POORNIMA
COLLEGE OF ENGINEERING

Affiliated to RTU, Kota • Approved by AICTE & UGC under 2(f) • NAAC A+ Accredited

Assignment-1

Software Engineering (3IT4-07)

Time: 50 min. M.M. 20

Date: 15/10/2023

1.	CO1	PO1	Define the meaning of software design, explain the design fundamentals for software design. (5)
2.	CO1	PO2	What do you mean by effective modular design, explain in detail. (5)
3.	CO2	PO4	Explain the Design Documentation with example. (5)
4.	CO1	PO3	List the program evaluation and explain programming styles. (5)

Solution 1:

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.

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.

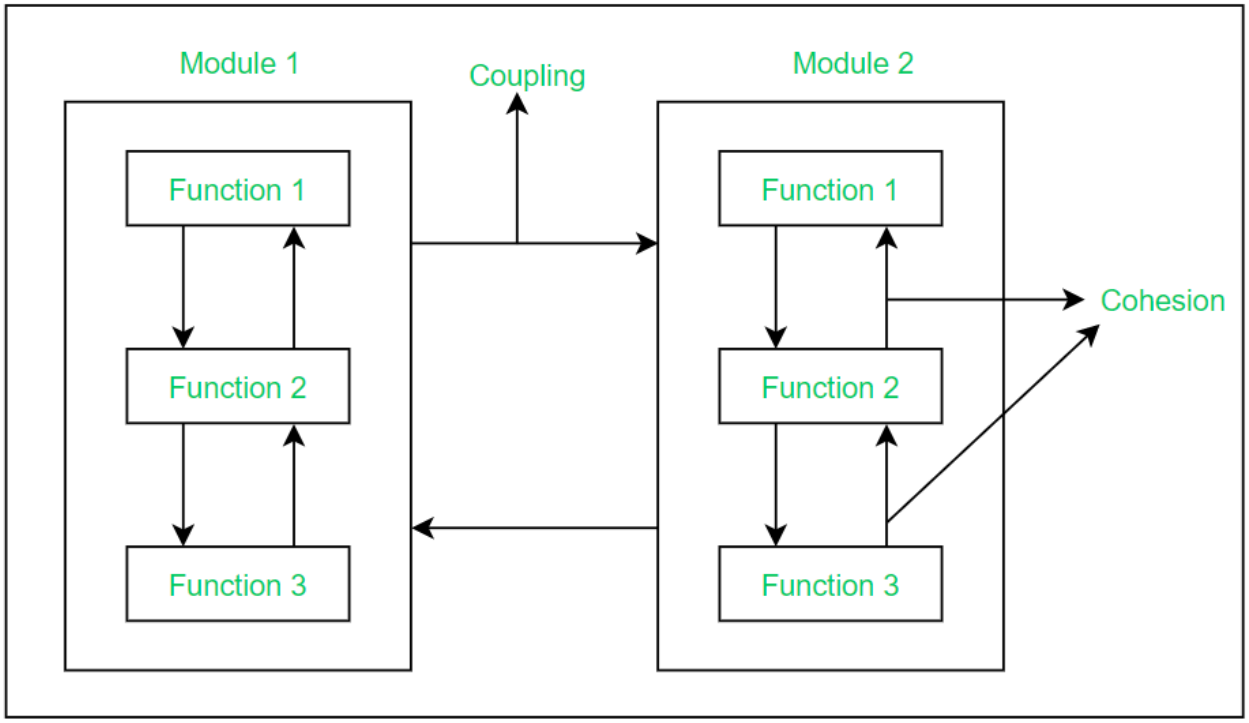
10. **Design is not coding and coding is not design –**
    Design means describing the logic of the program to solve any problem and coding is a type of language that is used for the implementation of a design.

Solution 2:

Any software comprises of many systems which contains several sub-systems and those sub-systems further contains their sub-systems. So, designing a complete system in one go comprising of each and every required functionality is a hectic work and the process can have many errors because of its vast size.

Thus in order to solve this problem the developing team breakdown the complete software into various modules. A module is defined as the unique and addressable components of the software which can be solved and modified independently without disturbing (or affecting in very small amount) other modules of the software. Thus every software design should follow modularity.

The process of breaking down a software into multiple independent modules where each module is developed separately is called **Modularization**.
Effective modular design can be achieved if the partitioned modules are separately solvable, modifiable as well as compilable. Here separate compilable modules means that after making changes in a module there is no need of recompiling the whole software system.



Cohesion:
Cohesion is a measure of strength in relationship between various functions within a module. It is of 7 types which are listed below in the order of high to low cohesion:
1. Functional cohesion
2. Sequential cohesion
3. Communicational cohesion
4. Procedural cohesion
5. Temporal cohesion
6. Logical cohesion
7. Co-incidental cohesion

Coupling:
Coupling is a measure of strength in relationship between various modules within a

software. It is of 6 types which are listed below in the order of low to high coupling:
1. Data Coupling
2. Stamp Coupling
3. Control Coupling
4. External Coupling
5. Common Coupling
6. Content Coupling

Solution 3:

The design phase of software development deals with transforming the customer requirements as described in the SRS documents into a form implementable using a programming language. The software design process can be divided into the following three levels of phases design:

- Interface Design
- Architectural Design
- Detailed Design

Software Design Document:

Software Design Document is a written document that provides a description of a software product in terms of architecture of software with various components with specified functionality.

The design specification addresses different aspects of the design model and is completed as the designer refines his representation of the software.

Importance of Design Documentation:
1**. Requirements are well understood:** With proper documentation, we can remove inconsistencies and conflicts about the requirements. Requirements are well understood by every team member.

2. **Architecture/Design of product:** Architecture/Design documents give us a complete overview of how the product look like and better insight to the customer/user about their product.
3. **New Person can also work on the project:** New person to the project can very easily understand the project through documentations and start working on it. So,

developers need to maintain the documentation and keep upgrading it according to the changes made in the product/software.
4**. Everything is well Stated:** This documentation is helpful to understand each and every working of the product. It explains each and every feature of the product/software.
5**. Proper Communication:** Through documentation, we have good communication with every member who is part of the project/software. Helpful in understanding role and contribution of each and every member.

Solution 4:

Programming style refers to the technique used in writing the source code for a computer program. Most programming styles are designed to help programmers quickly read and understands the program as well as avoid making errors. (Older programming styles also focused on conserving screen space.) A good coding style can overcome the many deficiencies of a first programming language, while poor style can defeat the intent of an excellent language.

The goal of good programming style is to provide understandable, straightforward, elegant code. The programming style used in a various program may be derived from the coding standards or code conventions of a company or other computing organization, as well as the preferences of the actual programmer.

1. Clarity and simplicity of Expression: The programs should be designed in such a manner so that the objectives of the program is clear.

2. Naming: In a program, you are required to name the module, processes, and variable, and so on. Care should be taken that the naming style should not be cryptic and non-representative.

3. Control Constructs: It is desirable that as much as a possible single entry and single exit constructs used.

4. Information hiding: The information secure in the data structures should be hidden from the rest of the system where possible. Information hiding can decrease the coupling between modules and make the system more maintainable.

5. Nesting: Deep nesting of loops and conditions greatly harm the static and dynamic behavior of a program. It also becomes difficult to understand the program logic, so it is desirable to avoid deep nesting.

6. User-defined types: Make heavy use of user-defined data types like enum, class, structure, and union. These data types make your program code easy to write and easy to understand.

7. Module size: The module size should be uniform. The size of the module should not be too big or too small. If the module size is too large, it is not generally functionally cohesive. If the module size is too small, it leads to unnecessary overheads.

8. Module Interface: A module with a complex interface should be carefully examined.

9. Side-effects: When a module is invoked, it sometimes has a side effect of modifying the program state. Such side-effect should be avoided where as possible.