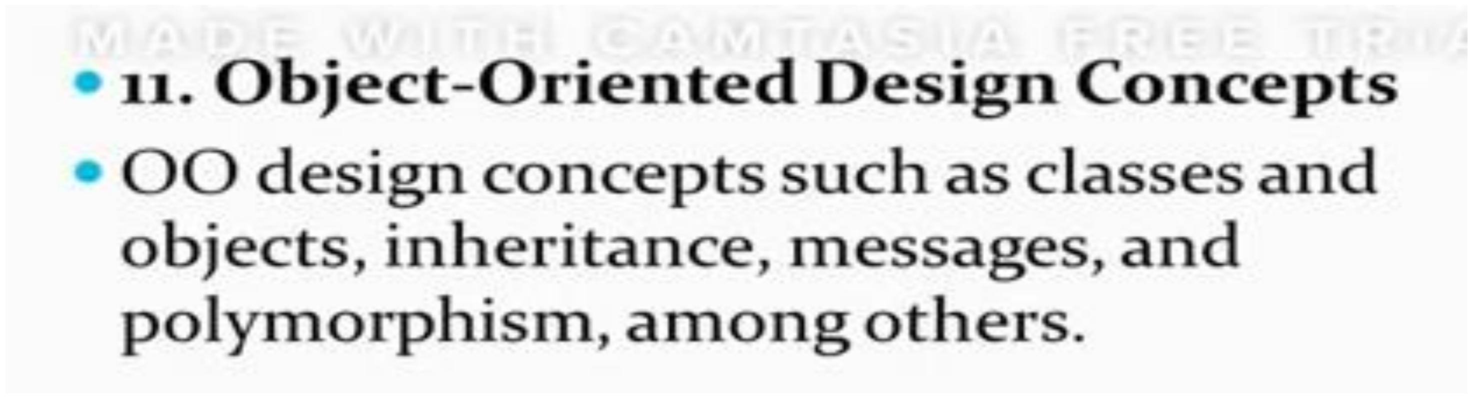
10. Refactoring

- "Refactoring is the process of changing a software system in such a way that it does not alter the external behavior of the code [design] yet improves its internal structure."

9. Object Oriented Design Concepts

- Object Oriented is a popular design approach for analyzing and designing an application.
- Advantage is that faster, low cost development and creates a high quality software.





- 11. Object-Oriented Design Concepts
- OO design concepts such as classes and objects, inheritance, messages, and polymorphism, among others.