Class Goals

- The students will be able to define the concepts of software architecture and design.
- The students will be able to describe three different viewtypes needed to develop a software architecture.
- The students will have an appreciation on the notation required to document each viewtype.
After requirements have been analyzed and specified, software architecture and design are the first of three technical activities (architecture/design, code generation, testing) that are required to build and verify the software.
Software Architecture

“A software architecture for a system is the structure or structures of the system, which consist of elements, their externally visible properties, and the relationships among them”

Software Architecture vs. Software Design

Architecture feeds the design activity. Many design decisions are left unbound by the architecture and are left to the detailed designers. The architecture defines constraints on downstream activities, and those activities must produce artifacts (detailed design and code) that are compliant with the architecture, but architecture does not define implementation.
Uses of Architecture Documentation

- Architecture serves as a means of education
- Architecture serves as a main means of communication among relevant stakeholders (architect and requirements engineers, designers, implementers, testers, maintainers, managers, QA team, etc.)
- Architecture serves as a basis for system analysis and design
A view is a representation of a set of system elements and the relationships associated with them.

Documenting an architecture means to document the relevant views under which a system can be observed.

A software architecture documentation package is a set of one or more view documents and documentation that explains how the views relate to each other, introduces the package to its readers, and guides them through it.

Different views expose different quality attributes to different degrees.
Architecture Document for a View

The documentation for a view contains:

- Graphical representation that depicts the primary system elements and its relationships
- Element catalogue that defines elements and its properties
- A specification of the element’s interfaces and behaviour
- Rationale and design information
Documentation for all Views

- Documentation that applies to all of the views contains:
  - An introduction to the entire package
  - Information describing how the views relate to each other, and to the system as a whole
  - Constraints and rationale for the overall architecture
  - Management information needed to maintain the whole package
“4+1” Architectural Views – A Model

- **Logical view**: supports behavioral requirements or the services the system must provide its users

- **Process view**: addresses concurrency and distribution, system integrity, and fault tolerance

- **Development view**: focuses on the organization of the software modules in the software development environment

- **Physical view**: takes into account the system’s requirements such as system availability, reliability, performance, and scalability

Kruchten (1995) from rational Software Corporation
Architecture Views: Siemens – Another Model

- **Conceptual view**: describes the system in terms of its major design elements and the relationships among them.
- **Module interconnection view**: encompasses two orthogonal views: functional decomposition and layers.
- **Execution view**: describes the dynamic view of the system.
- **Code view**: describes how the source code, binaries, and libraries are organized in the development environment.

Soni, Nord, and Hofmeister from Siemens Corporate Research (1995)
Viewtypes represent the perspectives that an architect must consider when designing a system:

- Module viewtype
- Component and Connector viewtype
- Allocation viewtype

Each viewtype has associated styles. A style is a specialization of a viewtype and reflects recurring patterns of interaction, independent of any particular system.

Within the confines of a style, choices need to be made on how the elements in a style are bound to elements in a system, and these are called views.
Rules for a Sound Architecture Documentation

- Rule 1: Write the documentation from the reader’s perspective
- Rule 2: Avoid unnecessary repetition
- Rule 3: Avoid ambiguity
- Rule 4: Use a standard organization
- Rule 5: Record rationale
- Rule 6: Keep documentation current but not too current
- Rule 7: Review documentation for fitness of purpose
Module Viewtype

- A module is a code unit that implements a set of responsibilities. A module can be a class, a collection of classes, or any decomposition of the code unit. The module viewtype documents the principal implementation units, or modules of a system, together with the relationships among these modules.

- **Elements**: module which is an element of software that provides a coherent unit of functionality

- **Relations**: is-part-of, depends-on, is-a relationships

- **Properties of elements**: name, responsibilities, implementation information
Component and Connector Viewtype

- Component-and-connector views define models consisting of elements that have some runtime presence, such as processes, objects, clients, servers, and data sources. They include the pathways of interaction, such as communication links, and protocols, information flows, and access to shared storage.

- **Elements**: component types: principal processing units and data stores

- **Relations**: Attachments: component ports associated with specific connector roles, component port \( p \), attached to a connector role \( r \), if the component interacts over the connector, using the interface described by \( p \), and conforming to expectations described by \( r \)

- **Properties of elements**: Component (name, type, other properties) and Connector (name, type, other properties)
... Continued Component and Connector Viewtype

Key

Component Types

- Database
- Application
- Database
- Client
- Server

Connector Types

- Attachment
- Client/Server – Request/Reply
- Database Access
- Publish-Subscribe
The Allocation Viewtype

- The allocation viewtype presents a mapping of the software architecture onto its environment. It presents a mapping from the elements of either a module or a component-and-connector style onto elements of the environment. Three common allocation viewtype styles can be identified:
  - The deployment style describes the mapping of the components and connectors onto the hardware on which the software executes
  - The implementation style describes the mapping of modules onto a file system that contains these modules
  - The work assignment style describes the mapping of modules onto the people, groups, or teams tasked with the development of the modules
... Continued The Allocation Viewtype

- **Elements**: Software element and environment element
- **Relations**: Allocated-to. A software element is allocated to an environment element
- **Properties of elements**: A software element has required properties. An environmental element has provided properties that need to be matched
Class Exercise

- 10 minutes
- Discuss any issues associated with Design and Architecture in both project teams