CS 487 – SOFTWARE ENGINEERING

Week 5 Engagement – Modeling and Architecture

Suhas Palani **A20548277** spalani3@hawk.iit.edu

Software modeling and architecture are crucial activities of the design phase, connecting the bridge between analysis and implementation. As the instructor reminded, "**a picture is worth a thousand words**" when it comes to the design of software systems. Models give graphical representations of much higher power than words alone can have in expressing system structure and behavior.

Key types of models discussed include:

1. **Context models** - Define system boundaries and interfaces
2. **Interaction models** - Capture user inputs/outputs and component interactions (e.g. use case diagrams, sequence diagrams)
3. **Structural models** - Show system components and relationships (e.g. class diagrams)
4. **Behavioral models** - Illustrate system states and transitions (e.g. state machines)

This lecture indeed put a strong emphasis on how modelling contributes to lessening the degree of abstraction from the "real world" and, therefore, makes the designs intuitive and reusable. Object-oriented approaches were found particularly apt regarding modelling real-world entities.

Architectural patterns and styles are the building blocks of any software system, as they define its structure and behaviour. As much as they provide established solutions to common design problems, it must be recognised that no single pattern or style can be used universally. In all projects, software architects must carefully consider the specific demands, constraints, and quality attributes of the project at hand, to make informed selections and adaptations of suitable architectural approaches. That is, this often requires combining several patterns or even engineering hybrid architectures to solve complex system needs. More importantly, architects must bear in mind the long-term evolution of the system. We need to ensure that the selected architecture will allow us to support changes, and scalability needs in the future.

Architecture was said to provide a "**solid foundation**" and "**structure that meets basic needs**" for software systems. Key architectural decisions include:

- Reusing generic application architectures
- Distributing the system across processors
- Selecting appropriate architectural styles/patterns
- Decomposing structural elements into modules
- Controlling operation of system units

Common architectural patterns covered included layered, repository, client-server, and pipe-and-filter. The trade-offs between centralized vs. event-based control styles were also discussed.

For distributed systems, the lectures examined benefits like resource sharing and fault tolerance, as well as challenges like complexity and security. Patterns like master-slave and multi-tier client-server were presented as solutions for different distributed scenarios.

The use of design patterns had been discussed as "solutions to common problems". Patterns, like an Observer pattern, provided repeatable and tried-and-true approaches, therefore saving years of labour.

Overall, the lecture explained how good modelling and architecture serve not only to bridge analysis and implementation but also to communicate to stakeholders and address key quality attributes such as performance, security, and maintainability. The iterative nature of design was promoted, and prototyping was introduced to both validate models and expedite development.