Architectural Structures

SEN 522 Advanced Software System Architecture

(Fall 2020)

Dr Awais Majeed

Department of Software Engineering
Bahria University, Islamabad

Topics Covered

Architectural Structures

- Module Structure
- Component & Connector Structure
- Allocation Structure

Module Structure

- The elements are modules – *units of implementation (Code or data units)*
- Modules are assigned areas of functional responsibilities
- A static (logical) view of a system.
- Module structure focuses on:
  - What is the primary functional responsibility assigned to each module?
  - What other software elements is a module allowed to use?
  - What other softwares does it actually use and depend on?
  - What modules are related to other modules by generalisation or specialisation relationships?

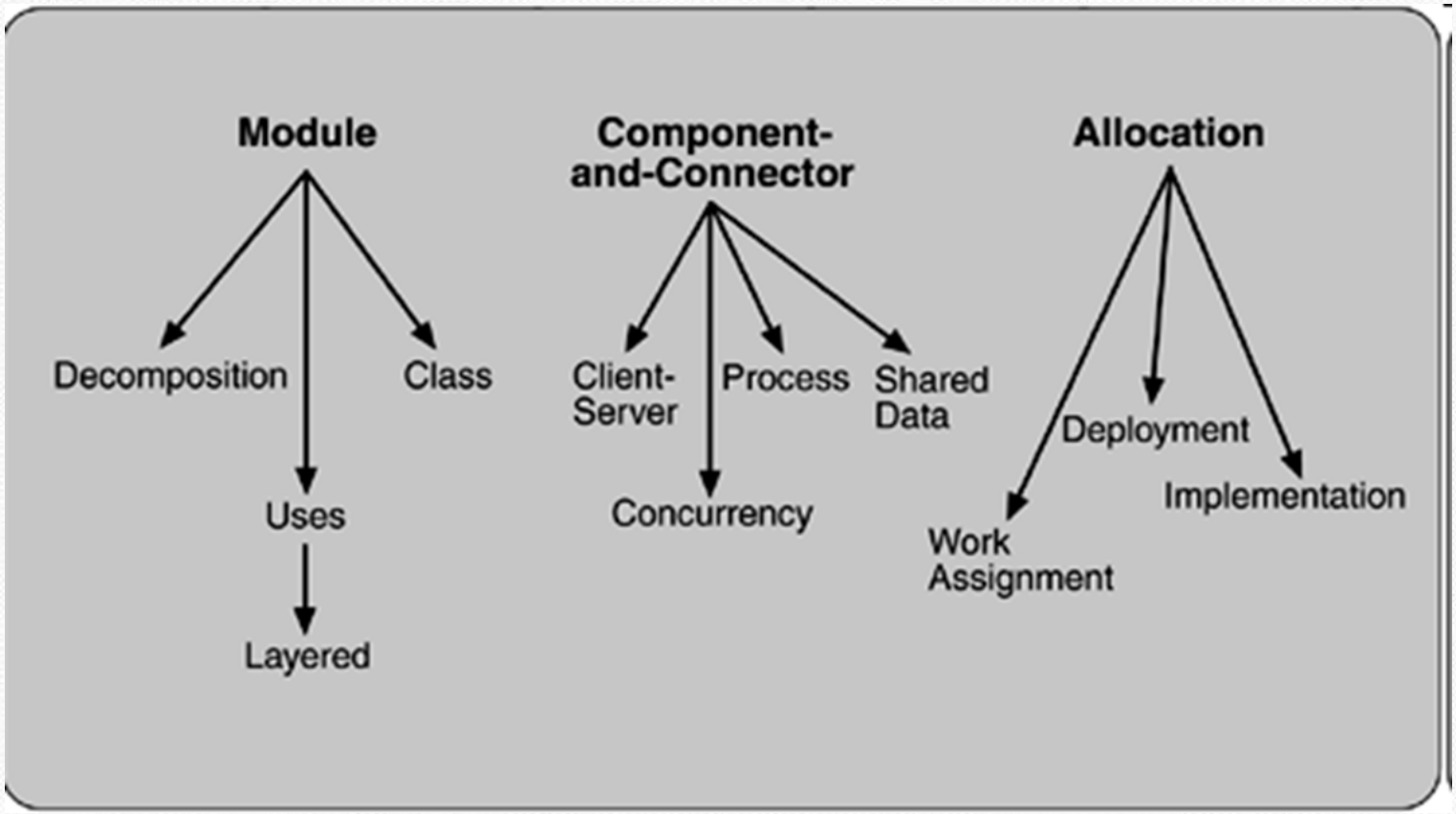
Component & Connector Structure

- **Elements** are runtime components (which are the principal units of computation) and
- **Connectors** (which are the communication vehicles among components)
- C&C structure focuses on:
  - What are the major executing components and how do they interact?
  - What are the major shared data stores?
  - Which parts of the system are replicated?
  - How does data progress through the system?
  - What parts of the system can run in parallel?
  - How can the system's structure change as it executes?

Allocation Structure

- Allocation structures show the relationship between the software elements and the elements in one or more external environments
- Allocation structure's focus is on:
  - What processor does each software element execute on?
  - In what files is each element stored during development, testing, and system building?
  - What is the assignment of software elements to development teams?

Summary of Architectural Structures



Decomposition Structure

- Relationship between modules is “is a sub-module of”
- The decomposition structure enhances system‘s modifiability – changes fall in a small number of modules
- The modules have associated products (interface specifications, code, test plans etc.)

Uses Structure

- The units are modules or procedures or resources on the interfaces of modules.
- One unit uses another if the correctness of the first requires the presence of a correct version of the second.
- Extend or abstract (extract) functionality

Layered Structure

- A system of layers emerges when *uses* relationship is controlled in a particular fashion
  - in which a layer is a coherent set of related functionality.
- In strict sense : layer n may only use the services of layer $n-1$.
- Layers are often designed as abstractions (virtual machines) that hide implementation specifics below from the layers above $\rightarrow$?

Example

Think of the most common example !!

Class or Generalisation

- The module units in this structure are called classes
- The relation is "**inherits-from**" or "**is-an-instance-of**"
- This view supports reasoning about collections of similar behavior or capability

Data Model

- It describes the static information structure in terms of data entities and their relationships.
- For example: in a banking system entities can be Account, Customer, and Loan
- Relationships can dictate that a customer can have one or more accounts and each account must be associated with atleast 1 customer.

Process or Communicating Processes

- Deals with the dynamic aspects of a running system
- The units here are processes or threads that are connected with each other by communication, synchronization, and/or exclusion operations.
- The relationship is of attachment, showing how the components and connectors are hooked together
- It is important to design a system's execution performance and availability.

Concurrency

- A structure which allows the architect to determine opportunities for parallelism and the locations where resource contention may occur
- The units are components and the connectors are "logical threads."
- A logical thread is a sequence of computation that can be allocated to a separate physical thread later in the design process.

Shared Data or Repository

- Components and connectors that create, store, and access persistent data

Client-Server Structure

- A structure of components and connectors for a system built as a group of cooperating clients and servers
- The components are the clients and servers
- the connectors are protocols and messages they share to carry out the system's work
- Useful for:
  - Separation of concerns
  - Physical distribution, and
  - Load balancing (supporting runtime performance).

Service structure

- The units here are services that interoperate with each other by service coordination mechanisms such as SOAP.
- It is important for designing a Service Oriented Architecture (SOA) where a system is composed of components that may have been developed anonymously and independently of each other.

Deployment Structure

- The deployment structure shows how software is assigned to *hardware-processing* and *communication elements*
- The elements are software (usually a process from a component-and-connector view), hardware entities (processors), and communication pathways.
- Relations are "allocated-to" →which physical units the software elements reside,
- "migrates-to" →if the allocation is dynamic

Implementation

- This structure shows how software elements (usually modules) are mapped to the file structure(s) in:
  - system's development,
  - integration, or
  - configuration control environments.
- Helpful in the management of development activities and build environment

Work Assignment

- This structure assigns responsibility for implementing and integrating the modules to the appropriate development teams
- Work assignment structure has architectural as well as management implications

Which structure to choose?

- Not all of them may be relevant
- one of the obligations of the architect is to understand how the various structures lead to quality attributes

References & Further reading

- Chapter 1 of "Software Architecture and Practice" 3rd Ed by Len Bass