SNS COLLEGE OF TECHNOLOGY

Coimbatore-37.
An Autonomous Institution

COURSE NAME : 19CST201-Agile Software Engineering

II YEAR/ III SEMESTER

Topic: Design Concepts

Ms.G.Swathi
Assistant Professor
Department of Computer Science and Engineering

Introduction -Types

In software development, there are many stages of planning and analysis before the project is finalized and development can formally begin. Design always comes before development, and functional design makes coding and maintenance very simple.

There are seven main principles to keep in mind in the design model in object-oriented programming (OOP):

- Abstraction
- Patterns
- Separation of data
- Modularity
- Data hiding
- Functional independence
- Refactoring

Abstraction

- In abstraction, is the process of hiding complex properties or characteristics from the software itself to keep things more simplistic.
- This allows for a much higher level of efficiency for complex software designs since it allows the developers to list out only the necessary elements or objects required.
- In this principle, the developer will define the properties, type of functions, and the interface for each of said objects in the project.
- The developers will be able to hide the complicated and unnecessary details in the background while retaining core information in the foreground.

Patterns

- **Architectural**, which is a high-level pattern type that can be defined as the overall formation and organization of the software system itself.
- **Design**, which is a medium-level pattern type that is used by the developers to solve problems in the design stage of development. It can also affect how objects or components interact with one another.
- And, finally, **idioms**, which are low-level pattern types, often known as coding patterns, and are used as a workaround means of setting up and defining how components will be interacting with the software itself without being dependent on the programming language.

Separation of Data & Modularity

- This principle states that the software code must be separated into two sections called layers and components.
- To ensure proper implementation, the two sections must have little to no overlap between them and must have a defined purpose for each component.
- This principle allows each component to be developed, maintained, and reused independently of one another.
- **Modularity**, on the other hand, refers to the idea of using predetermined code to increase overall efficiency and management of the current project.
- The software components will usually be divided into unique items known as modules.
- Their specific functions divide these modules. Modularity makes the systems easy to manage.

Hiding Independence & Refactoring

- Also known as information hiding, data hiding allows modules to pass only the required information between themselves without sharing the internal structures and processing.
- The specific purpose of hiding the internal details of individual objects has several benefits.

Refactoring is the process of changing a software system in such a way that it does not alter the function of the code yet improves its internal structure.

References

- Lisa Crispin, Janet Gregory, “Agile Testing; A Practical Guide for Testers and Agile Teams”, Addison Wesley, 3rd Edition, 2015. 1
- Robert C.Martin, “ Agile Software Development, Principles, Patterns and Practices”, Prentice Hall, 2nd Edition, 2014.
- Alistair Cockburn, “Agile Software Development: The Cooperative Game”, Addison Wesley, 2nd Edition,2015.
- Mike Cohn, “User Stories Applied: for Agile Software”, Addison Wesley, 2nd Edition,2015.


Thank
you