Lecture 2

Contexts of Software Architecture & Architectural Structures

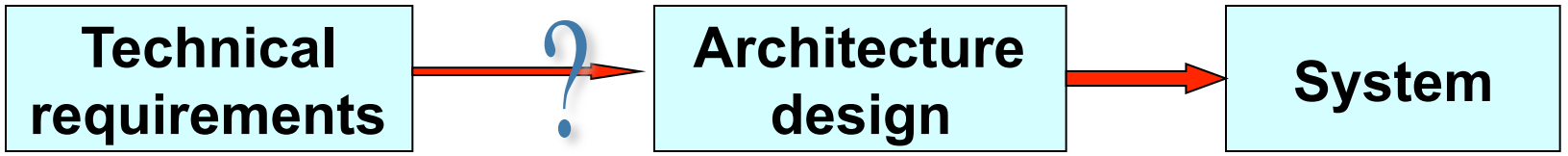
主讲教师：王灿
Email: wcan@zju.edu.cn
TA: 李奇平 liqiping1991@gmail.com
Course FTP: ftp://sa:sa@10.214.51.13

What Is Software Architecture?

- [BCK13] The software architecture of a system is the ***set of structures*** needed to reason about the system, which comprise software elements, relations among them, and properties of both.

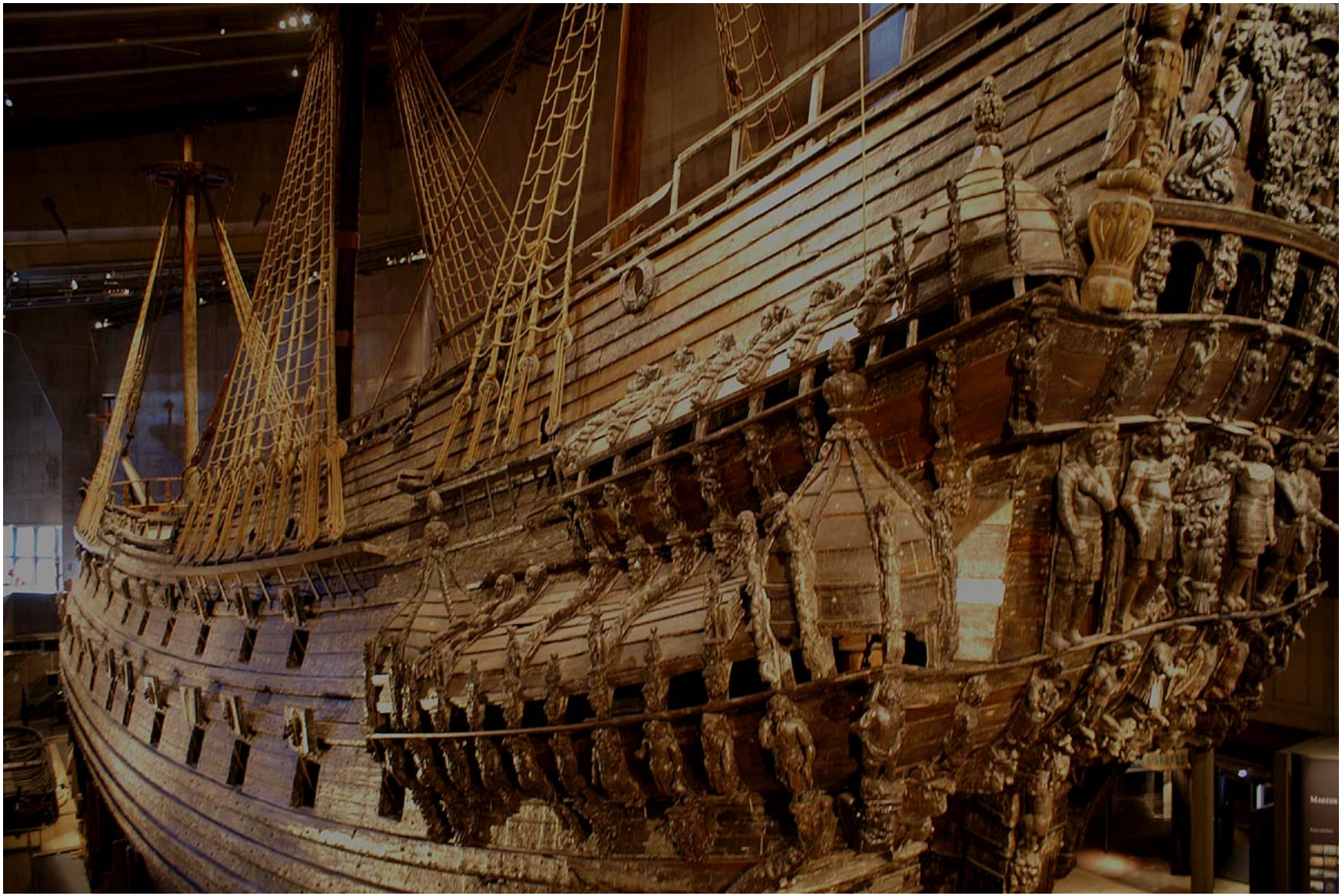
Where Do Architectures Come From?

- What Determines an Architecture?



- Given the same requirements specification to two different architects, will they produce the same architecture?

The Swedish Warship Vasa (1)



The Swedish Warship Vasa (2)

- Built from 1626 to 1628, the Vasa was to be the world's largest war ship
  - 70 meters long
  - carrying 300 soldiers
  - two gun decks with an astonishing 64 heavy guns mounted on
- The Vasa foundered and sank after sailing less than 2km into her maiden voyage on 10 August 1628.

Contexts of Software Architecture

- Architecture supports and is influenced by the various contexts in which it plays a role. These contexts include:
  - Technical
  - Project life cycle
  - Business
  - Professional

Architecture in a Technical Context

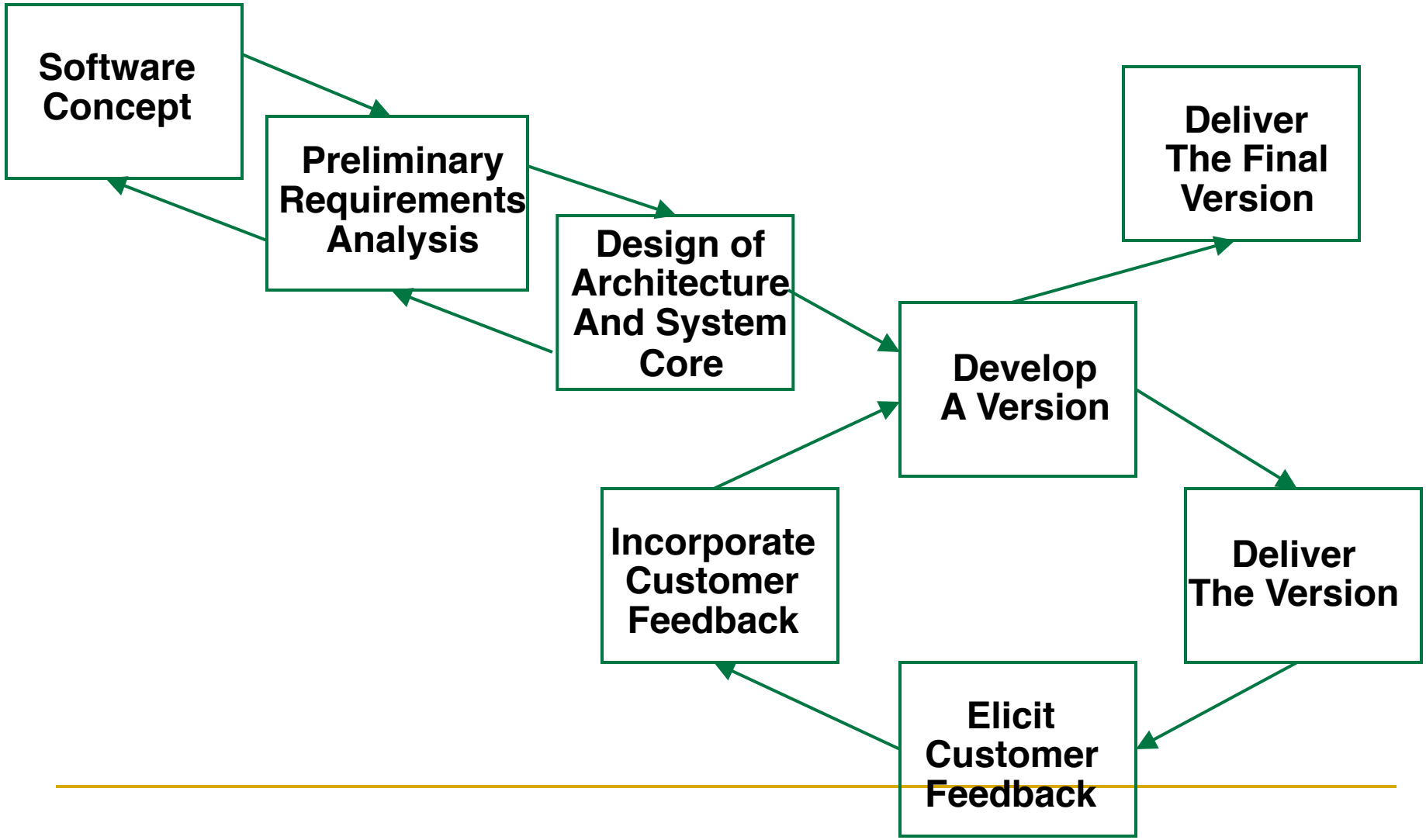
- Architecture inhibit or enable the achievement of *quality attributes*

- Technical environment will influence the architecture
  - e.g. popular design for an information system nowadays: Web-based, object-oriented, service-oriented, mobility-aware, cloud-based, social networking-friendly

Architecture in a Project Life-Cycle Context

- Four dominant software development processes
  - Waterfall
  - **Iterative**
  - Agile
  - Model-driven development

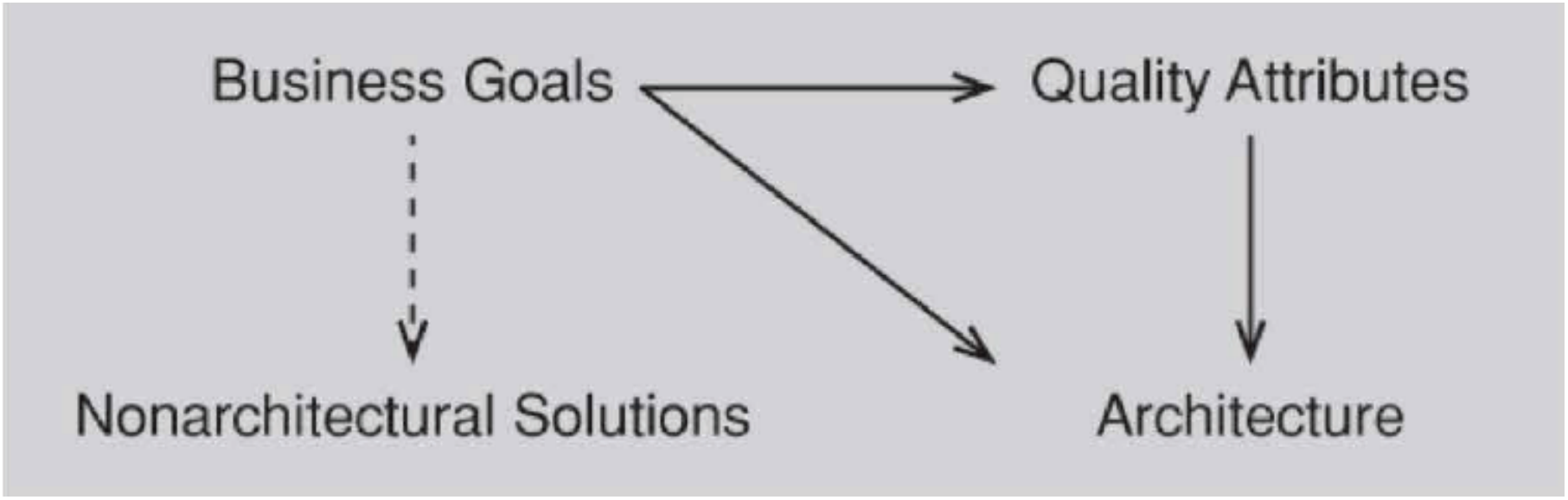
Evolutionary Delivery Life Cycle



Activities Involving Software Architecture in a Development Process

1. Making a business case for the system
2. Understanding the architecturally significant requirements
3. Creating or selecting the architecture
4. Documenting and communicating the architecture
5. Analyzing or evaluating the architecture
6. Implementing and testing the system based on the architecture
7. Ensuring that the implementation conforms to the architecture

Architecture in a Business Context



Architecture in a Professional Context

- Architecture is influenced by the architects' background and experience.

Stakeholders

Name	Interest in architecture
Architect	Negotiating and making tradeoffs among competing requirements and design approaches. A vessel for recording design decisions. Providing evidence that the architecture satisfies its requirements
Deployer	Understanding the architectural elements that are delivered and to be installed at the customer or end user's site, and their overall responsibility toward system function.
Designer	Resolving resource contention and establishing performance and other kinds of runtime resource consumption budgets. Understanding how their part will communicate and interact with other parts of the system.
Implementer	Understanding inviolable constraints and exploitable freedoms on development activities.
Integrator	Producing integration plans and procedures, and locating the source of integration failures
Maintainer	Understanding the ramifications of a change
Project Manager	Helping to set budget and schedule, gauging progress against established budget and schedule, identifying and resolving development-time resource contention
Tester	Creating tests based on the behavior and interaction of the software elements.
User	Users, in the role of reviewers, might use architecture documentation to check whether desired functionality is being delivered. Users might also use the documentation to understand what the major system elements are, which can aid them In emergency field ma1ntenance.

See Table 3.1 in P54

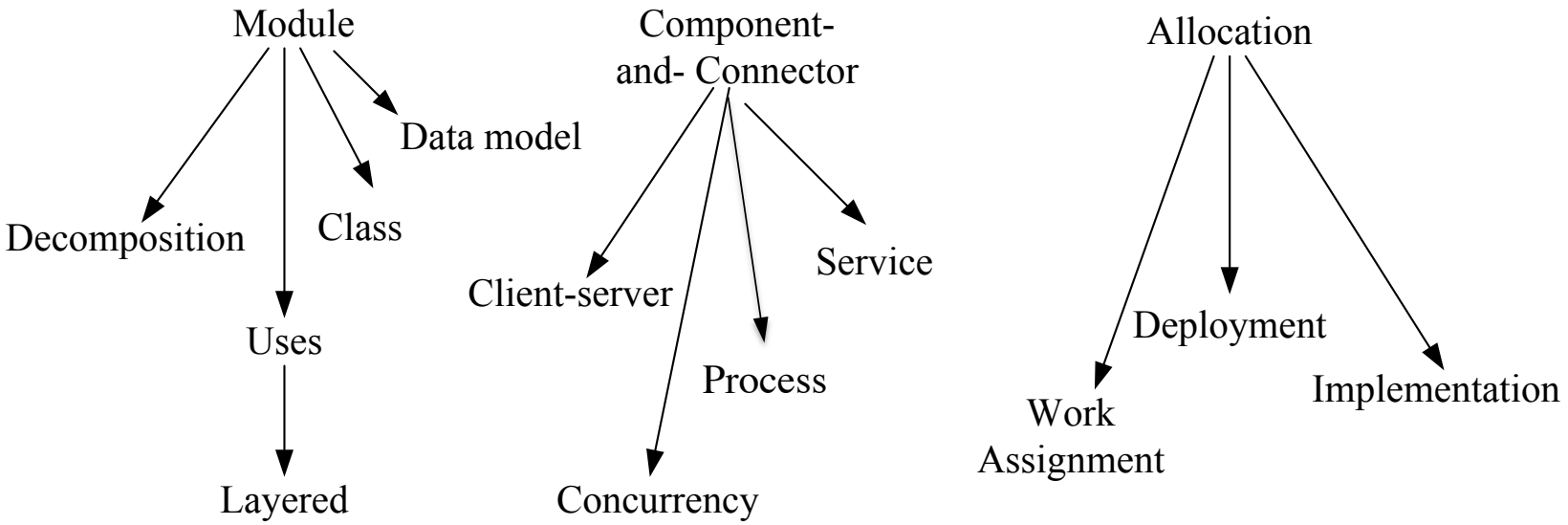
Architectural Structures

- In a house, there are plans for
  - rooms
  - electrical wiring
  - plumbing
  - ventilation
- Each of these constitutes a “view” of the house.
  - used by different people
  - used to achieve different qualities in the house
  - serves as a description and prescription
- So it is with software, a structure is a ***view*** of architecture.

Groups of Software Architectural Structures

- Module structures:
  - How is the system to be structured as a set of code units (modules)?
- Component-and-connector structures
  - How is the system to be structured as a set of elements that have runtime behavior (components) and interactions (connectors)?
- Allocation structures
  - How is the system to relate to non-software structures in its environment

Common Software Architecture Structures



Decomposition Structure

- Units: modules
- Relations: “is a sub-module of”
- Used: as a basis for development project's organization, including the structure of the documentation, and its integration and test plans.
- Affected attributes include: modifiability, understandability

Uses Structure

- Units: modules or procedures or resources on the interfaces of modules
- Relations: uses (“assumes the correct presence of”)
- Used: to engineer subsets, supersets
- Affected attributes include: reusability, testability, incremental development

Layered Structure

- Units: layers (a coherent set of related functionality)
- Relations: layer n only uses the services of layer n-1
- Used: to hide implementation details, to provide design abstraction (virtual machine)
- Affected attributes include: portability

Class Structure (Generalization)

- Units: classes
- Relation: “inherits from,” “is an instance of”
- Used: to exploit similarity among classes
- Affected attributes include: reusability, incremental addition of functionality

Data Model

- Units: data entities
- Relation: relationship between data entities
- Used: to describe static information structure

Service Structure

- Components: services
- Connectors: service coordination mechanism such as SOAP
- Used: analyze and engineer service-oriented system
- Affected attributes include: performance, modifiability

Concurrency Structure

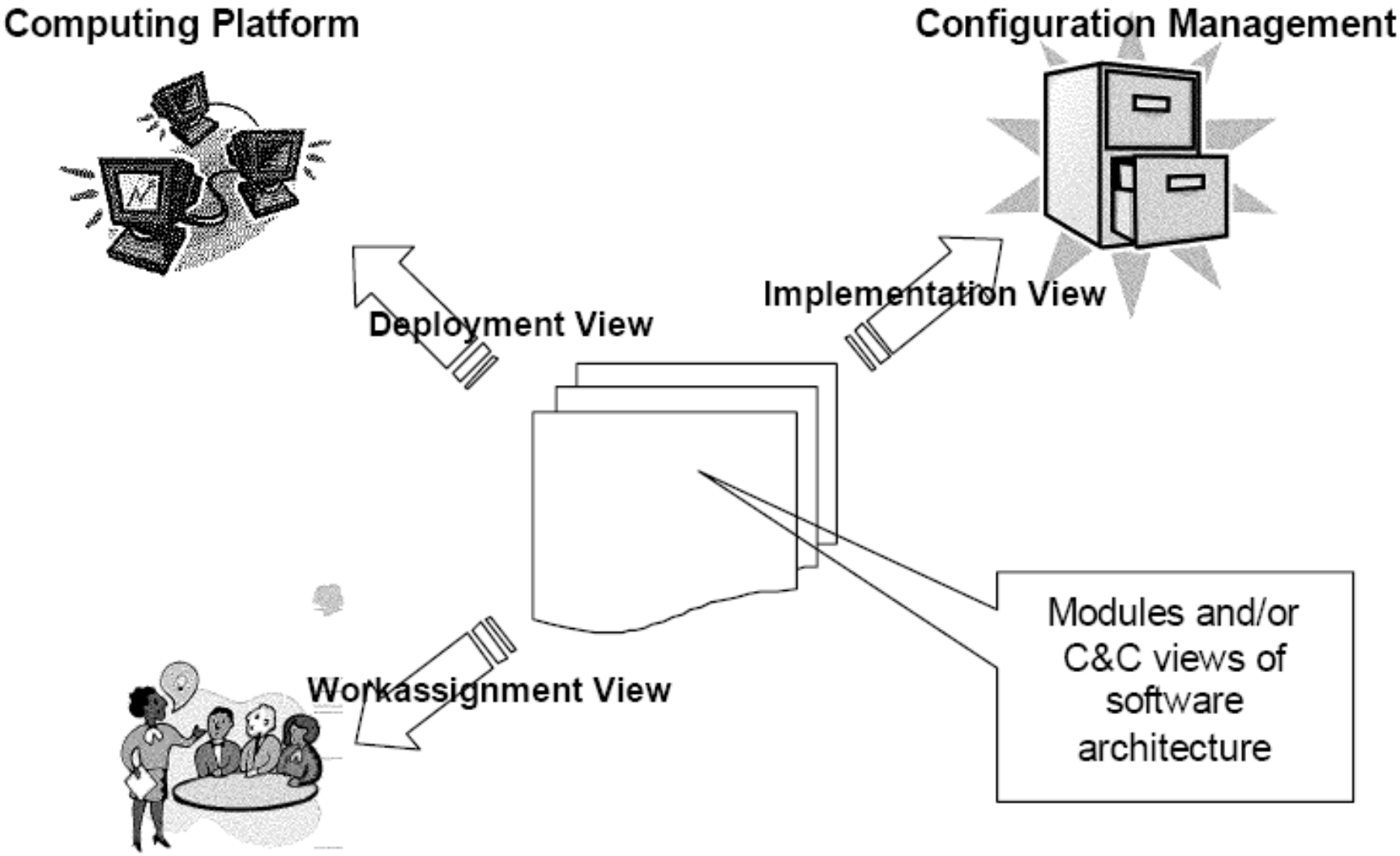
- Components: “logical threads of execution”
- Connectors: corresponding communication mechanisms
- Used: to identify locations where resource contention exists, where threads may fork, join, be created or be killed
- Affected attributes includes: parallelism, performance

Process Structures (communicating process)

- Units: processes or threads
- Relations: “synchronizes with,” “excludes,” “preempts”
- Used: to tune system runtime performance, exploit multiprocessing hardware
- Affected attributes include: performance

Client-Server Structure

- Components: clients and servers
- Connectors: request/reply, the asymmetric invocation of server's services by a client
- Used: to separate concerns, to distribute operations and to balance load
- Affected attributes include: performance, modifiability


Computing Platform
Configuration Management
Deployment View
Implementation View
Workassignment View
Modules and/or C&C views of software architecture
Development Organization

Deployment Structures

- Units: software, hardware and communication pathways
- Relations: “allocated to”, “migrates to”
- Used: to reason about performance, data integrity, availability, and security.

Implementation Structures

- Units: modules, files
- Relations: mapping, stored in
- Used: to manage development activity, to configure, integrate and test software modules

Work Assignment Structures

- Units: modules, development teams
- Relations: “assigned to”
- Used: to assist project management, to enable best use of expertise and to manage commonality

Example of Using Multiple Views: Concurrency Design

- Module decomposition view
  - Providing containers for functionality
  - Discover major data flow relationships among modules
- Concurrency view
  - Presenting “logical threads of execution”
  - Uncovering resource contention, deadlock, data consistency issues, thus revealing synchronization points in the design
- Process view
  - Revealing the concurrency infrastructure at the OS level
- Deployment view

Relate Structures to Each Other

- Though each of these structures provides a different system perspective, they are interrelated with each other.
- In most cases, the dominant structure of a software system is module decomposition.
- Not all systems warrant consideration of many architectural structures.
- Structures represent the primary engineering leverage points of an architecture.

Reading Assignment

- Read Chapter 4 & 5 of the textbook.