



## Software Architecture and Design I SOEN 343

**Instructor: Dr. Rodrigo Morales**  
rodrigo.moralesalvarado@concordia.ca

## Lecture 1: Introduction to software architecture

### **Textbook:**

Software architecture and design illuminated / Kai Qian ... [et al.]  
Publisher Sudbury, Mass. : Jones and Bartlett Pub., c2010.

# 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

# Software architect

- Software architects use various design strategies in software construction to divide and conquer the complexities of a problem domain and 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 that have higher quality attributes



The input of software design is a Software Requirements Specification (**SRS**)

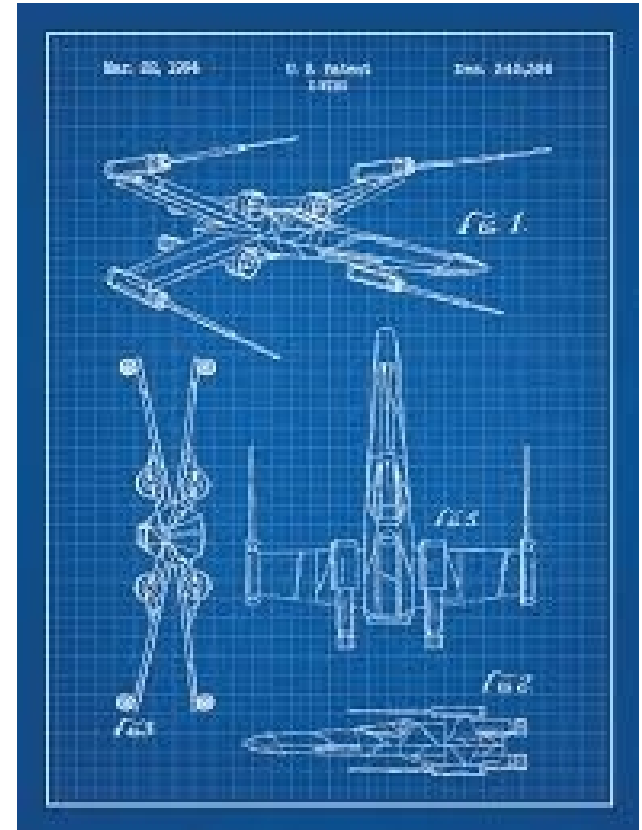
The **SRS** is the result of requirement analysis; it records the functional and non-functional requirements that must be met by the software system.

The output of software design is a document called Software Design Description (**SDD**)

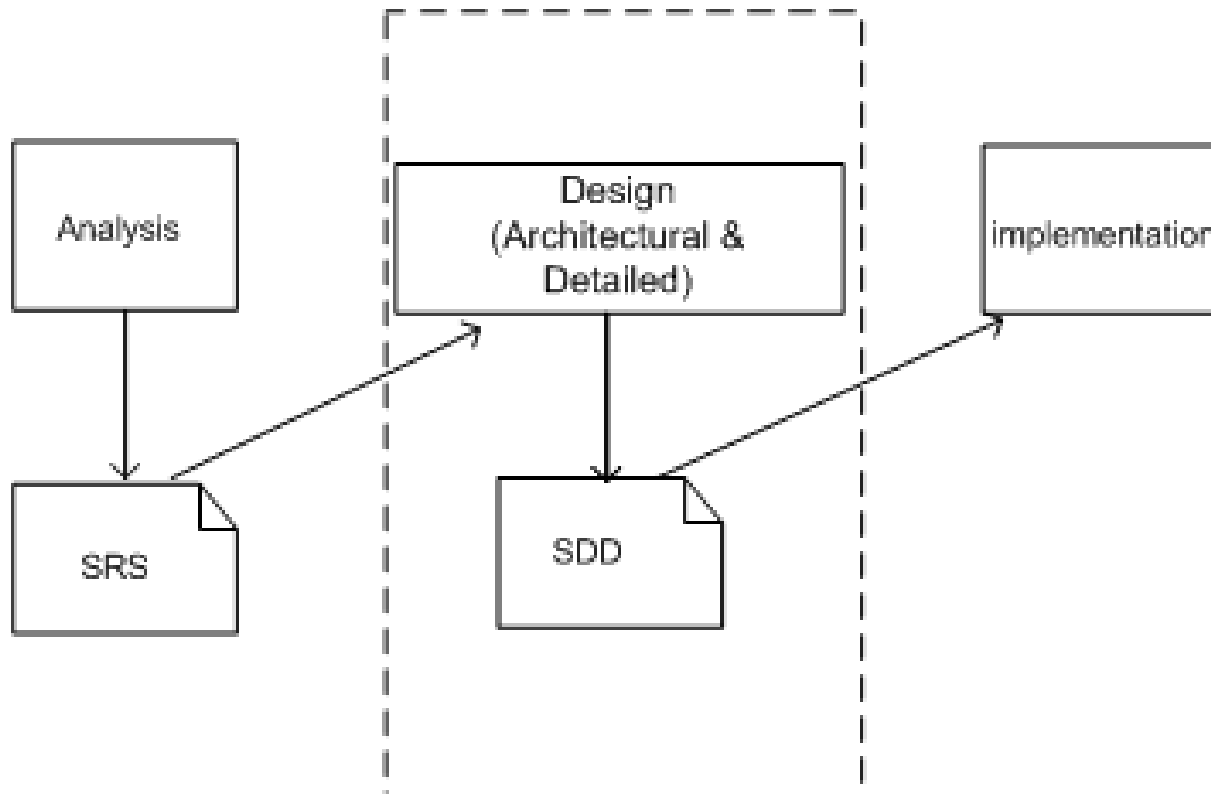
The **SDD** describes the software architecture or high-level design and the detailed design of the system.

# Software design description (SDD)

- The SDD describes the components of a system, the modules that comprise each component, and the detailed information (such as data attributes, operations, and algorithms) of each module
- From the SDD, the system is then implemented using programming language, which is followed by debugging, testing, and maintenance



# A simplified software development life cycle



# 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)



# Architectural vs detailed design

- The software design stage can be further split into two steps: the *architectural design* step and the *detailed design* step
- During the *architectural design* step we describe user accessible components and the interconnections among them that are visible to stakeholders
- During the *detailed design* step we specify the internal details of each component and we might introduce new invisible components (for the stake holder) into the design

# Architectural style

- Designers abstract common features into “families of architectures”
- Each style represents a layout topology of elements, and connections and interactions among these
- It also describes its semantic constraints and behaviors concerning data transfer and control transfer among the elements in the system and the quality attributes trade-off as well



- Quality attributes are closely related to architectural styles
- Each architectural style supports some quality features
- An architectural style encapsulates tradeoffs among many conflicting quality attributes

# Architectural style

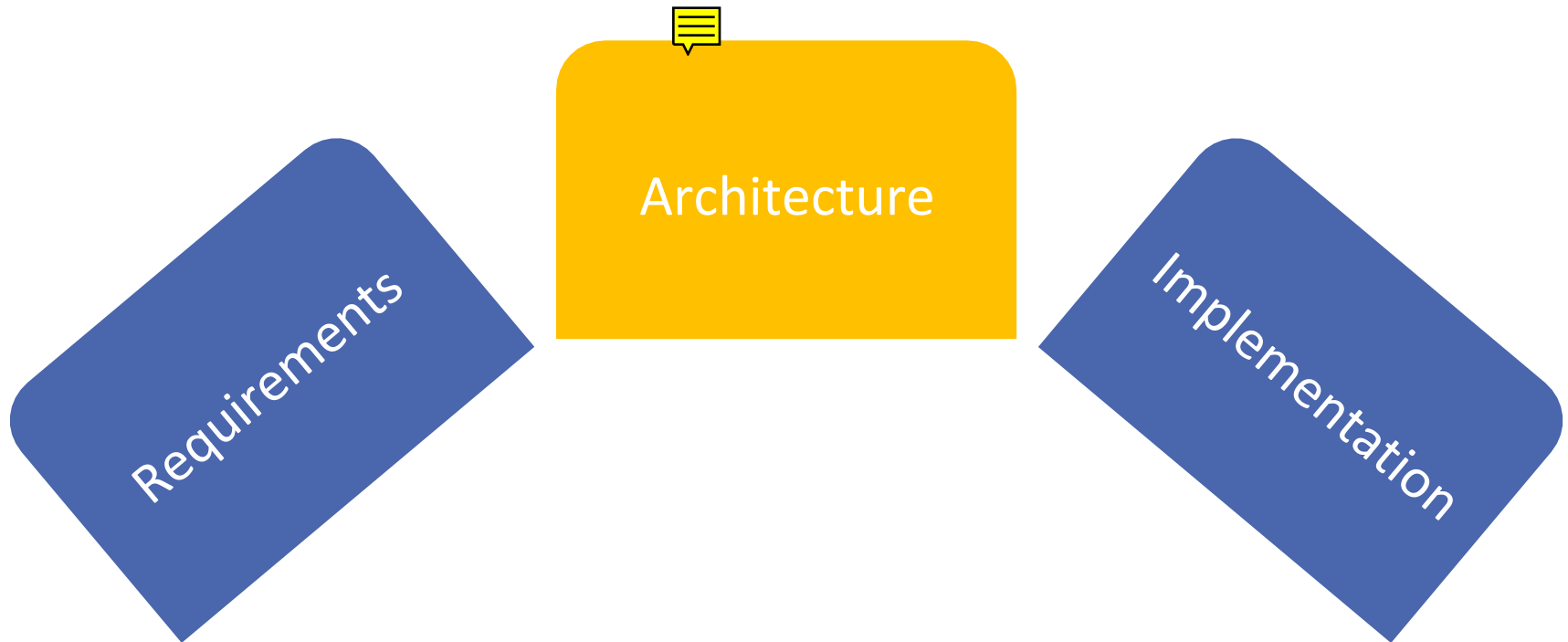


List of sample  
quality attributes:

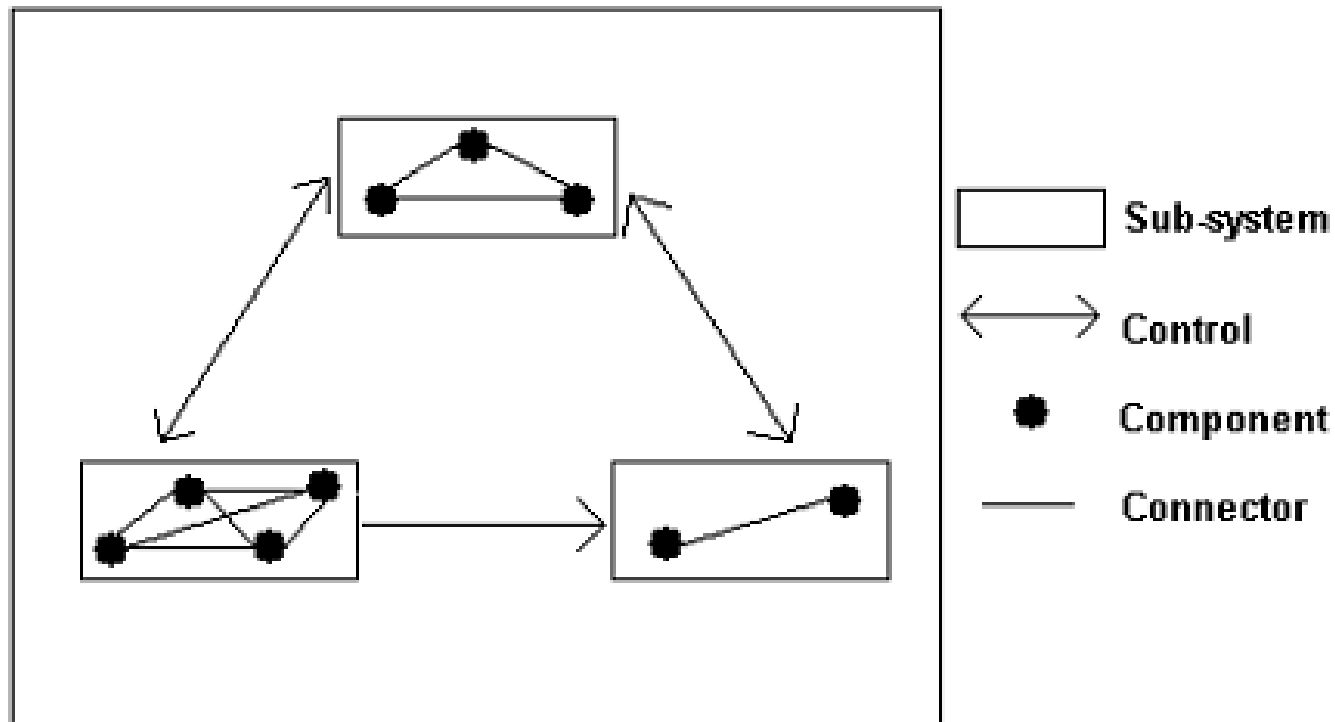
- Performance
  - Reliability
  - Portability
  - Usability
  - Security
  - Testability
- Maintainability
  - Adaptability
  - Modifiability
  - Usability
  - Scalability.

# Software Architecture: Bridging Requirements and Implementation

# Relationship between software requirements and architecture



# Box-and-line diagram showing an architecture design



# Software architecture definitions



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





A complete software architecture specification must describe **not only the elements and connections between elements** *but also* **the constraints and runtime behaviors** so that developers know what must be implemented and how it should be implemented.

# Software architect's tasks:

Perform system static partitioning and 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, or control flow orchestration, or message dispatching

Perform **trade-off analysis on quality attributes** and other non-functional requirements **during the selection of architecture styles**

**Map the software requirements specification to the software architecture** and guarantee that the software architecture **satisfies functional and non-functional requirements**

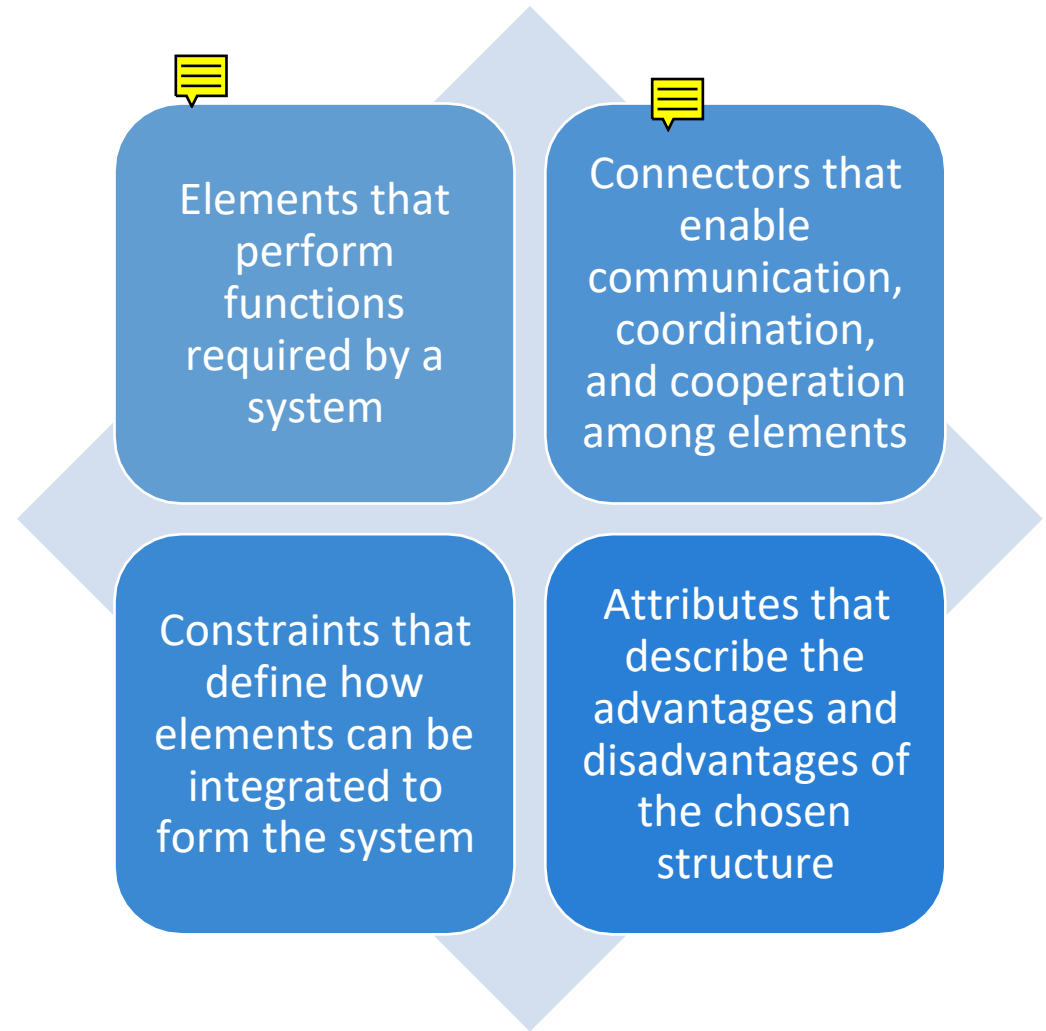
# Architectural Styles

An architectural style abstracts the common properties of a family of similar designs

It contains a **set of rules, constraints, and patterns** of how to structure a system into a set of elements and connectors

It governs the overall structure design pattern of constituent element types and their runtime interaction of flow control and data transfer

# The key components of an architectural style



# Architectural styles

---

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 (Interaction Oriented)

---

# Quality Attributes

---

Each architectural style 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

---

Quality attributes are identified in the requirement analysis process

Implementation  
attributes (not  
observable at  
runtime)

**Interoperability**

**Maintainability &  
extensibility**

**Testability**



Implementation  
attributes (not  
observable at  
runtime)

**Portability**

**Scalability**

**Flexibility**

Runtime  
attributes  
(observable  
at runtime)

**Availability**

**Security**

**Performance**




**Usability**



## Business attributes

- Time to market 🗨️
- Cost 🗨️
- Lifetime

## Some tradeoffs to consider

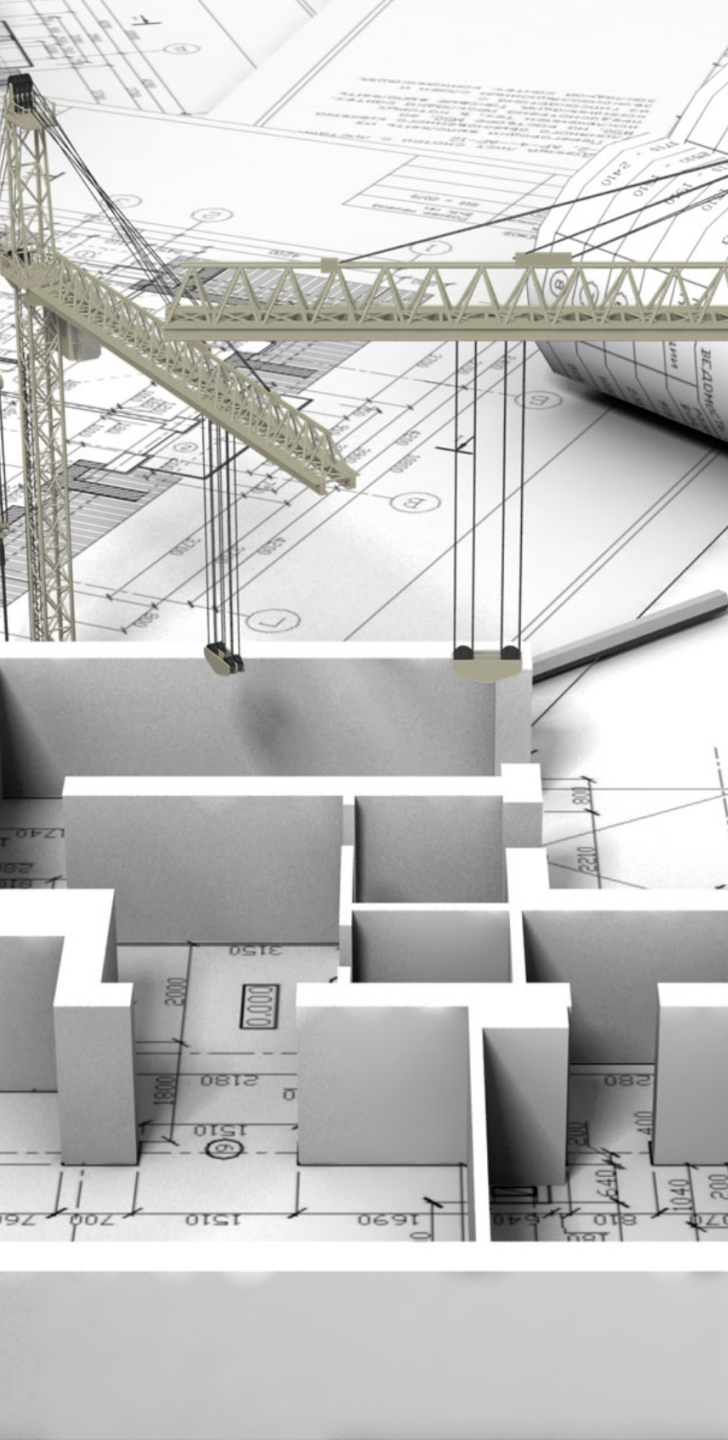
- Tradeoff between space and time 
- Tradeoff between reliability and performance 
- Tradeoff between scalability and performance 





# Software Architecture Design Guidelines

---

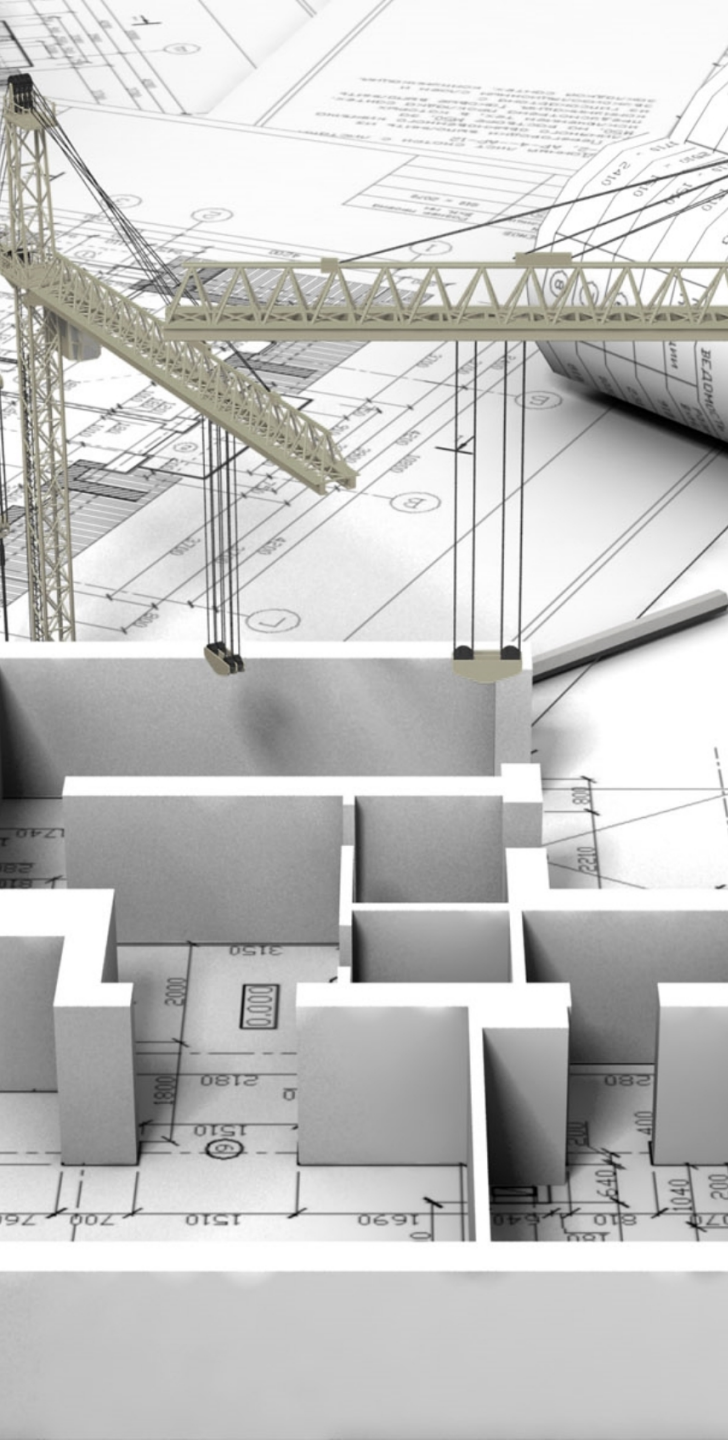


# Think of *what* to do before thinking of *how* to do it

---

- 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

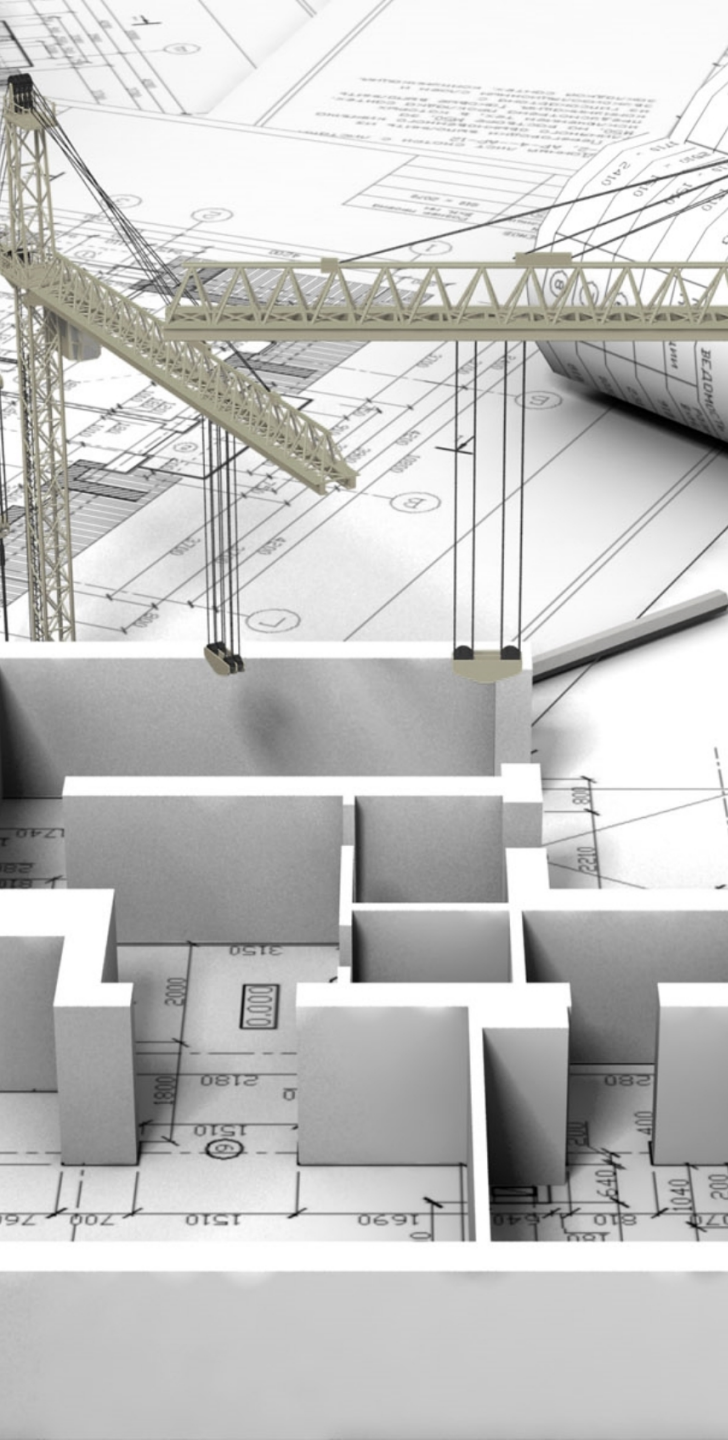




## Think of abstract design before thinking of concrete design

---

- 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

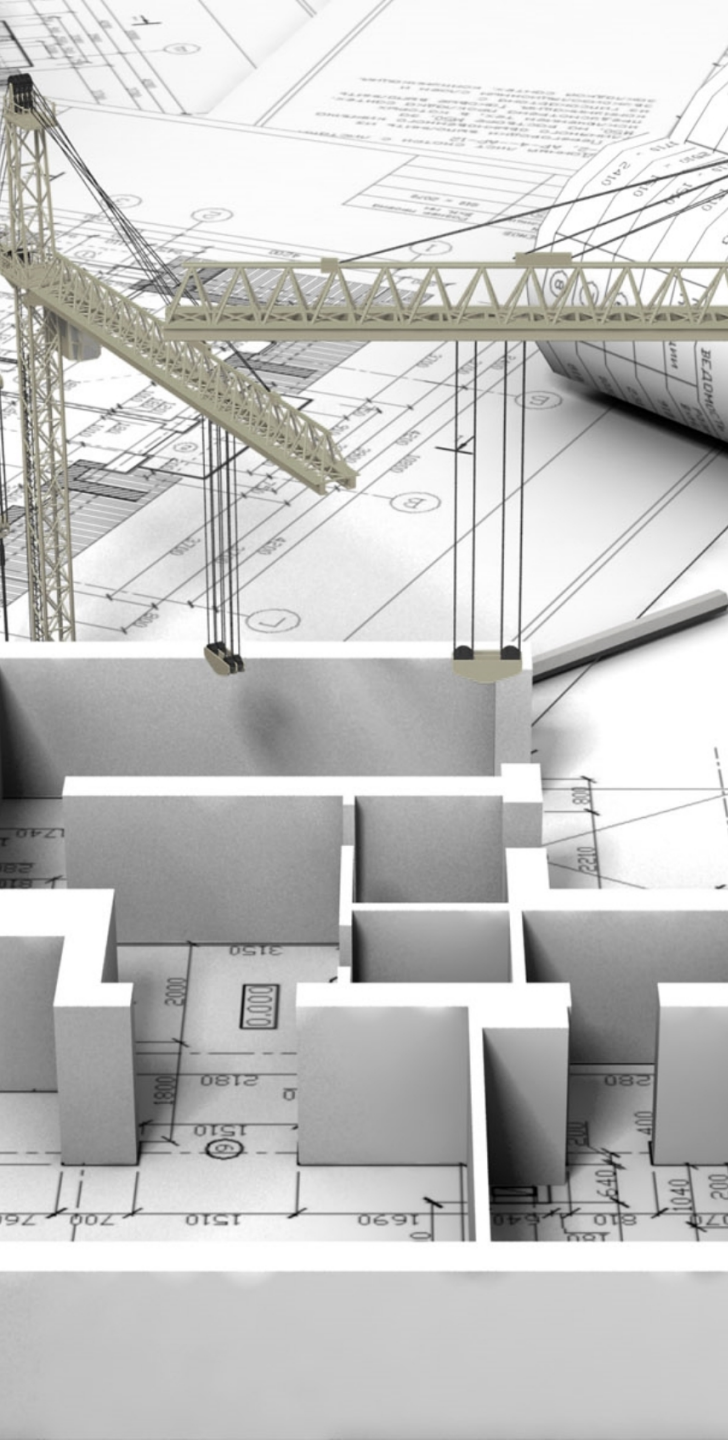


## Think of non-functional requirements earlier

---

- 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

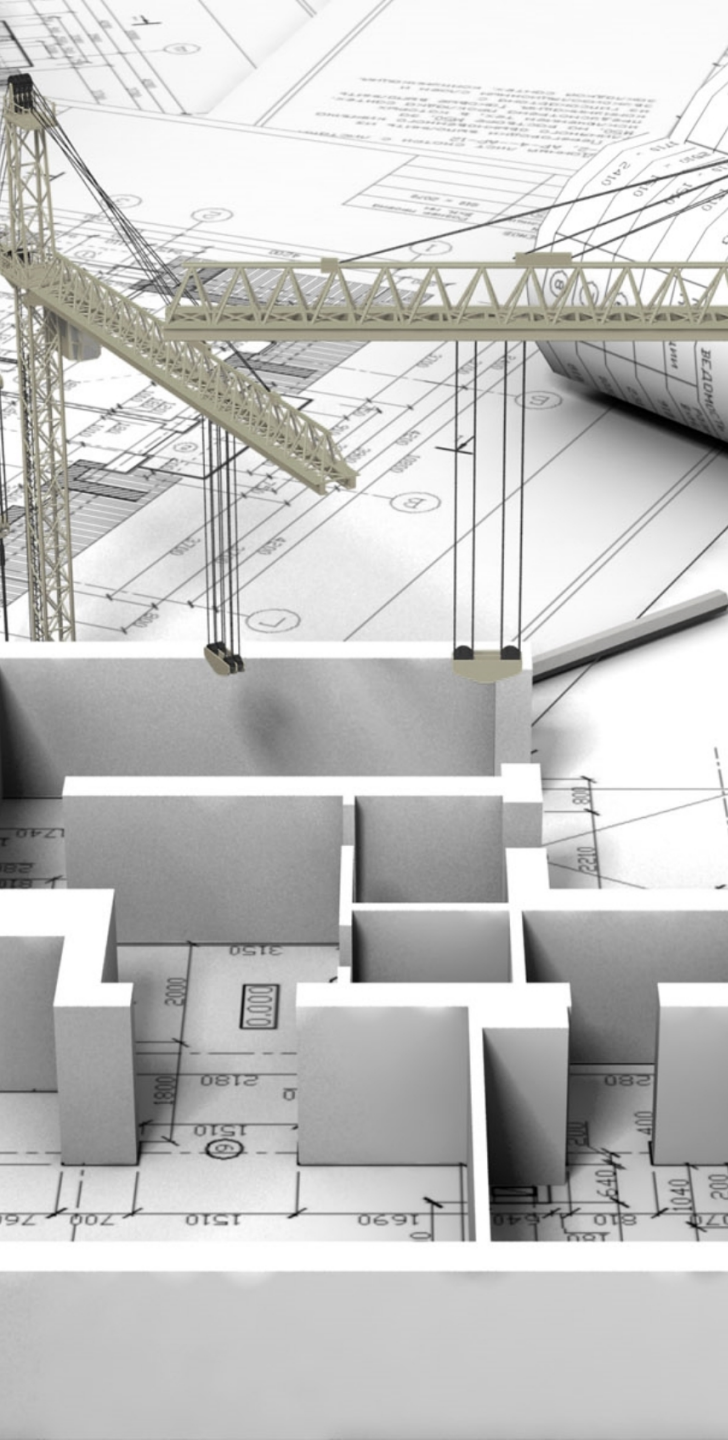




# Think of software reusability and extensibility as much as possible

---

- 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



## 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

# 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 (cont.)

---

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 (cont.)

---

There are always quality attribute tradeoffs between different styles

---

Hence, how to keep a balance on quality attributes is an important design issue