Architectural Structures and Views

Software Architecture
3rd Year – Semester 1
Lecture 8

Architectural Structures and Views

- View

  A view is a representation of a coherent set of architectural elements, as written by and read by system stakeholders. It consists of a representation of a set of elements and the relations among them

- Structure

  The set of elements itself, as they exist in software or hardware

Purpose:

- Restrict our attention at any one moment to one (or a small number) of the software system's structures.
- To communicate meaningfully about an architecture, we must make clear which structure or structures we are discussing at the moment

Example:
Software Module - View Vs. Structure

- View

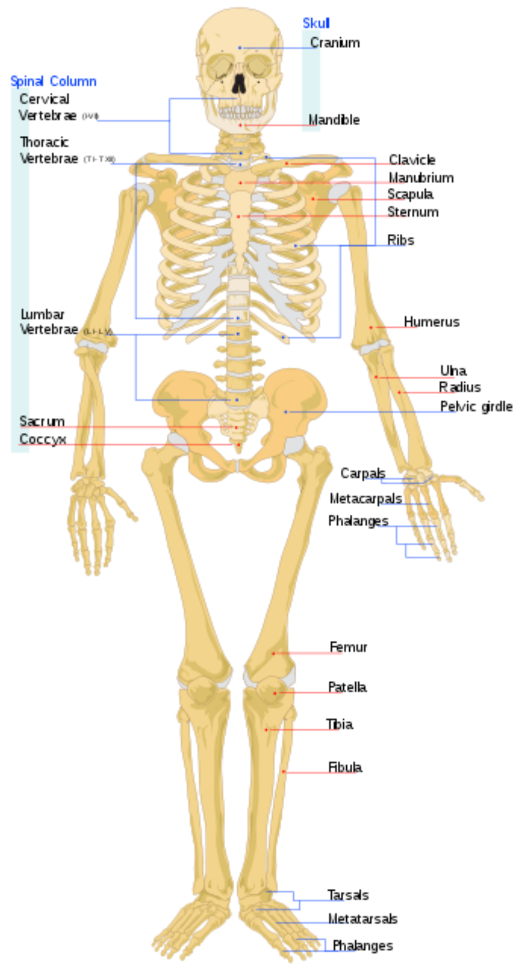
  A module view is the representation of that structure, as documented by and used by some system stakeholders

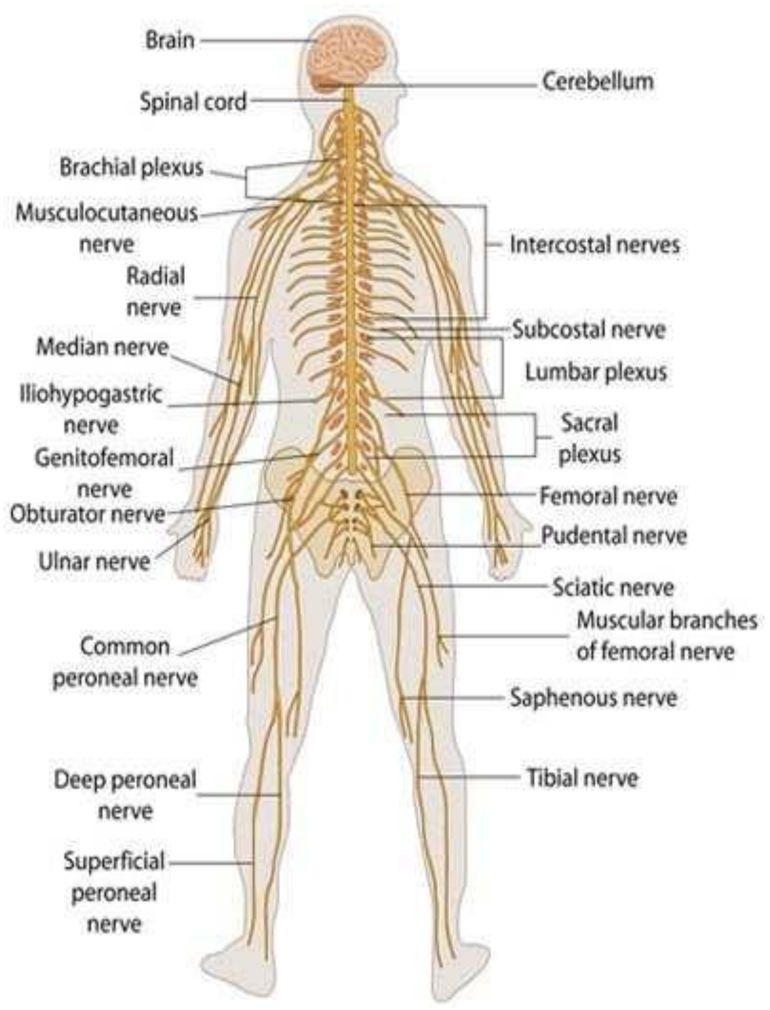
- Structure

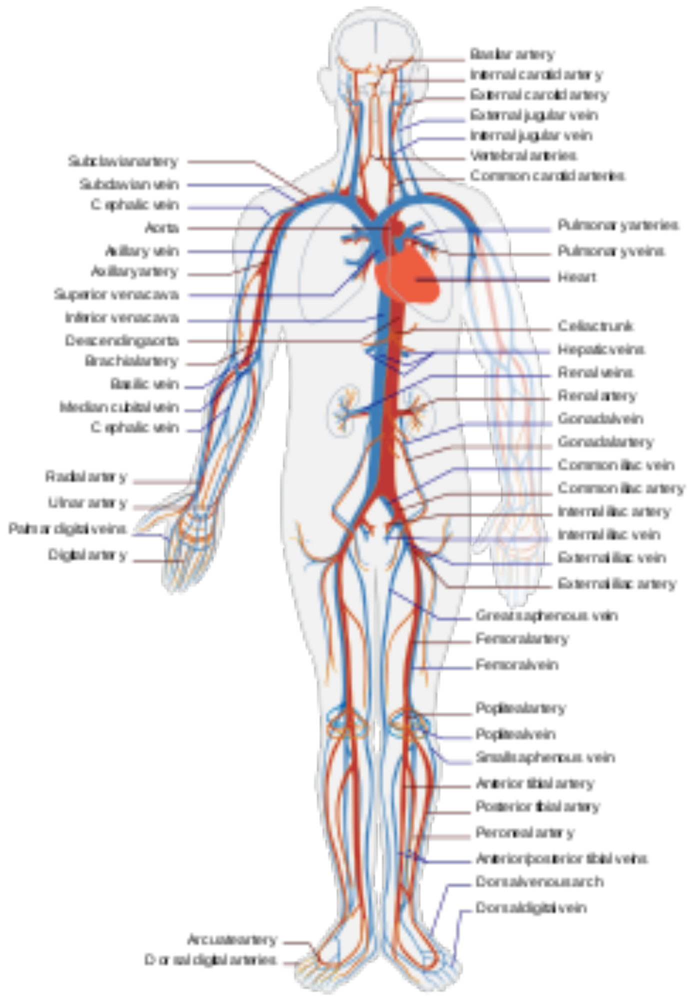
  A module structure is the set of the system's modules and their organization

Note: These terms are often used interchangeably, but we will adhere to these definitions

An example from elsewhere…





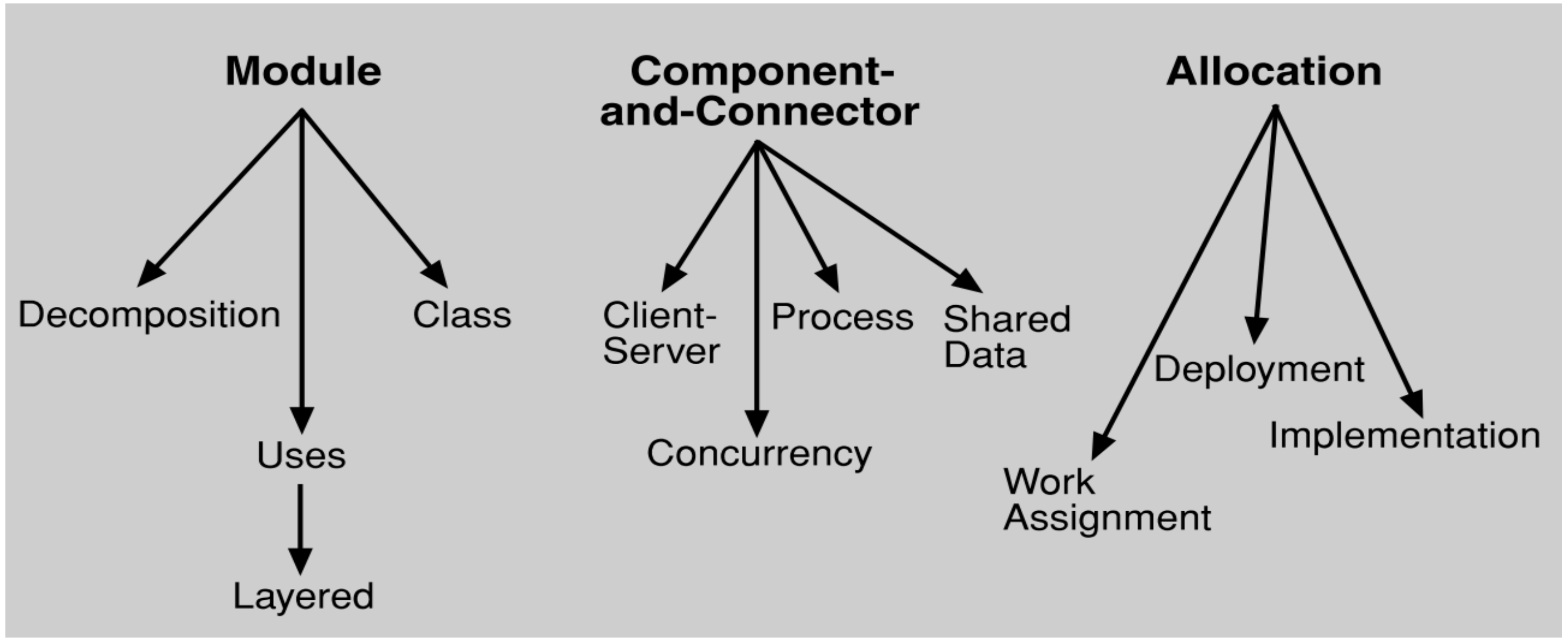


Architectural Considerations

- How is the system to be structured as a set of <u>code units</u>?
  - Modules
- How is the system to be structured as a set of elements that have runtime <u>behavior</u> and <u>interactions</u>?
  - Components
  - Connectors
- How is the system to <u>relates to</u> non-software structures in its environment?
  - CPU, File System, Network
  - Development Team (non-software)

Software Structures

- Module Structures
- Component & Connector Structures
- Allocation Structures



Module Structures

Elements are modules - which are <u>units of implementation</u>

- What is the primary functional responsibility assigned to each module?
- What other software elements is a module allowed to use?
- What other software does it actually use?

- **Decomposition:** Shows how larger modules are decomposed into smaller ones recursively
- **Uses:** The units are; modules, procedures or resources on the interfaces of modules. The units are related by the uses relation
- **Layers:** Structured into layers by using relations
- **Class (Generalization):** Shows the inherits-from or is-an-instance-of relations among the modules

Component-and-connector Structures

Elements are runtime components and connectors. The relation is attachment, showing how the components and connectors are linked together

- 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?

- **Process (or communicating processes):** Units are processes or threads that are connected with each other by communication, synchronization, and/or exclusion operations
- **Concurrency:** 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
- **Shared Data (or repository):** This structure comprises components and connectors that create, store, and access persistent data
- **Client-Server:** The components are the clients and servers, and the connectors are protocols and messages

Allocation Structures

The relationship between the software elements and the elements in one or more external environments

- 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?

- **Deployment:** Shows how software is assigned to hardware-processing (execution) and communication elements. Relations are allocated-to and migrates-to if the allocation is dynamic
- **Implementation:** How software elements (usually modules) are mapped to the file structure(s)
- **Work Assignment:** Assigns responsibility for implementing and integrating the modules to development teams

Summary of Structures of a System

Software Structure	**Relations**	**Useful for**
Decomposition	Is a submodule of; shares secret with	Resource allocation and project structuring and planning; information hiding, encapsulation; configuration control
Uses	Requires the correct presence of	Engineering subsets; engineering extensions
Layered	Requires the correct presence of; uses the services of; provides abstraction to	Incremental development; implementing systems on top of "virtual machines" portability
Class	Is an instance of; shares access methods of	In object-oriented design systems, producing rapid almost-alike implementations from a common template
Client-Server	Communicates with; depends on	Distributed operation; separation of concerns; performance analysis; load balancing
Process	Runs concurrently with; may run concurrently with; excludes; precedes; etc.	Scheduling analysis; performance analysis
Concurrency	Runs on the same logical thread	Identifying locations where resource contention exists, where threads may fork, join, be created or be killed
Shared Data	Produces data; consumes data	Performance; data integrity; modifiability
Deployment	Allocated to; migrates to	Performance, availability, security analysis
Implementation	Stored in	Configuration control, integration, test activities
Work Assignment	Assigned to	Project management, best use of expertise, management of commonality

Relating Software Structures to each other

- Different Structures provide different perspectives / concerns
- Different Structure are not independent – Elements of one Structure may be related to Elements of another
  - E.g. A module in the Decomposition Structure may be related to another structure in component-and-connector structure (to represent its runtime)
- Structures represent the primary engineering leverage points of an architecture

What Structures to Document?

- Its not practical (and often not needed) to document all Structures
    - Individual structures bring with them the power to manipulate one or more quality attributes – Architect should decide what are the key Architectural Qualities to achieve
    - Dominant Structures: There can be a structure that is dominant for a particular system which may need to satisfy the key requirements

Views

- The concept of a view, which you can think of as capturing a structure, provides us with the basic principle of documenting software architecture
- Different views support different goals and uses
  - For the Architect:
    - Deployment view will let you reason about your system's performance and reliability
    - Layered view will tell you about your system's portability
  - For the Developer:
    - Class view (Class Diagram) will tell you about Generalization/Inheritance and help you to reason about collections of similar behavior or capability

Stakeholders Vs. Architecture Documentation

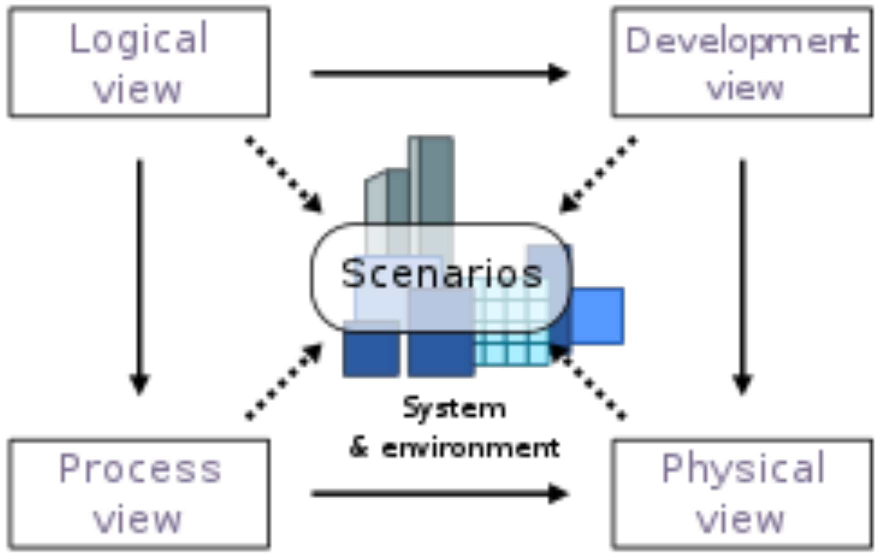
Stakeholder	**Module Views**				**Component & Connector Views**	**Allocation Views**	
	Decomposition	**Uses**	**Class**	**Layer**		**Deployment**	**Implementation**
Project Manager	S	S		S		D	
Member of Development Team	D	D	D	D	D	S	S
Testers and Integrators		D	D		S	S	S
Maintainers	D	D	D	D	D	S	S
Product Line Application Builder		D	S	O	S	S	S
Customer					S	O	
End User					S	S	
Analyst	D	D	S	D	S	D	
Infrastructure Support	S	S		S		S	D
New Stakeholder	X	X	X	X	X	X	X
Architect (Current & Future)	D	D	D	D	D	D	S

Information Level:	*D – Detailed*	*S - Some*
	O – Overview	*X - Any*

4+1 View Model

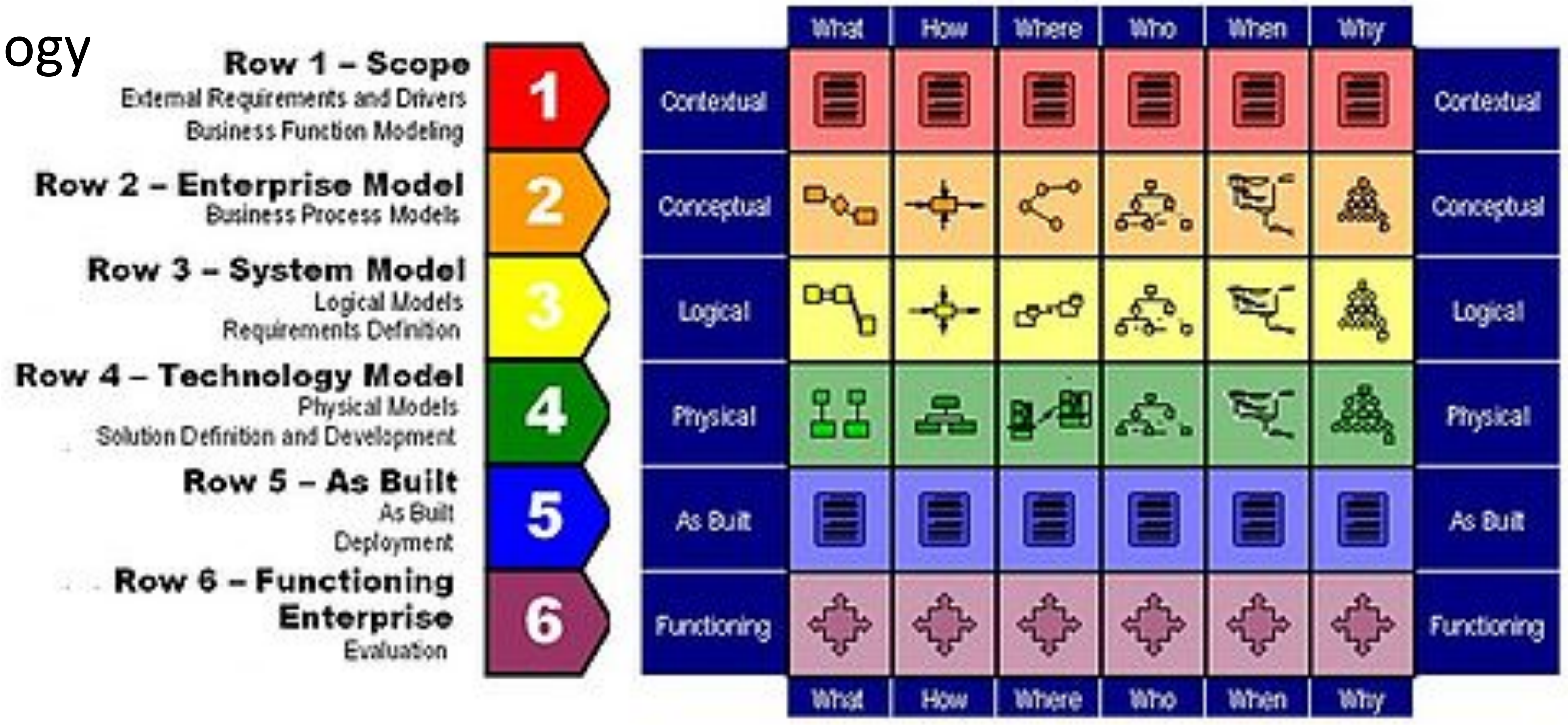
- The views are used to describe the system from the viewpoint of different stakeholders
- Concentrates on:
  - Logical View
  - Development View
  - Process View
  - Physical View

  + Scenarios (Use Cases)



Enterprise Architectural Viewpoints

- Zachman Framework
  - Formal and structured way of viewing and defining an enterprise
  - An Enterprise Ontology



References

- http://www.ece.ubc.ca/~matei/EECE417/BASS/ch02lev1sec5.html
- http://www.ece.ubc.ca/~matei/EECE417/BASS/ch09lev1sec3.html
- https://www.cs.ubc.ca/~gregor/teaching/papers/4+1view-architecture.pdf
- https://www.zachman.com/