What is Software Architecture?

- **Definition:**
  - A software system’s architecture is the set of principal design decisions about the system
  - Software architecture is the blueprint for a software system’s construction and evolution
  - Design decisions encompass every facet of the system under development
    - Structure
    - Behavior
    - Interaction
    - Non-functional properties
What is “Principal”?

- “Principal” implies a degree of importance that grants a design decision “architectural status”
  - It implies that not all design decisions are architectural
  - That is, they do not necessarily impact a system’s architecture
- How one defines “principal” will depend on what the stakeholders define as the system goals

Other Definitions of Software Architecture

- Perry and Wolf
  - Software Architecture = \{ Elements, Form, Rationale \}
    - what how why
- Shaw and Garlan
  - Software architecture [is a level of design that] involves
    - the description of elements from which systems are built,
    - interactions among those elements,
    - patterns that guide their composition, and
    - constraints on these patterns.
- Kruchten
  - Software architecture deals with the design and implementation of the high-level structure of software.
  - Architecture deals with abstraction, decomposition, composition, style, and aesthetics.
Temporal Aspect

- Design decisions are and unmade over a system’s lifetime
  → Architecture has a temporal aspect
- At any given point in time the system has only one architecture
- A system’s architecture will change over time

Prescriptive vs. Descriptive Architecture

- A system’s *prescriptive architecture* captures the design decisions made prior to the system’s construction
  - It is the *as-conceived* or *as-intended* architecture
- A system’s *descriptive architecture* describes how the system has been built
  - It is the *as-implemented* or *as-realized* architecture
As-Designed vs. As-Implemented Architecture

Software Architecture: Foundations, Theory, and Practice

Richard N. Taylor, Nenad Medvidovic, and Eric M. Dashofy
As-Designed vs. As-Implemented Architecture

- Which architecture is “correct”?
- Are the two architectures consistent with one another?
- What criteria are used to establish the consistency between the two architectures?
- On what information is the answer to the preceding questions based?

Architectural Evolution

- When a system evolves, ideally its prescriptive architecture is modified first
- In practice, the system – and thus its descriptive architecture – is often directly modified
- This happens because of
  - Developer sloppiness
  - Perception of short deadlines which prevent thinking through and documenting
  - Lack of documented prescriptive architecture
  - Need or desire for code optimizations
  - Inadequate techniques or tool support
Architectural Degradation

- Two related concepts
  - Architectural drift
  - Architectural erosion
- Architectural drift is introduction of principal design decisions into a system’s descriptive architecture that are not included in, encompassed by, or implied by the prescriptive architecture but which do not violate any of the prescriptive architecture’s design decisions.
- Architectural erosion is the introduction of architectural design decisions into a system’s descriptive architecture that violate its prescriptive architecture.

Architectural Recovery

- If architectural degradation is allowed to occur, one will be forced to recover the system’s architecture sooner or later.
- Architectural recovery is the process of determining a software system’s architecture from its implementation-level artifacts.
- Implementation-level artifacts can be:
  - Source code
  - Executable files
  - Java .class files
Implementation-Level View of an Application

Complex and virtually incomprehensible!
Deployment

- A software system cannot fulfill its purpose until it is deployed
  - Executable modules are physically placed on the hardware devices on which they are supposed to run
- The deployment view of an architecture can be critical in assessing whether the system will be able to satisfy its requirements
- Possible assessment dimensions
  - Available memory
  - Power consumption
  - Required network bandwidth

A System’s Deployment Architectural Perspective
Software Architecture’s Elements

- A software system’s architecture typically is not (and should not be) a uniform monolith
- A software system’s architecture should be a composition and interplay of different elements
  - Processing
  - Data, also referred as information or state
  - Interaction

Components

- Elements that encapsulate processing and data in a system’s architecture are referred to as software components
- **Definition**
  - A *software component* is an architectural entity that
    - encapsulates a subset of the system’s functionality and/or data
    - restricts access to that subset via an explicitly defined interface
    - has explicitly defined dependencies on its required execution context
  - Components typically provide application-specific services
Connectors

- In complex systems *interaction* may become more important and challenging than the functionality of the individual components

**Definition**
- A *software connector* is an architectural building block tasked with effecting and regulating interactions among components
- In many software systems connectors are usually simple procedure calls or shared data accesses
  - Much more sophisticated and complex connectors are possible!
- Connectors typically provide application-independent interaction facilities

Examples of Connectors

- Procedure call connectors
- Shared memory connectors
- Message passing connectors
- Streaming connectors
- Distribution connectors
- Wrapper/adaptor connectors
Configurations

- Components and connectors are composed in a specific way in a given system’s architecture to accomplish that system’s objective
- **Definition**
  - An architectural configuration, or topology, is a set of specific associations between the components and connectors of a software system’s architecture

**An Example Configuration**
Architectural Styles

- Certain design choices regularly result in solutions with superior properties
  - Compared to other possible alternatives, solutions such as this are more elegant, effective, efficient, dependable, evolvable, scalable, and so on
- **Definition**
  - An architectural style is a named collection of architectural design decisions that
    - are applicable in a given development context
    - constrain architectural design decisions that are specific to a particular system within that context
    - elicit beneficial qualities in each resulting system

Architectural Patterns

- **Definition**
  - An architectural pattern is a set of architectural design decisions that are applicable to a recurring design problem, and parameterized to account for different software development contexts in which that problem appears
  - A widely used pattern in modern distributed systems is the three-tiered system pattern
    - Science
    - Banking
    - E-commerce
    - Reservation systems
Three-Tiered Pattern

- **Front Tier**
  - Contains the user interface functionality to access the system’s services

- **Middle Tier**
  - Contains the application’s major functionality

- **Back Tier**
  - Contains the application’s data access and storage functionality

Architectural Models, Views, and Visualizations

- **Architecture Model**
  - An artifact documenting some or all of the architectural design decisions about a system

- **Architecture Visualization**
  - A way of depicting some or all of the architectural design decisions about a system to a stakeholder

- **Architecture View**
  - A subset of related architectural design decisions
Architectural Processes

- Architectural design
- Architecture modeling and visualization
- Architecture-driven system analysis
- Architecture-driven system implementation
- Architecture-driven system deployment, runtime redeployment, and mobility
- Architecture-based design for non-functional properties, including security and trust
- architectural adaptation

Stakeholders in a System’s Architecture

- Architects
- Developers
- Testers
- Managers
- Customers
- Users
- Vendors