**LOUIE JAY A. CENIZA MARCH 21, 2024**

**SYSTEMS (CHAPTER 11)**

**How Would You like To Build a City?.** What I’ve understand is that when building a clean code for your codebase system, imagine it like building a city where there is always a team who manage it that has different work in that particular city or a project in our system.

**Separate Constructing a System from Using It.** In this lesson it talks about the importance of separating the construction process of a software system from its runtime logic, and various techniques to achieve this separation effectively to manage the object construction and dependency wiring in software systems to improve modularity, maintainability, and testability.

**Scaling Up.** In this lesson it compares building software systems to building cities. Just like cities start small and grow over time, software systems begin with basic features and gradually expand to meet new needs.

**Java Proxies.** Java proxies offer a way to dynamically alter the behavior of objects at runtime, they may not be suitable for all use cases due to their complexity and limitations, particularly when dealing with more advanced scenarios requiring broader interception capabilities.

**Pure Java AOP Frameworks.** Using pure Java AOP frameworks allows developers to focus on business logic while seamlessly incorporating cross-cutting concerns, resulting in cleaner, more maintainable codebases compared to traditional approaches like EJB2.

**Test Drive the System Architecture.** It talks about the importance of separating concerns in software development, advocating for test-driven development, avoiding excessive upfront design, maintaining flexibility, and striving for efficient, adaptable designs to deliver value effectively and respond to changing needs.

**Optimize Decision Making.** The benefits of modularity and decentralized decision-making in both large-scale systems and software projects. It emphasizes the importance of assigning responsibilities to qualified individuals and delaying decisions until necessary, allowing for informed choices based on optimal information and maximizing customer feedback, reflection, and experience with implementation choices.

**Use Standards Wisely, When They Add Demonstrable Value.** The importance of using standards judiciously in software development, drawing parallels with the construction industry. While standards can facilitate reusability, recruitment, encapsulation of ideas, and component integration, blindly adhering to them without considering their actual value to customers can lead to inefficiencies and loss of focus. It warns against becoming overly fixated on trendy or heavily promoted standards, emphasizing the need for practicality and alignment with real needs rather than just conformity for the sake of it.

**Systems Need Domain-Specific Languages.** The importance of Domain-Specific Languages (DSLs) in software development, drawing parallels with the rich language developed in building construction. DSLs, which are separate scripting languages or APIs tailored to specific domains, bridge the communication gap between domain concepts and their implementation in code. By allowing developers to write code that closely resembles the language of domain experts, DSLs minimize the risk of misinterpretation and enable clearer expression of intent at various levels of abstraction within the application.