SOFTWARE ARCHITECTURE

SENG401

WINTER 2019

DR. EHSAN MOHAMMADI

WHAT IS SOFTWARE ARCHITECTURE?

ARCHITECTURE

Architecture is

- All about communication
- What ‘parts” are there?
- How do the ‘parts’ fit together?

Architecture is NOT

- About development
- About algorithms
- About data structures

ARCHITECTURE

Architecture Description:

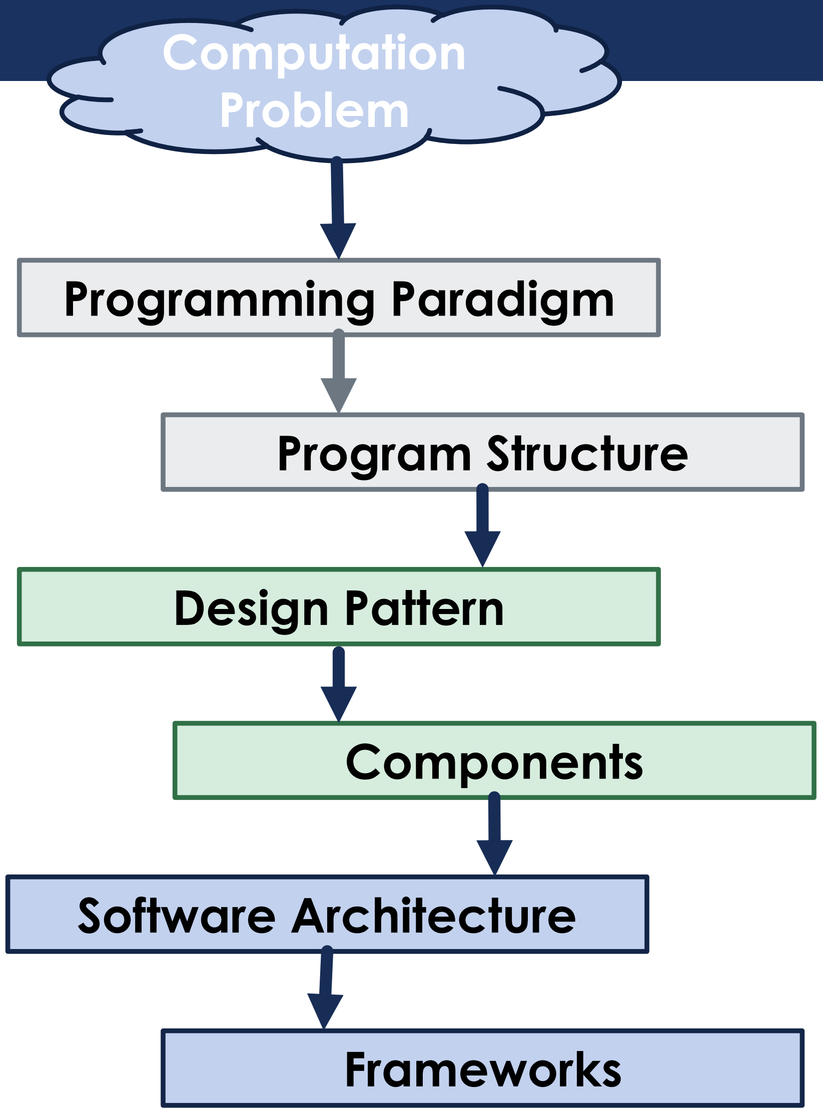
“Fundamental **concepts** or **properties** of a system in its environment embodied in its **elements**, **relationships**, and in the principles of its **design** and **evolution**”

42010-2011-ISO/IEC/IEEE Systems and software Engineering

WHAT IS SOFTWARE ARCHITECTURE

- The conceptual fabric that defines a system
  - All architecture is design but not all designs are architecture
- Architecture focuses on those aspects of a system that would be **difficult to change** once the system is built
- Architectures capture three primary dimensions:
  - **Structure**
  - **Communication**
  - **Non-functional requirements (Quality Attributes)**

THE BIG PICTURE



ARCHITECTURE VS. CODING

Architecture	Programming
Interactions among parts	Implementation of parts
Structural properties	Computational properties
Declarative	Operational
Mostly static	Mostly dynamic
System-level performance	Algorithmic performance
Outside module boundary	Inside module boundary

SOFTWARE ARCHITECTURE

- Software systems are constructed to satisfy organizations' **business goals**. The architecture is a **bridge** between those business goals and the final concrete resulting system.

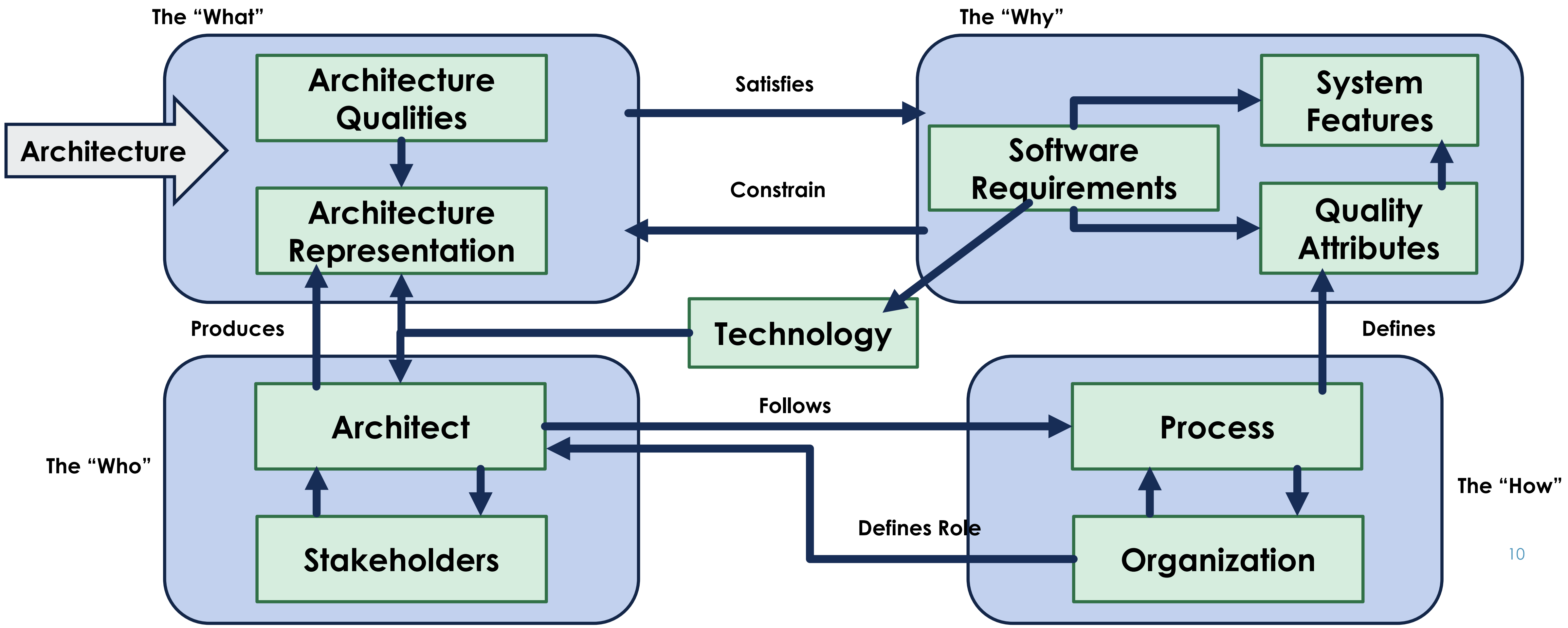
- The software architecture of a system is the set of **structures** needed to **reason** about the system, which comprise **software elements**, **relations** among them, and **properties** of both.

SOFTWARE ARCHITECTURE

The architecture of a software system:

- Defines the system in terms of **components** and **interaction** among components
- Shows correspondence between **requirements** and **elements** of the constructed system
- Addresses system-level **properties** such as: scale, capacity, throughput, consistency, compatibility
- Describes and integrates different **views** on the system

THE DOMAIN OF ARCHITECTURE



SOFTWARE ARCHITECTURE

- **Architecture is a Set of Software Structures**
  - A **Structure** is simply a set of **elements** held together by a **relation**
  - A Structure is architectural if it supports **reasoning** about the system and the system's properties. The reasoning should be about an **attribute** of the system that is important to some stakeholder. (e.g. functionality, availability, responsiveness)
  - Architectural Structures:
    - **Modules:** implementation units, static, (e.g. output of object-oriented analysis and design)
    - **Component-and-Connector (C&C) structures:** runtime structures, dynamic, focus on the interaction of elements.
    - **Allocation structures:** describe mapping from software structures to the system's environment

SOFTWARE ARCHITECTURE

- **Architecture is an Abstraction**
  - An architecture is foremost an **abstraction** of a system that selects certain details and suppresses others.
  - Architecture is concerned with the **public** side of interface (not private side; details having to do solely with internal implementation are not architectural)

SOFTWARE ARCHITECTURE

- **Architecture includes Behavior**
  - The **behavior** of each element is part of the architecture as that behavior can be used to reason about the system.
  - Behavior embodies how elements **interact** with each other

SOFTWARE ARCHITECTURE

- Structures and Views
  - Modern systems are frequently too complex to grasp all at once. We take **views** of architecture to communicate about structures.
  - A **view** is a representation of a coherent set of architectural elements, as written by and read by system stakeholders.
  - A view consists of a representation of a set of elements and the relations among them.
  - A view is a representation of a structure. (e.g. a module structure and a module view)

SOFTWARE ARCHITECTURE

Architectural View:

- A **view** is a representation or description of the entire system from a single perspective.
- An architectural view is an abstraction of a system, covering particular concerns, and omitting entities that are not relevant to this perspective.

STRUCTURES

- Types of Structure:
  - Module Structures
  - Component-and-Connector Structures
  - Allocation Structures

STRUCTURES

- **Module Structures**
  - Embody decisions as to how the system is to be structured as a set of **code** or **data units**
  - Represent a **static** way of considering the system
  - Allow us to answer questions such as:
    - 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 and depend on?
    - What modules are related to other modules by generalization or specialization relationships?
  - **Useful module structures:** Decomposition structure, Uses structure, Layer structure, Class structure, Data Model

STRUCTURES

- **Component-and-Connector Structures**
  - Embody decisions as to how the system is to be structured as a set of **elements** that have **runtime behavior** (components; such as: services, peers, clients, servers, filters) and **interactions** (connectors; such as: call-return, process synchronization operators, pipes)
  - Allow us to answer questions such as:
    - What are the major executing components and how do they interact at runtime?
    - 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?
    - Can the system's structure change as it executes and how?
  - **Useful C&C structures:** Service structure, Concurrency structure

STRUCTURES

- **Allocation Structures**
  - Embody decisions as to how the system will relate to **non-software structures** in its environment (such as CPUs, file systems, networks, development teams, etc.)
  - Help us to answer questions such as:
    - What processor does each software element execute on?
    - In what directories or files is each element stored during development, testing, and system building?
    - What is the assignment of each software element to development teams?
  - **Useful Allocation structures:** Deployment structure, Implementation structure, Work assignment structure

ARCHITECTURAL PATTERNS

- An Architectural Pattern delineates the element types and their forms of interaction used in solving a problem.
- Patterns can be characterized according to the type of architectural elements they use.
- Common **module** type pattern:
  - Layered pattern
- Common **C&C** type patterns:
  - Shared data pattern
  - Client-Server pattern
- Common **allocation** patterns:
  - Multi-tier pattern
  - Competence Center pattern
  - Platform pattern

WHY IS SOFTWARE ARCHITECTURE IMPORTANT?

WHY IS SOFTWARE ARCHITECTURE IMPORTANT?

- Software architecture is the set of design decisions which, if made incorrectly, may cause your project to be cancelled.

E. Woods

WHY IS SOFTWARE ARCHITECTURE IMPORTANT?

1. An architecture will inhibit or enable a system’s driving **quality attributes**

- **high performance:** time-based behavior of elements, use of shared resources, frequency and volume of communication
- **Modifiability:** assigning responsibilities to elements
- **Security:** manage and protect inter-element communications
- **Scalability:** localize the use of resources, avoid hard-coding in resource assumptions or limits
- **Reusability:** restrict inter-element coupling

A good architecture is necessary, but not sufficient, to ensure quality.

WHY IS SOFTWARE ARCHITECTURE IMPORTANT?

2. The decisions made in an architecture allow you to reason about and manage **change** as the system evolves. (Modifiability)

Modifiability is the ease with which changes can be made to a system. Possible changes:

- **Local**: modifying a single element
- **Non-local**: multiple element modifications; leaves the underlying architecture intact
- **Architectural:** affects the fundamental ways in which the elements interact with each other and will probably require changes all over the system.

An effective architecture is one in which the most common changes are local.

WHY IS SOFTWARE ARCHITECTURE IMPORTANT?

3. The analysis of an architecture enables **early prediction** of a system's qualities.

It is possible to make quality predictions about a system based solely on an evaluation of its architecture. If we know that certain kinds of **architectural decisions** lead to certain **quality attributes** in a system, then we can make those decisions and rightly expect to be rewarded with the associated quality attributes.

WHY IS SOFTWARE ARCHITECTURE IMPORTANT?

4. A documented architecture **enhances communication** among stakeholders.

Each stakeholder of a software system (e.g. customer, user, project manager, coder, tester, and so on) is concerned with different **characteristics** of the system that are affected by its architecture.

Architecture provides a **common language** in which different concerns can be **expressed**, **negotiated**, and **resolved** at a level that is intellectually manageable even for large, complex systems.

WHY IS SOFTWARE ARCHITECTURE IMPORTANT?

5. The architecture is a carrier of the earliest and hence most **fundamental**, **hardest-to-change** design decisions.

An Architecture design can be viewed as **a set of decisions**. The early design decisions constrain the decisions that follow, and changing these decisions has enormous ramifications. (e.g. number of processors, layered software?, synchronous or asynchronous communication, dependency, information encryption, operating system, communication protocols, …)

WHY IS SOFTWARE ARCHITECTURE IMPORTANT?

6. An architecture defines a set of **constraints** on subsequent implementation.

An implementation exhibits an architecture as the set of **prescribed elements**, with prescribed **interactions** and **responsibilities**. Each of these prescriptions is a **constraint** on the implementer.

Example: Performance in banking transactions

WHY IS SOFTWARE ARCHITECTURE IMPORTANT?

7. The architecture dictates the **structure of an organization**, or vice versa.

Not only does architecture prescribe the structure of the system being developed, but that structure becomes engraved in the structure of the development project

Work-breakdown structure: method of dividing up the labor in a large project that assigns different groups different portions of the system to construct.

Once the architecture has been agreed upon, it becomes very costly to significantly modify it.

WHY IS SOFTWARE ARCHITECTURE IMPORTANT?

8. An architecture can provide the basis for **evolutionary prototyping**.

Once an architecture has been defined, it can be analyzed and prototyped as a **skeletal system**. A skeletal system is one in which at least some of the infrastructure (e.g. how the elements initialize, communicate, share data, access resources, report errors, log activity, and so forth) is built before much of the system's functionality has been created.

Example: Systems built as plug-in architectures

WHY IS SOFTWARE ARCHITECTURE IMPORTANT?

9. An architecture is the key artifact that allows the architect and project manager to reason about **cost** and **schedule**.

Cost and schedule estimates are important tools for the project manager both to acquire the necessary resources and to monitor progress on the project, to know if and when a project is in trouble. One of the duties of an architect is to help the project manager create cost and schedule estimates early in the project life cycle.

The best cost and schedule estimates emerge from a consensus between the **top-down** estimates (by the architect) and the **bottom-up** estimates (by developers).

WHY IS SOFTWARE ARCHITECTURE IMPORTANT?

10. An architecture can be created as a transferable, reusable **model** that forms the heart of a product line.

While **code reuse** provides a benefit, **reuse of architectures** provides tremendous leverage for systems with similar requirements. Not only can code be reused, but so can the requirements that led to the architecture in the first place, as well as the **experience** and **infrastructure** gained in building the reused architecture.

WHY IS SOFTWARE ARCHITECTURE IMPORTANT?

11. Architecture-based development focuses attention on the **assembly of components**, rather than simply on their creation.

Earlier software paradigms focused on programming. Architecture-based development focuses on composing or assembling elements that are likely to have been developed separately, even independently. This composition is possible because the architecture defines the elements that can be incorporated into the system.

Payoffs: Decreased time to market, Increased reliability, Lower cost, Flexibility

WHY IS SOFTWARE ARCHITECTURE IMPORTANT?

12. By restricting design alternatives, architecture channels the creativity of developers, reducing **design** and **system complexity**.

Minimizing the design complexity of the system is the direct result of choosing an architecture which restricts our choices of elements and their interactions.

Architectural patterns guide the architect and focus the architect on the **quality attributes** of interest.

WHY IS SOFTWARE ARCHITECTURE IMPORTANT?

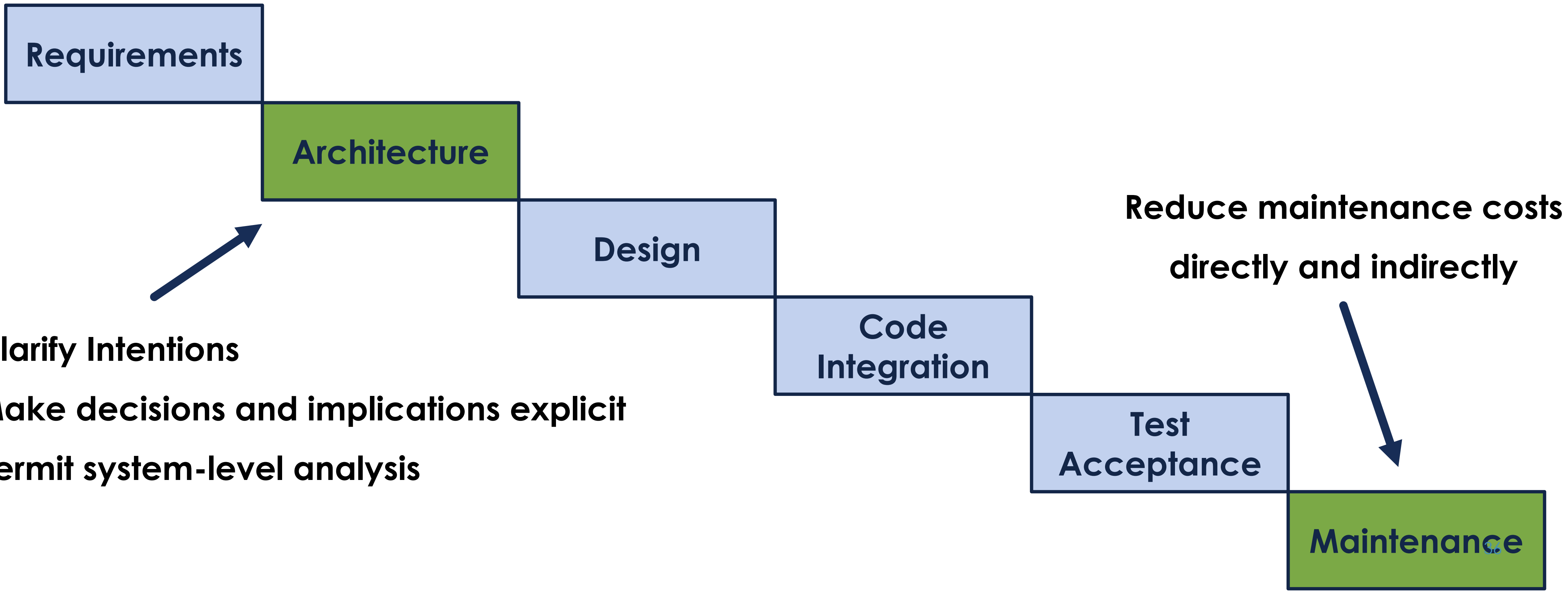
13. An architecture can be the foundation for **training** a new team member.

The architecture, including a description of how the elements interact with each other to carry out the required behavior serves as the first introduction to the system for new project members.

Module views are excellent for showing someone the structure of a project.

Component-and-Connector view are excellent for explaining how the system is expected to work and accomplish its job.

BENEFITS OF ARCHITECTURE



- Clarify Intentions
- Make decisions and implications explicit
- Permit system-level analysis

CONCLUSIONS

- As both the **scale** and **complexity** of the software systems increase, algorithms and data structures become less important than getting the right structure for the system.
- Software architecture describes the structure of the solution both abstractly and concretely

However …

- There are many different ways to specify architecture. Choosing the right architecture for a problem can be a difficult problem on itself.

REFERENCES

- Software Architecture in Practice; Len Bass, Paul Clements, Rick Kazman