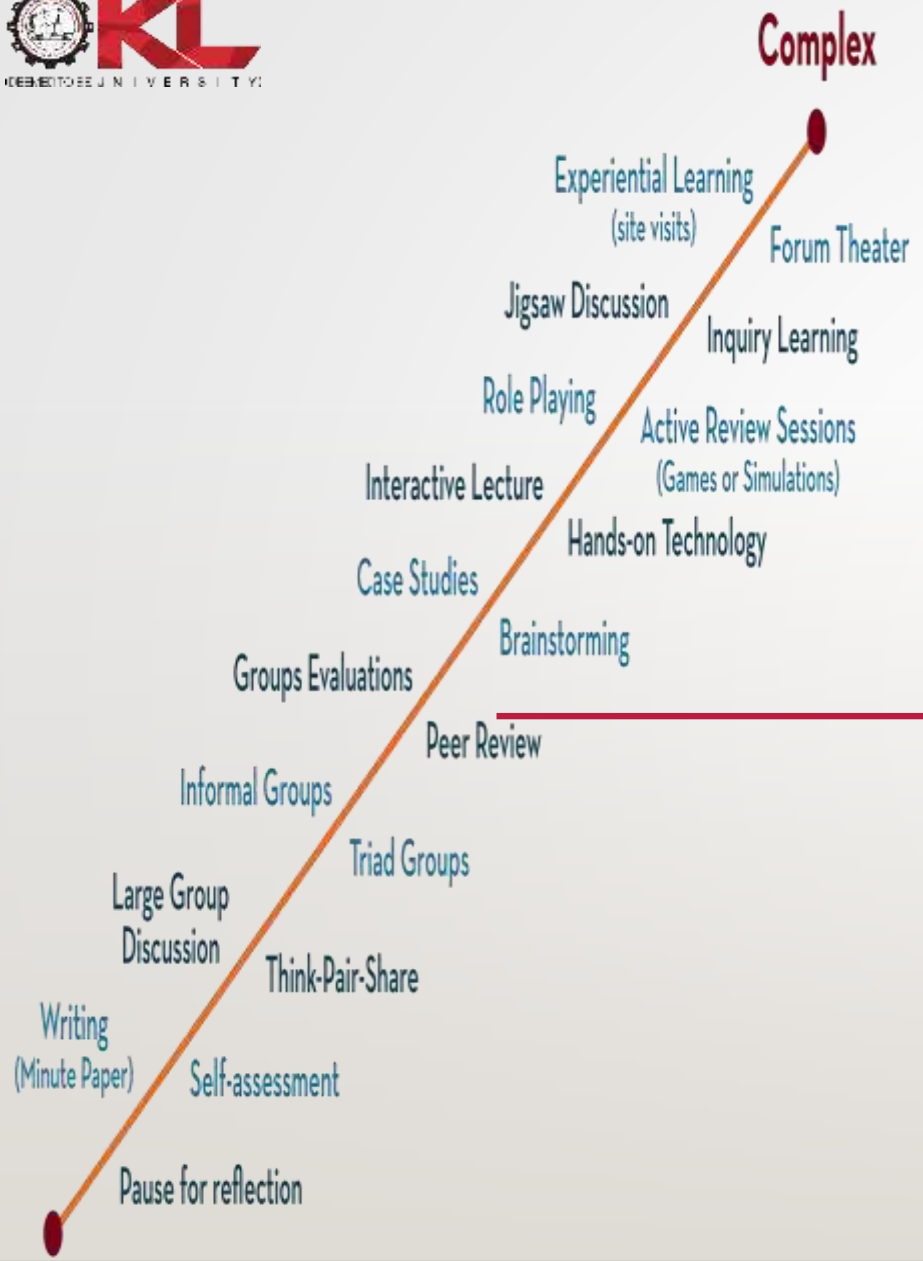

DEPARTMENT OF CSE , CSIT & AI&DS

COURSE NAME : ADAPTIVE SOFTWARE ENGINEERING

COURSE CODE : 23CI2001

Topic:

DESIGN PATTERNS

Session - 17

AIM OF THE SESSION

To familiarize students with the concept of **DESIGN PATTERNS**

INSTRUCTIONAL OBJECTIVES

This Session is designed to:

1. Describe and Demonstrate **DESIGN PATTERNS**

LEARNING OUTCOMES

At the end of this session, you would be able to:

1. Know and understand **DESIGN PATTERNS**

AGENDA

- Introduction
- Design pattern characterization
- Kinds of Patterns
- Frameworks
- describing a Pattern
- Pattern Languages and Repositories

INTRODUCTION

- A design pattern can be characterized as "a three-part rule which expresses a relation between a certain context, a problem, and a solution" .
- For software design, context allows the reader to understand the environment in which the problem resides and what solution might be appropriate within that environment.
- A set of requirements, including limitations and constraints, acts as a system of forces that influences how the problem can be interpreted within its context and how the solution can be effectively applied.
- It is reasonable to argue that most problems have multiple solutions, but that a solution is effective only if it is appropriate within the context of the existing problem. Finally, every solution has consequences that may have an impact on other aspects of the software and may themselves become part of the system of forces for other problems to be solved within the larger system.

DESIGN PATTERN CHARACTERIZATION

characterizes an effective design pattern in the following way:

- **It solves a problem**: Patterns capture solutions, not just abstract principles or strategies.
- **It is a proven concept**: Patterns capture solutions with a track record, not theories or speculation.
- **The solution isn't obvious**: Many problem-solving techniques (such as software design paradigms or methods) try to derive solutions from first principles. The best patterns generate a solution to a problem indirectly—a necessary approach for the most difficult problems of design.
- **It describes a relationship**: Patterns don't just describe modules, but describe deeper system structures and mechanisms.
- The pattern has a significant human component (minimize human intervention).

All software serves human comfort or quality of life; the best patterns explicitly appeal to aesthetics and utility.

KINDS OF PATTERNS

- **Architectural patterns** describe broad-based design problems that are solved using a structural approach.
- **Data patterns** describe recurring data-oriented problems and the data modelling solutions that can be used to solve them.
- **Component patterns** (also referred to as design patterns) address problems associated with the development of subsystems and components, the manner in which they communicate with one another, and their placement within a larger architecture.
- **Interface design patterns** describe common user interface problems and their solution with a system of forces that includes the specific characteristics of end users.
- **WebApp patterns** address a problem set that is encountered when building WebApps and often incorporates many of the other patterns categories just mentioned. At a lower level of abstraction, idioms describe how to implement all or part of a specific algorithm or data structure for a software component within the context of a specific programming language.

KINDS OF PATTERNS CONT…

- focus on **three** types of patterns that are particularly relevant to object-oriented design:

1. **creational patterns** focus on the "creation, composition, and representation" of objects.it provides mechanisms that make the instantiation of objects easier within a system and enforce "constraints on the type and number of objects that can be created within a system"
2. **Structural patterns** focus on problems and solutions associated with how classes and objects are organized and integrated to build a larger structure. In essence, they help to establish relationships between entities within a system.
3. **Behavioral patterns** address problems associated with the assignment of responsibility between objects and the manner in which communication is effected between objects.

FRAMEWORKS

- Patterns themselves may not be sufficient to develop a complete design. In some cases it may be necessary to provide an implementation-specific skeletal infrastructure, called a framework, for design work
- A framework is not an architectural pattern, but rather a skeleton with a collection of "plug points" (also called hooks and slots) that enable it to be adapted to a specific problem domain.
- In an object-oriented context, a frame-work is a collection of cooperating classes.

FRAMEWORKS CONT…

The differences between design patterns and frameworks in the following manner:

- 1. **Design patterns are more abstract than frameworks**. Frameworks can be embodied in code, but only examples of patterns can be embodied in code. A strength of frameworks is that they can be written down in programming languages and not only studied but executed and reused directly. . .
- 2. **Design patterns are smaller architectural elements than frameworks**. A typical framework contains several design patterns but the reverse is never true.
- 3. **Design patterns are less specialized than frameworks**. Frameworks always have a particular application domain. In contrast, design patterns can be used in nearly any kind of application. While more specialized design patterns are certainly possible, even these wouldn't dictate an application architecture.

DESCRIBING A PATTERN

- Pattern-based design begins with the recognition of patterns within the application you intend to build, continues with a search to determine whether others have addressed the pattern, and concludes with the application of an appropriate pattern to the problem at hand. The second of these three tasks is often the most difficult.
- How do you find patterns that fit your needs?

- the context in which the pattern resides, the system of forces that mold the context, and the solution that is proposed. To communicate this information unambiguously, a standard form or template for pattern descriptions is required.

- A pattern template provides a standardized means for describing a design pattern. Each of the template entries represents characteristics of the design pattern that can be searched (e.g., via a database) so that the appropriate pattern can be found.

PATTERN LANGUAGES AND REPOSITORIES

- A pattern language encompasses a collection of patterns, each described using a standardized template and interrelated to show how these patterns collaborate to solve problems across an application domain.
- in a pattern language, design patterns are organized in a way that provides a "structured method of describing good design practices within a particular domain."
- Dozens of pattern languages have been proposed for software design . In most cases, the design patterns that are part of pattern language are stored in a Web accessible patterns repository .
- The repository provides an index of all design patterns and contains hypermedia links that enable the user to understand the collaborations between patterns.

SELF-ASSESSMENT QUESTIONS

1. Define design Patterns?
2. Name the different kinds of design Patterns and explain them?
3. Explain about the frameworks of Design Patterns?

Reference Books:

TEXTBOOKS:

1. Roger S.Pressman, "Software Engineering – A Practitioner's Approach" 7th Edition, Mc Graw Hill,(2014).
2. Ian Sommerville, "Software Engineering", Tenth Edition, Pearson Education, (2015).
3. Agile Software Development Ecosystems, Jim Highsmith, Addison Wesley; ISBN: 0201760436; 1st edition

Reference Book

Agile Modelling: Effective Practices for Extreme Programming and the Unified Process Scott Amber John Wiley & Sons; ISBN: 0471202827; 1st edition.

WEB REFERNCES/MOOCS:

https://www.digite.com/kanban/what-is-kanban/
http://www.scaledagileframework.com
https://www.guru99.com/test-driven-development.html
https://junit.org/junit5/

THANK YOU

(DEEMED TO BE UNIVERSITY)

Team – Adaptive Software Engineering