Module: 4
Software Design
Lecture 1

1. Motivation:
Presented chapter is the foundation of basic software engineering and therefore the motivation of this course is to understand the design process involved with software.

2. Learning Objective:
This module explains that people involved in the Software Design Process will be able to understand the Design Process during the design development of the software.

1. Students will be able to understand Software Design – Abstraction , Modularity.
2. Students will be able to understand the Software Architecture – Effective modular design, Cohesion and Coupling, Example of code for cohesion and coupling.
3. Students will be able to understand User Interface Design – Human Factors, Interface standards, Design Issues – User Interface Design.

4. Objective:
This module explains the Software Design Process during the development of the software.

5. Learning Outcomes:

- This module will help students to learn the Software Design Process during the development of the software along with the concepts of Abstraction, Modularity.

- The Students will also be able to develop an effective Software Architecture modular design, Concepts like Cohesion and Coupling for making the design of the software.
- The Students will also be able to create efficient UI designs.

6. Prerequisite:

Workflow of different phases of software life cycle

7. Syllabus:

Prerequisites	Syllabus	Duration	Self Study
Basic design Knowledge	5.1 Software Design – Abstraction , Modularity	4 Hrs	4 Hrs
	5.2 Software Architecture – Effective modular design, Cohesion and Coupling, Example of code for cohesion and coupling.		
	5.3 User Interface Design – Human Factors, Interface standards, Design Issues – User Interface Design Process.	4 Hrs	4 Hrs

Design Workflow

This involves the activities involved in the design phase of software life cycle. System design concepts, architectural styles and design patterns are explained below.

3.3.1 System Design Concepts:

Abstraction

At the highest level of abstraction, a solution is stated in broad terms using the language of the problem environment. At lower levels of abstraction, a more detailed description of the solution is provided.

System design concepts describe subsystem decompositions and their properties. This includes the concept of subsystems and classes, subsystem interfaces, coupling, cohesion.

Modularity

Modularity is the most common manifestation of separation of concerns. Software is divided into separately named and addressable components, sometimes called modules.

Coupling:

Coupling is the number of dependencies between two subsystems. A desirable property of subsystem decomposition is that subsystems are as loosely coupled as reasonable. This minimizes the impact that errors or future changes in one subsystem have on other subsystems.

The modules that interact with each other through message passing have low coupling while those who interact with each other through variables that maintain information about the state have high coupling. The diagram shows examples of two such systems

Interface design

- Using information developed during interface analysis, define interface objects and actions (operations).
- Define events (user actions) that will cause the state of the user interface to change. Model this behavior.
- Depict each interface state as it will actually look to the end user.
- Indicate how the user interprets the state of the system from information provided through the interface.

Design Issues

- **Response time.** System response time is the primary complaint for many interactive applications. system response time is measured from the point at which the user performs some control action (e.g., hits the return key or clicks a mouse) until the software responds with desired output or action.
- **Help facilities**. Almost every user of an interactive, computer-based system requires help now and then. In some cases, a simple question addressed to a knowledgeable colleague can do the trick. In others, detailed research in a multivolume set of "user manuals" may be the only option. In most cases, however, modern software provides online help facilities that enable a user to get a question answered or resolve a problem without leaving the interface.
- **Error handling**. Error messages and warnings impart useless or misleading information and serve only to increase user frustration. There are few computer users who have not encountered an error of the form
- **Menu and command labeling**. The typed command was once the most common mode of interaction between user and system software and was commonly used for applications of every type. Today, the use of window-oriented, point-andpick interfaces has reduced reliance on typed commands,

but some power-users continue to prefer a command-oriented mode of interaction.

- **Application accessibility**. As computing applications become ubiquitous, software engineers must ensure that interface design encompasses mechanisms that enable easy access for those with special needs. Accessibilityfor users (and software engineers) who may be physically challenged is an imperative for ethical, legal, and business reasons
- **Internationalization.** Software engineers and their managers nvariably underestimate the effort and skills required to create user interfaces that accommodate the needs of different locales and languages.

Types of Coupling:

1. Data coupling
2. Stamp coupling
3. Control coupling
4. Hybrid coupling
5. Common coupling
6. Content coupling

Data coupling:

Modules communicate by parameters. Each parameter is an elementary piece of data. Each parameter is necessary to the communication. Nothing extra is needed.

Drawbacks: (i) Too many parameters make the interface difficult to understand and possible error to occur.
(ii) Tramp data - data 'traveling' across modules before being used

Stamp coupling:

A composite data is passed between modules. Internal structure contains data not used. Here, unrelated data are grouped into an artificial structure which is called 'bundling'.

Control coupling:

A module controls the logic of another module through the parameter. The controlling module needs to know how the other module works.

Hybrid coupling:

This is defined as a subset of data used as control.

Common coupling:

Here global data is used as communication between modules.

Drawbacks: (i) Inflexibility
(ii) Difficulty in understanding the use of data

Content coupling:

A module refers to the inside of another module. It can also branch into another module and refers to data within another module. By doing so, a module can change the internal workings of another module.

Cohesion:

Cohesion is the number of dependencies within a subsystem. If a subsystem contains many objects that are related to each other and perform similar tasks, its cohesion is high. If a subsystem contains a number of unrelated objects, its cohesion is low. A desirable property of a subsystem decomposition is that it leads to subsystems with high cohesion.

Strong cohesion implies that all parts of a subsystem should have a close logical relationship with each other. That means, in case some kind of change is required in the software, all the related pieces are found at one place.

A class will be cohesive if most of the methods defined in a class use most of the data members most of the time. If we find different subsets of data within the same class being manipulated by separate groups of functions then the class is not cohesive and should be broken down as shown below.

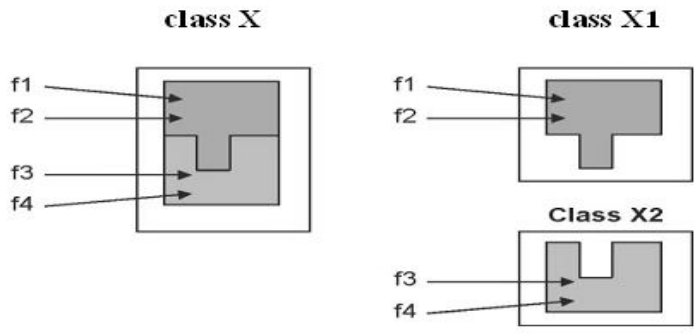

Figure 3.13: Illustration of cohesion

Types of cohesion:

1. Functional cohesion

2. Sequential cohesion
3. Communicational cohesion
4. Procedural cohesion
5. Temporal cohesion
6. Logical cohesion
7. Coincidental cohesion

Functional cohesion:

All elements contribute to the execution of one and only one problem-related task. It has a focused - strong, single-minded purpose. Unrelated activities are not done by the elements.

Sequential cohesion:

Elements are involved in activities such that output data from one activity becomes input data to the next. Usually this cohesion has good coupling and is easily maintained. This is not so readily reusable because activities that will not in general be useful together.

Communicational cohesion:

Elements contribute to activities that use the same input or output data. This cohesion is not flexible, for example, if we need to focus on some activities and not the others. It is possible that links cause activities to affect each other. So it is better to split the elements into functional cohesive ones.

Procedural cohesion:

Elements are related only by sequence, otherwise the activities are unrelated. This is similar to sequential cohesion, except for the fact that elements are unrelated. Commonly found at the top of hierarchy, such as the main program module.

Temporal cohesion:

Elements are involved in activities that are related in time. Commonly found in initialization and termination modules. Elements are basically unrelated, so the module will be difficult to reuse. A good practice is to initialize as late as possible and terminate as early as possible.

Logical cohesion:

Elements contribute to activities of the same general category (type). For example, a report module, display module or I/O module. The elements usually have control coupling, since one of the activities will be selected.

Coincidental cohesion:

Elements contribute to activities with no meaningful relationship to one another (i.e., mixture of activities). Similar to logical cohesion, except the activities may not even be the same type. This is difficult to understand and maintain, with strong possibilities of causing side effects every time the module is modified.

Human Interface

A user interface is a collection of techniques and mechanisms to interact with something. In a graphical interface the primary interaction mechanism is a pointing device of some kind. This device is the electronic equivalent to the human hand. What the user interacts with is a collection of elements referred to as objects. They can be seen, heard, touched, or otherwise perceived. Objects are always visible to the user and are used to perform tasks. They are interacted with as entities independent of all other objects. People perform operations, called actions, on objects. The operations include accessing and modifying objects by pointing, selecting, and manipulating. All objects have standard resulting behaviors.

Common usability problems

- Ambiguous menus and icons.
- Languages that permit only single direction movement through a system.
- Input and direct manipulation limits.
- Complex linkage.
- Inadequate feedback.
- Lack of system anticipation.
- Inadequate error messages.

Important Human Characteristics in Design

Importance in design are perception, memory, visual acuity, foveal and peripheral vision, sensory storage, information processing, learning, skill, and individual differences.

- Memory: Memory is not the most stable of human attributes, as anyone who has forgotten why they walked into a room, or forgotten a very important birthday, can attest.

  - *Short-term,* or working, memory.
  - *Long-term* memory
  - Mighty memory
  - Sensory Storage

Mental Models: As a result of our experiences and culture, we develop mental models of things and people we interact with

Movement Control: Once data has been perceived and an appropriate action decided upon, a response must be made;

Learning: Learning, as has been said, is the process of encoding in long-term memory information that is contained in short-term memory.

Skill: The goal of human performance is to perform skillfully. To do so requires linking inputs and responses into a sequence of action.

Individual Differences : In reality, there is no average user. A complicating but very advantageous human characteristic is that we all differ-in looks, feelings, motor abilities, intellectual abilities, learning abilities and speed, and so on

Questions:

1. What is Modularity?
2. What is Abstraction?
3. Explain different types of Coupling.
4. What is Cohesion? Explain different types of Cohesion
5. What are the design issues in Human interface design?