Lecture_8

Input : requirements specification
Output : architecture and design documents

Arcitecture - Big picture

- Architectural patterns
  - Subsystems
  - Their interaction principals
- The allocation of modules into subsystems

Plan Based Development

Step	Name
1	Requirements
2	Arcitecture
3	Design
4	Implementation

> *Agile is the same process (Object Oriented Design)*

Software Architecture

- The critical link between design and requirements engineering, as it identifies the main structural components in a system and the relationship between them
- The architectural view of a system is useful for communication within the team

Box and Line Diagrams

- criticized because they lack semantics and do not show the types of relationships between entities nor the visible properties of entities
- Depends on the use of architectural models. The requirements depend on how they are used.

What's included in Architecture

- Major subsystems and modules
- Interfaces and communication protocols
- Database and data structures
- Security mechanisms
- Fault Tolerance

Designing Subsystems / Modules

- May be broken up into smaller systems
- Easier to implement a number of smaller subsystems
- If the subsystems are independent they can be implemented by programming teams working in parallel
- It does not make sense to break up small systems
- Every module should have a single responsibility

Loose coupling principle

- Keep the connection between parts
- Cohesion - The degree to which the elements of a module belong together (related code should be close to each other)
- Coupling - The degree to which the different modules depend on each other (modules should be independent as much as possible)

Minimum Complexity Principle

- The simple solution over the clever solution
- Easier to understand, test, maintain, replace
- Similar to the Principle of Least Knowledge - One should not know about the internal details of other modules
- Dont Repeat Yourself - Do not duplicate functionality
- Kiss (Keep it simple stupid) - make it simple only focus on what's needed

Layered Architecture

- Providing clean separation between layers is sometimes difficult and a high-level layer may need to interact with a layer below it. Performance may suffer because of this

MVC

- Model View Controller
- The model manages the system data and associated operations on that data
- The view defines and manages how the data is presented to the user
- The controller manages user interaction and passes these interactions to the View and the Model

Type	Example
Controller	HTTP request processing application, data validation
View	Dynamic Page Generation, forms management
Model	Business logic Database

- The pattern is commonly used where there are multiple ways to view and interact with data
- Also used when the future requirements for interaction and presentation of the data are unknown
- Can involve additional code and code complexity when the data model and interactions are simple

Model View View Model

- The model manages the system data and associated operations on that data
- The view defines and manages how the data is displayed and reacted to the user
- View model sits between these two providing binding behavior