FPT University Can Tho

Introduction to Software Architecture
Chapter 1

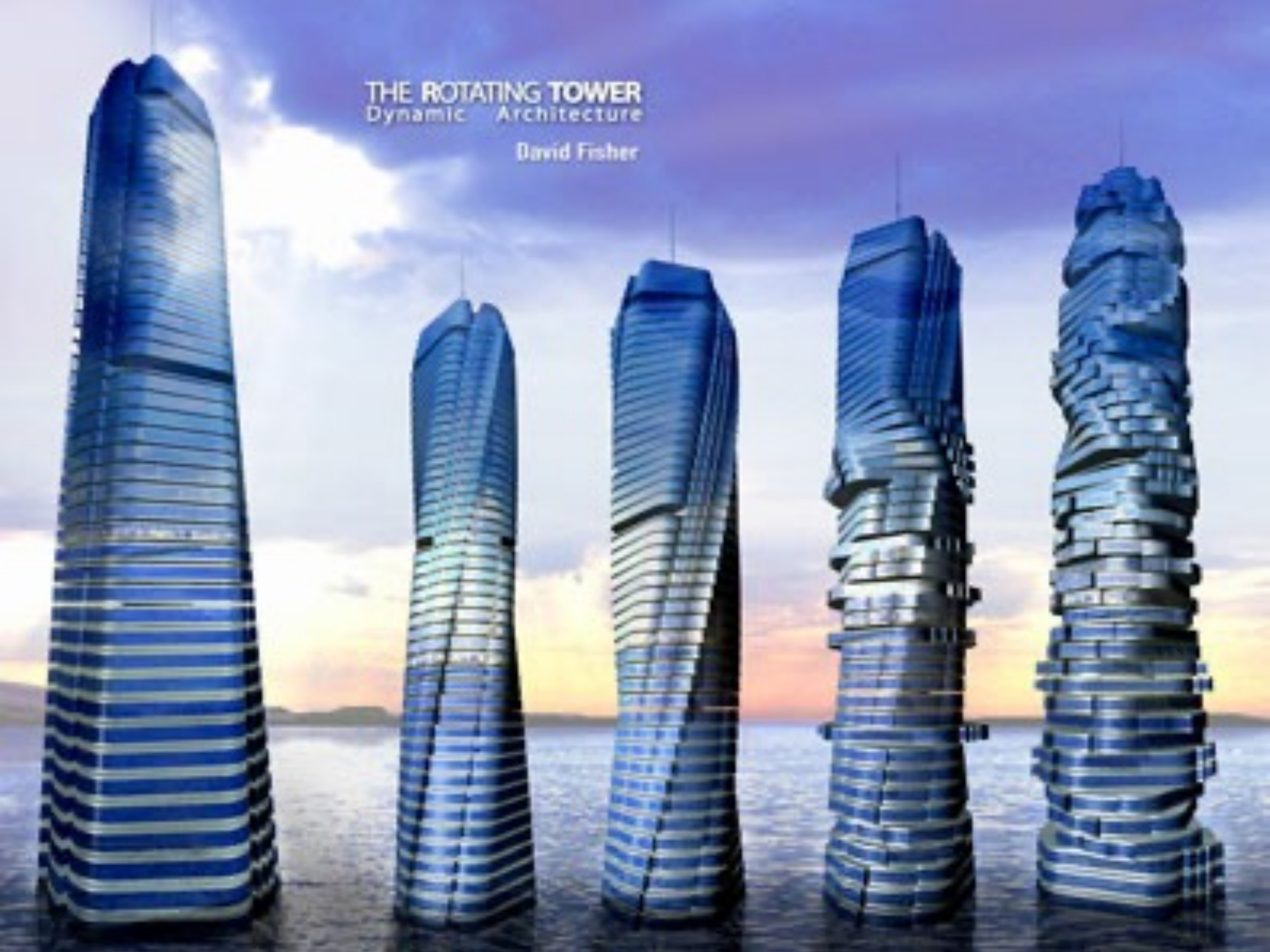
Objectives

- Introduce the relationship between software requirements and architecture

- Introduce the relationship between architectural styles and architecture

- Introduce the elements of software architecture

- Describe quality attributes trade-off analysis



THE ROTATING TOWER
Dynamic Architecture

David Fisher

Software Architecture

- Definitions
  - IEEE Std 1471
    - the system architecture as “the fundamental organization of a system
      - embodied in
        - its components,
        - their relationships to each other, and to the environment
        - the principles guiding its design and evolution.
  - Shaw and Garlan (1996)
    - software architecture as *the description of*
      - *elements that comprise a system,*
      - *the interactions and patterns of these elements,*
      - *the principles that guide their composition*
      - *the constraints on these elements.*

Software Architecture

- Architects use various design strategies in software construction
  - To divide and conquer the complexities of a problem domain
  - To solve the problem.
- A good software design
  - Reduces risks in software production,
  - Coordinates development teams to work together orderly
  - Makes the system traceable for implementation and testing
  - Leads to software products have higher quality attributes.

Software Architecture…

- The input of software design
  - Software Requirements Specification (SRS).
    - is the result of requirement analysis
      - functional and non-functional requirements that must be met by the software system.

Software Architecture…

- The <u>output</u> of software design
  - Software Design Description (<u>SDD</u>).
    - Includes
      - the <u>software architecture</u> or <u>high-level design</u>
      - the <u>detailed design</u> of the system.
    - The blueprint for the implementation phase.
    - Describes
      - the <u>components</u> of a system
      - the <u>modules</u> that comprise each component
      - the detailed information of each module, such as
        - » Data attributes
        - » Operations
        - » Algorithms.

=>The system is then implemented using programming language, debugging, testing, and maintenance.

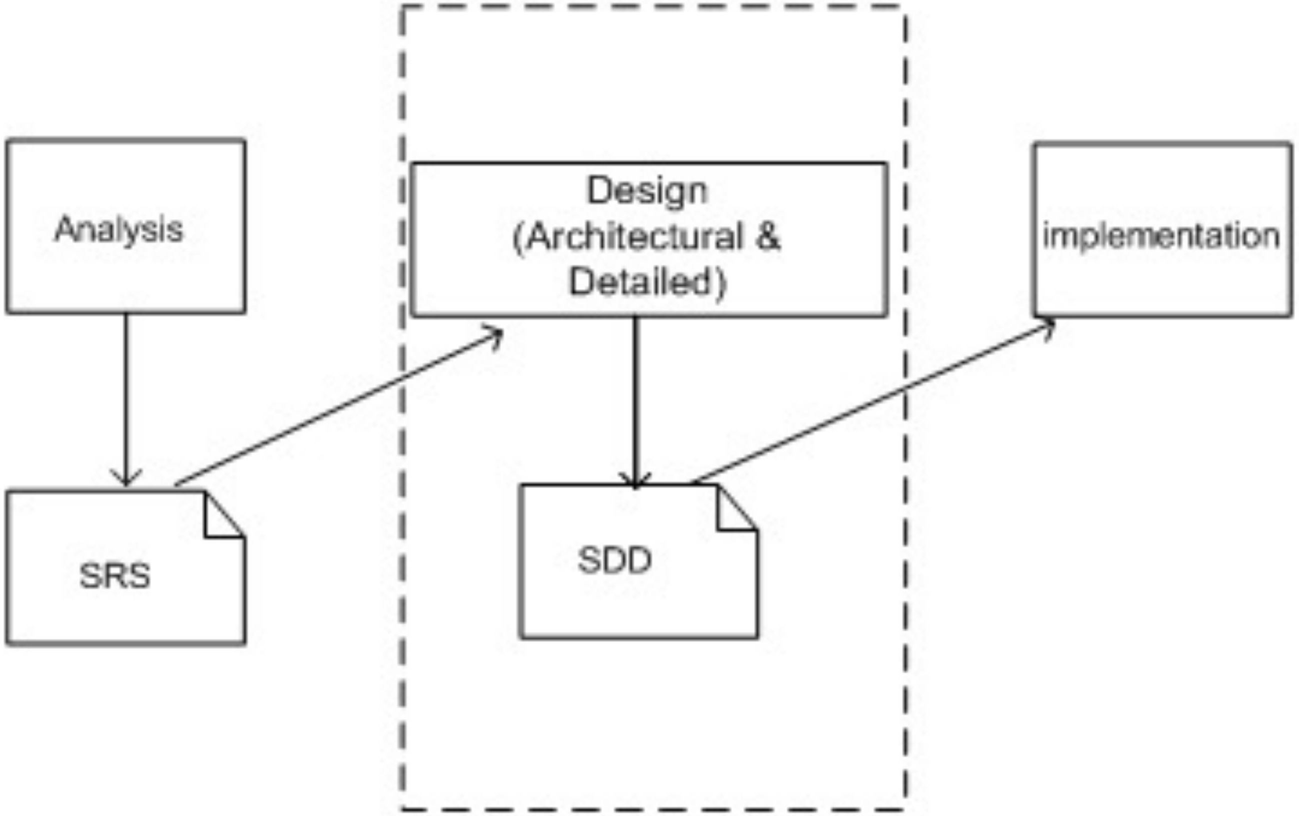


Figure 1.1 A simplified Software development life cycle

Software Architecture…

- Sample outline of SDD based on IEEE Std 1016.
  - Design overview, purpose, scope
  - Decomposition description
    - Module
    - Data
    - Process
  - Dependency and connection description
    - between modules, data, processes
  - Attributes
  - User interface description
  - Detailed design
    - module and data

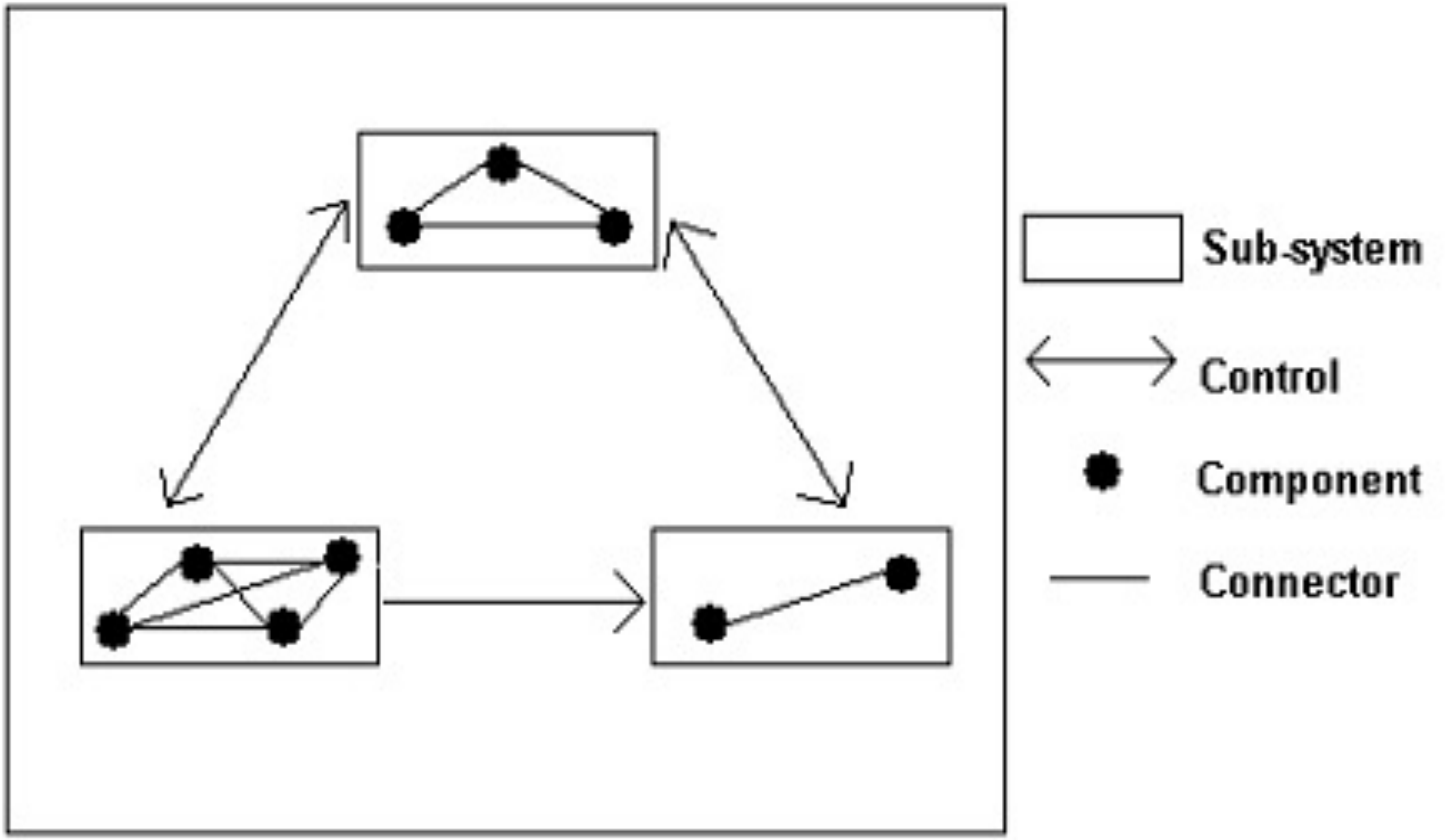
Software Architecture…

- Software design stage can be split into 2 steps
  - Architectural design step
    - we describe
      - user accessible components
      - the interconnections among them
        - visible to stakeholders
  - Detailed design step.
    - we specify
      - The internal details of each component
      - Might introduce new invisible components
        - to the stakeholders.

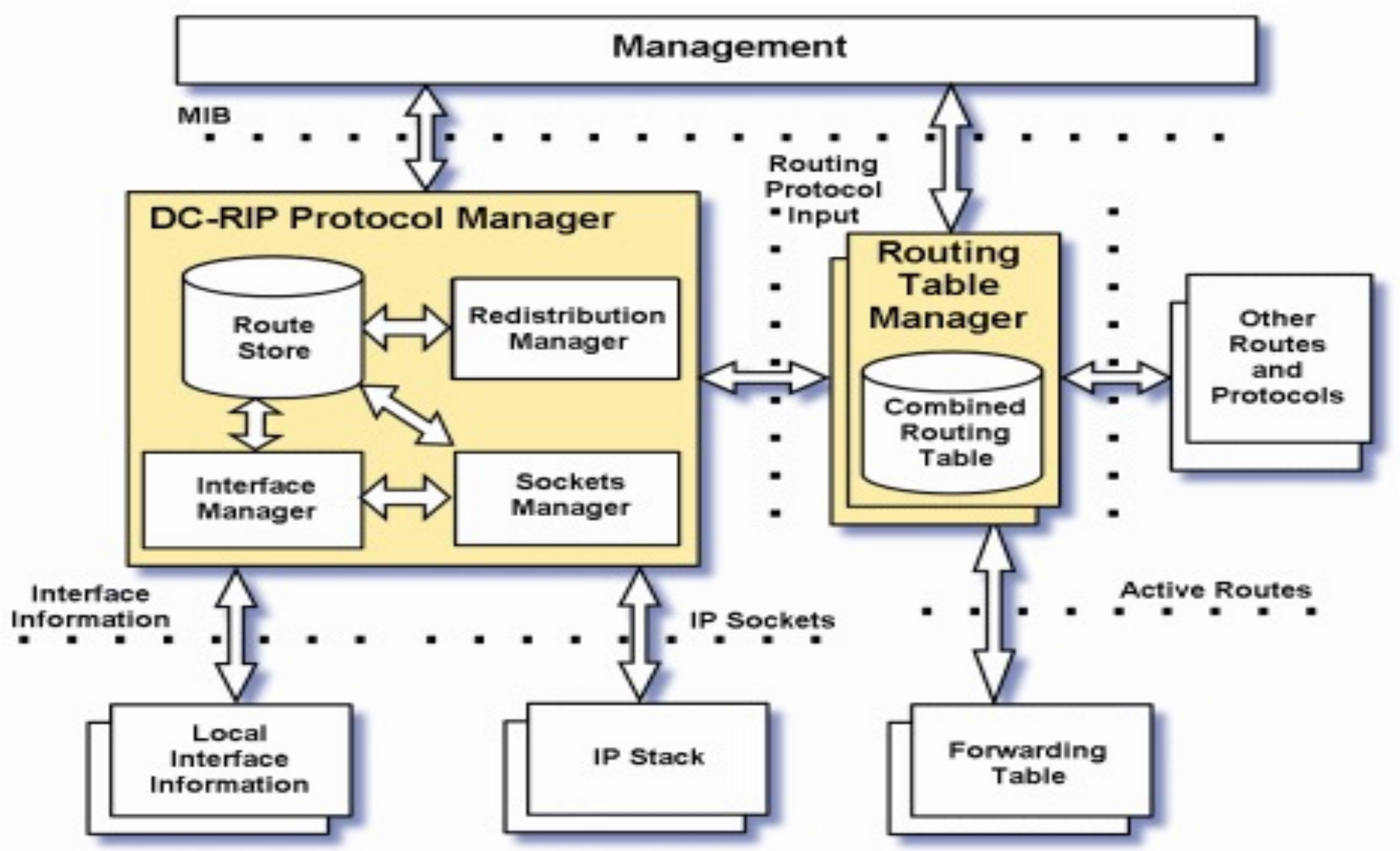
Software Architecture

- Bridging Requirements and Implementation
  - Software architecture plays a very important role in software development.
- The architectural design
  - embodies earliest decisions
    - that have a decisive impact on the ultimate success of the software product.

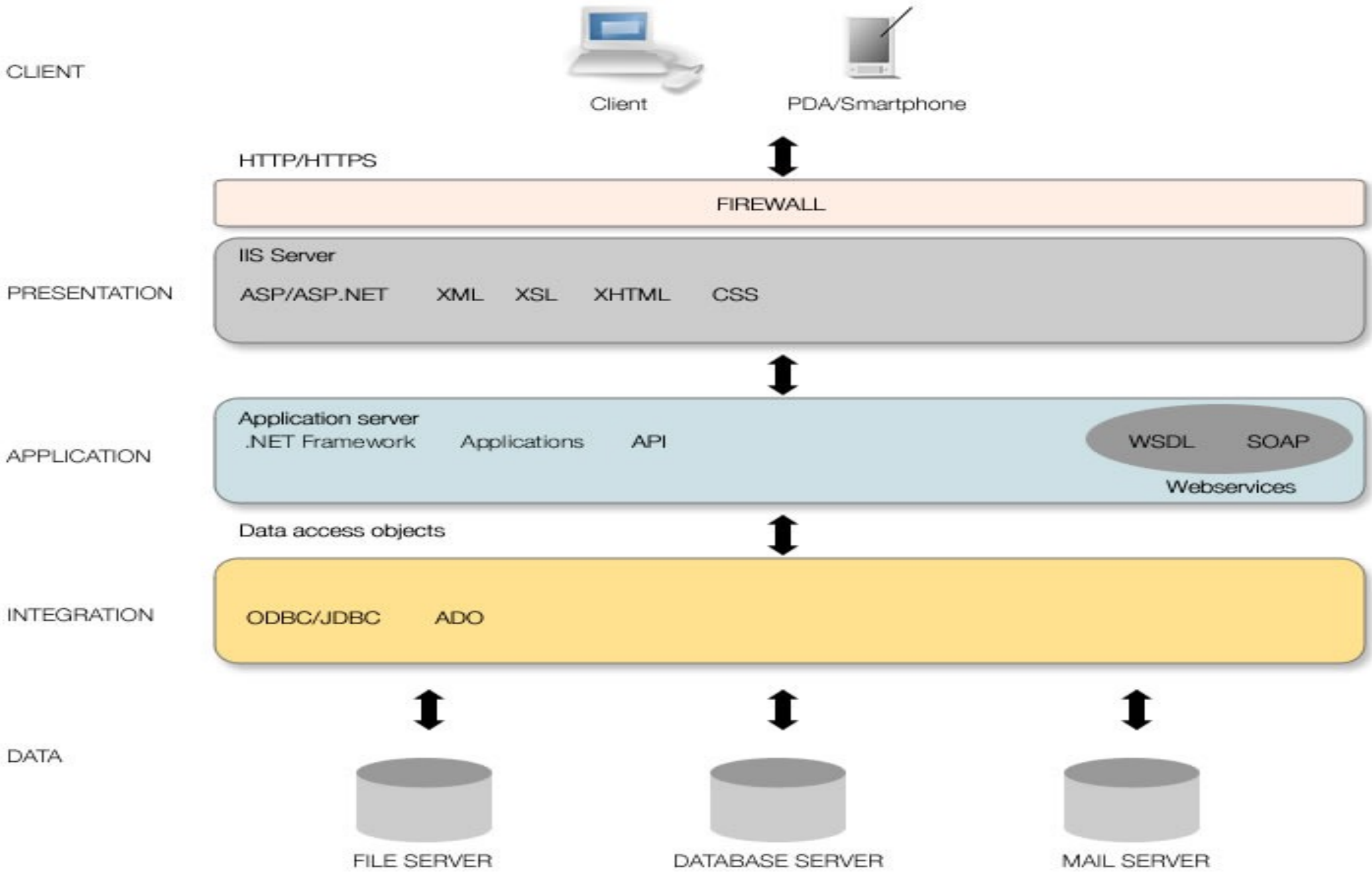
Software Architecture



Software Architecture



Software Architecture



Architectural Styles

- Abstracts the common properties of a family of similar designs.
- Also known as “architecture pattern”
- contains a set of
  - rules, constraints, and patterns of how to structure a system into a set of
    - elements and connectors.
- Governs the overall structure design pattern of
  - constituent element types and their runtime interaction of flow control and data transfer.

Architectural Styles

- Key components of an architectural style are:
  - Elements
    - that perform functions required by a system
  - Connectors
    - that enable communication, coordination, and cooperation among elements
  - Constraints
    - that define how elements can be integrated to form the system
  - Attributes
    - that describe the advantages and disadvantages of the chosen structure

Architectural Styles

- In this course, we will discuss about:
  - Batch Sequence, Pipe & Filter, Process Control (Data Flow)
  - Repository, Blackboard (Data Centered)
  - Object-Oriented
  - Layered, Virtual Machine, Main/Subroutine (Hierarchy)
  - Multi-tier, Client/Server (Distributed)
  - Event-Based, Buffered Messaging (Asynchronous Communication)
  - MVC, PAC (Presentation-Abstraction-Control,) (Interaction Oriented)

Introduction to Software Architecture - P2

Software architect’s tasks

- Perform
  - System static partitioning
  - System decomposition into sub-systems and communications between sub-systems.
    - A software element can be
      - configured, delivered, developed, deployed and is replaceable in future.
    - Each element has its interface
      - that encapsulates details and provides loose coupling with other elements or sub-systems.

Software architect’s tasks

- Establish dynamic control relationships
  - between different sub-systems in terms of
    - Data flow
    - Control flow orchestration
    - Message dispatching.
- Consider and evaluate
  - alternative architecture styles for the problem at hand.
- Perform trade-off analysis on quality attributes and other non-functional requirements
  - during the selection of architecture styles.
- The most important jobs
  - To map the software requirements specification to the software architecture
  - To guarantee that the software architecture satisfies
    - functional and non-functional requirements.

Quality Attributes

- Are closely related to architectural styles.
  - Each architectural style
    - supports some quality features.
    - has its advantages, disadvantages and potential risks.
      - Choosing the right architectural style to satisfy
        - required quality attributes is also very important in addition to function satisfaction.
- Are identified in the requirement analysis process.
- An architectural style encapsulates tradeoffs among many conflicting quality attributes.

Quality Attributes

List of sample quality attributes:

- Performance
- Reliability
- Portability
- Usability
- Security
- Testability
- Maintainability
- Availability
- Flexibility
- Interoperability
- Scalability.
- Time to market
- Cost
- Life time

Quality Attributes

- Classifications
  - Implementation attributes
    - not observable at runtime
  - Runtime attributes
    - observable at runtime
  - Business attributes

Implementation attributes

- Interoperability:
  - Refers to the universal accessibility and
  - The ability to exchange data with internal components and the outside world.
  - It needs loose dependency of infrastructure.
- Maintainability & extensibility:
  - Refers to the ability to modify the system and extend it conveniently.
- Testability:
  - Refers to the degree
    - to which the system facilitates the establishment of test cases.
  - Usually requires a complete set of documentation accompanied with system design and implementation.

Implementation attributes

- Portability:
  - Refers to the level of independence of the system on software and hardware platforms.
    - Systems developed using high-level programming languages usually have good portability.
    - Ex : Java
      - most Java programs need only be compiled once and can run everywhere.
- Scalability:
  - Refers to the ability to adapt to an increase of user requests volume.
  - It disfavors bottlenecks in system design.

Implementation attributes

- Flexibility:
  - Refers to the ease of modification of a system to cater for different environment or problems
    - for which the system is originally not designed.
  - Ex : Systems developed using the component based architecture or the service-oriented architecture usually possess this property.

Runtime attributes

- Availability:
  - Refers to the ability of a system to be available 24x7.
    - Availability can be achieved via replication and careful design to cope with failures of hardware, software, or the network
- Security:
  - Refers to the ability to cope with malicious attacks from outside or inside of the system.
    - Security can be improved by installing firewalls and establishing authentication and authorization processes, and using SSL and encryption.

Runtime attributes

- Performance
  - Refers to increasing efficiencies such as
    - Response time
    - Throughput
    - Generally resource utilization

    -> which most of the time conflict with each other.
- Usability
  - Refers to the level of "satisfaction" from a human perspective in using the system.
    - Includes
      - completeness
      - Correctness
      - Compatibility
      - more critically user friendliness.

Business attributes

- Time to market
  - Refers to the time it takes from requirement analysis to the date product is released.
- Cost
  - Refers to expense of building, maintaining, and operating the system.
- Lifetime
  - Refers to the period of time that the product is "alive" before retirement.

Quality attributes trade-off

- **Tradeoff between space and time.**
  - Ex: to increase the time efficiency of a hash table means to decrease its space efficiency.
- **Tradeoff between reliability and performance.**
  - Ex: Java programs are well protected against buffer overflow due to its security measures such as boundary check on arrays.
    - Such reliability features come at the cost of time efficiency, compared with the simpler and faster C language which provides the “dangerous” yet efficient pointers.

Quality attributes trade-off

- **Tradeoff between scalability and performance.**
  - Ex: one typical approach to increase the scalability of a service is to replicate servers. To ensure consistency of all servers (e.g., to make sure that each server has the same logically consistent data), performance of the whole service is sacrificed.

Software Architecture Design Guidelines

*Think of **what** to do before thinking of how to do it*.

- Functional and non-functional requirements should be identified, verified, and validated before architectural and detailed design.
- Using an abstract architectural design of a system to communicate with stakeholders helps avoid overhauling the system design in later stages of the software development cycle.

FPT UNIVERSITY

Software Architecture Design Guidelines

*Think of **abstract** design before thinking of concrete design.*

- Always start with an abstract design that specifies interfaces of components and abstract data types.
- Use multiple levels of abstraction if necessary.
- Make all implementation decisions depend on the abstract interfaces instead of concrete ones because those are more stable – they are the contracts between service providers and services requesters so they are defined at the early stages of software development cycle.

Software Architecture Design Guidelines

*Think of **non-functional requirements** earlier.*

- When we map functional requirements to an architectural design, we should consider non-functional requirements as well.
- Communicate with stakeholders and document their preferences of quality attributes.
- It is not possible to find a design that meets all quality attributes.
- Balance the quality attributes, and consider heterogeneous architecture styles when necessary.

Software Architecture Design Guidelines

*Think of **software reusability and extensibility** as much as possible*.

- For most software systems, it is very likely that new functionalities will be added after they are deployed.
- In addition, we need to consider how to reuse existing software components to increase the reliability and cost-effectiveness of new systems.
- Always try hard to make software extensible in the future.

Software Architecture Design Guidelines

*Tolerate **refinement** of design*.

- Never expect to have software design completely perfect within one step.
- We may need to use prototyping and iteration to refine the software design.
- *Avoid ambiguous design and over-detailed design*.
- Ambiguous design lacks constraints and over-detailed design restricts implementation.

*Try to **promote high cohesion** within each element and loose-coupling between elements*.

- A highly coherent sub-system, component, or module performs sole function only.
- A similar concern is that each architectural style should show a very clear division between elements to guarantee loose-coupling.

Summary

- Software architectural design has emerged as an important part of software development.
- A software architecture specification consists of software elements, connections and collaborations among the elements, and desired software quality attributes.

Summary

- An architectural style is a set of rules, constraints, or patterns that guide how to structure a system into a set of elements and connectors, and how to govern overall structure design patterns of constituent element types and their runtime interaction.
- One specific architectural style may not be able to honor all quality attributes of the system.

Summary

- There are always quality attribute tradeoffs between different styles.
- Hence how to keep a balance on quality attributes is an important design issue.

Review Question

- 1. The constituent elements of software architecture are software elements and their connections
  - True
  - False

Review Question

- 2. Software architecture design involves many software design methodologies and architecture styles
  - a. True
  - b. False

Review Question

- 2. Software architecture design involves many software design methodologies and architecture styles
  - a. True
  - b. False

Review Question

- 3. The purpose of the software design phase is to produce a software requirement specification
  - a. True
  - b. False

Review Question

- 4. Object-oriented design is a design methodology
  - a. True
  - b. False

Review Question

- 5. Pipe-and-filter is one of the architecture styles
  - a. True
  - b. False

Review Question

- 6. Software architecture is a static software structure description
  - a. True
  - b. False

Review Question

- 7. Software quality attributes must satisfy functional requirements.
  - a. True
  - b. False

Review Question

- 8. Architecture styles contribute to software quality attributes
  - a. True
  - b. False

Review Question

- 9. Software architecture = software architecture styles
  - a. True
  - b. False

Review Question

- 10. Software architecture design is based on the software requirement specification.
  - a. True
  - b. False

References

- Text book: Software Architecture and Design Illuminated
- FPTU slides
- MS.c Luong Hoang Huong’s SWD391 Slides