Introduction to Software Architecture

Version 5.0

Wyyzzk, Inc.

Lesson Goal & Objectives

- Understand the purpose of software architecture.
- Upon completion of the lesson, the participant will be able to:
  - Describe the purpose of software architecture
  - Describe the parts of the 4+1 architecture

Wyyzzk, Inc.

Lesson Outline

- Introduction to Software Architecture
  - What is software architecture?
  - 4+1 views of a software architecture
  - Architecture and Patterns
  - Documentation
  - Design
  - Lifecycle

Wyyzzk, Inc.

Introduction to Software Architecture

- In this course, we will be considering the structures of a software system, from the enterprise down to the individual project level

What is Software Architecture?

"Software architecture is concerned with the organization of software systems, the selection of components from which they are composed, the interactions between those components, the composition of interacting components into progressively larger subsystems, and the overall patterns that guide these compositions. It is concerned not only with the systems structure, but also with its functionality, performance, design, selection among alternatives, and comprehensibility."

Mary Shaw

Software Architecture: Perspectives on an Emerging Discipline

Structure

- Architecture deals with the structure and systems of something:
  - outside appearance
  - major subsystems
  - how subsystems communicate
  - the nature of the communication

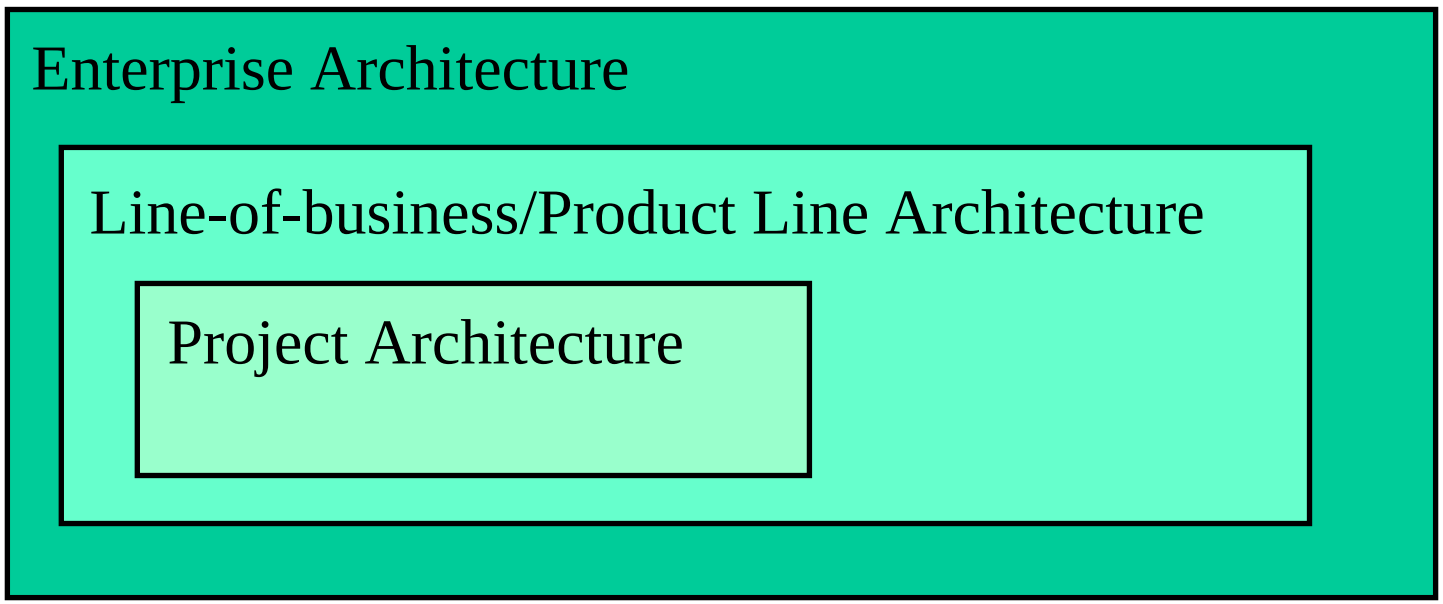
Wyyzzk, Inc.

Point of View

- One point of view is not enough
  - Users
  - Developers
  - Materials
  - Site

Complexity

- Complexity varies with:
  - Time
  - Hierarchy



Wyyzzk, Inc.

User View of Building

- The user view of a building is typically expressed as a series of drawings:
  - A drawing of the outside of the building from every direction (elevation)
  - A sketch of room layouts on each floor
- Sometimes usage scenarios for the building are written as well

Wyyzzk, Inc.

User View of Software

- This corresponds to a software project:
  - determining any look and feel standards the software must conform to
  - collecting requirements for the functionality of the software
  - writing scenarios for how the software will be used

Wyyzzk, Inc.

Not enough to Build

- This is not enough information to build a building
  - unless I have domain expertise that lets me “fill in the blanks”
- It is not enough information to build software either
- We need to look at the building and the software from some other viewpoints before we begin building

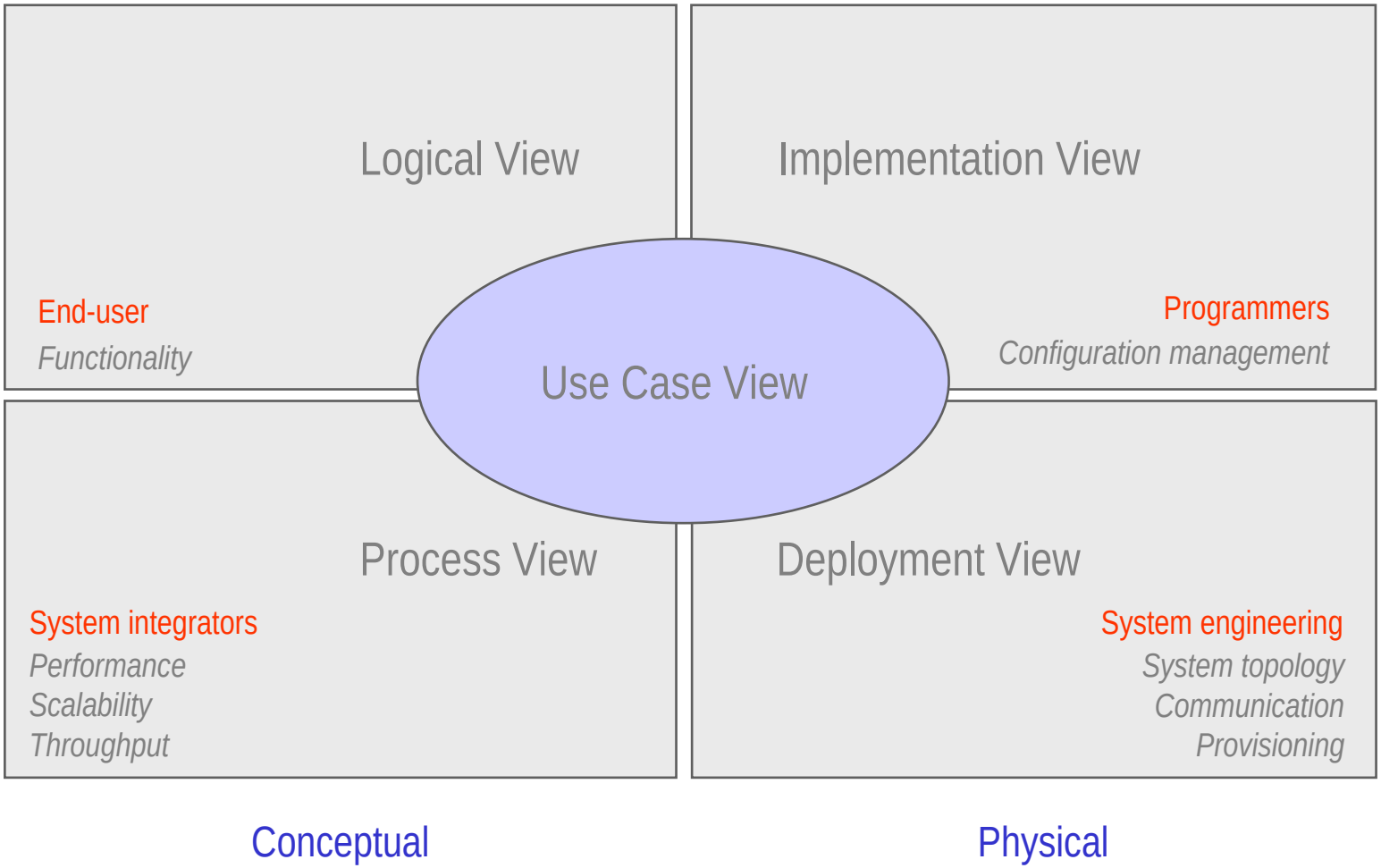
Wyyzzk, Inc.

4+1 Views

- To completely describe a software architecture, four views are needed:
  - The logical view to provide a static picture of the primary abstractions and their relationships
  - The development view to show how the code is organized into subsystems and libraries and the use of commercial off-the-shelf (COTS) software
  - The process view to show the processes and tasks
  - The physical view to show the processors, devices, and links in the operational environment
- Finally, a use case view explains how the other four views work together

Wyyzzk, Inc.

The “4+1 View” Model



Wyyzzk, Inc.

Logical View of Building

- For the building, lets look at the rooms in more detail
- We need to know:
  - The size and shape of the room
  - The functionality the room supports
    - bedroom, bathroom, gym, meeting room, computer room
  - Other rooms that connect to this room
  - The kind of connection between rooms
    - door, window, hallway

Logical View of Building (cont.)

- For each room we also need to know:
  - The number of walls, doors, and windows
  - The relationships between the walls, doors, and windows
- Other things required in this room, such as:
  - sink, or closet
- Special requirements such as:
  - air conditioning, light tight

Wyyzzk, Inc.

Logical View of Software

- The rooms correspond to software subsystems
- For each subsystem we need to know:
  - The outside appearance of the subsystem
    - its interface
  - The responsibilities (functionality) of the subsystem
  - Other subsystems this subsystem connects to
  - The kind of communication between subsystems

Logical View of Software (cont.)

- For each subsystem we also need to know:
  - The key classes that implement the subsystem
  - The relationships between the key classes
- Any special requirements, such as:
  - speed, size

Wyyzzk, Inc.

Process View of Building

- Things that are done inside this building
  - coordination between things done inside the building
    - using the kitchen, bathroom, and living room all at once when having a party
- Things that happen in this building that have to coordinate with things happening in other buildings

Wyyzzk, Inc.

Process View of Software

- If the system will contain multiple processes, we need to know:
  - Which processes we need
  - How the processes communicate
  - Where the processes are located
    - if there are multiple processors, show which processes run on each processor

Deployment View of Building

- The location of the building
- Its relationship to other buildings, parks, streets
- A description of any communication that happens between this building and other buildings

Wyyzzk, Inc.

Deployment View of Software

- If the software will run on multiple processors, we need to know:
  - The kind of processor
  - How many processors
  - Which processors communicate
  - How the processors communicate

Wyyzzk, Inc.

Implementation View of Building

- Materials to be used
- Work crews assigned
- Tools the crews need in each phase of construction
- Order of construction
- Sign off on inspections and permits

Wyyzzk, Inc.

Implementation View of Software

- The development platform and the target platform
  - including operating system, user interface, hardware
- The structure of the file system
- Any purchased software that will be used
- Any legacy systems that will be used
- Tools they will be using
- Any platforms the system will be ported to
  - user interface, operating system, hardware, etc.

More things to consider

- Plumbing, wiring, phone lines (macro design issues)
- The location and type of outlets, phone jacks, sinks (micro design issues)
- Blueprint
- Actual building corresponds

Wyyzzk, Inc.

More things to consider

- Communication, Persistence, Security (macro design issues)
- Design of classes and relationships (micro design issues)
- Class diagrams, component diagrams, deployment diagrams
- Code

Simplify

- Simplify the models to fit the context
- Not all systems require all views, especially for small systems
  - Single processor: drop deployment view
  - Single process: drop process view

Wyyzzk, Inc.

Architecture and Patterns

- Architecture is built from patterns
- Patterns have been used for centuries in many industries
  - designing clothing
  - designing buildings
  - designing cities
- You can solve a problem once, then provide the solution as a pattern that others can apply to solve the same problem over and over

What are patterns?

"Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice."

Christopher Alexander, "A Pattern Language"
a book on patterns in the design of buildings and cities

Examples

- Clothing
  - A - line skirt
  - raglan sleeve
- Buildings
  - French window
  - pocket door
- Cities
  - traffic circle
  - town square

Examples

- Enterprise
  - Service oriented
  - Event driven
  - Agents
- Software
  - Layered
  - Pipe and Filter
  - N-tier
  - Publish and Subscribe

Documenting the Architecture

- The architecture is documented in an architecture document
- The document includes:
  - A textual description of the architectural philosophy and the key driving requirements
  - Tradeoffs made and alternatives considered
  - A top-level diagram of the logical view
    - subsystems and key classes on class diagrams
  - Architecture-specific scenarios (use cases and use case diagrams)

Wyyzzk, Inc.

Documenting the Architecture (cont.)

- Top-level diagrams of the development (component diagram), process (component and deployment diagrams) and physical (deployment diagram) views
- The key mechanisms (usually class diagrams or text)

➢ The top level diagrams will be documented using Packages or Subsystems

Software Design

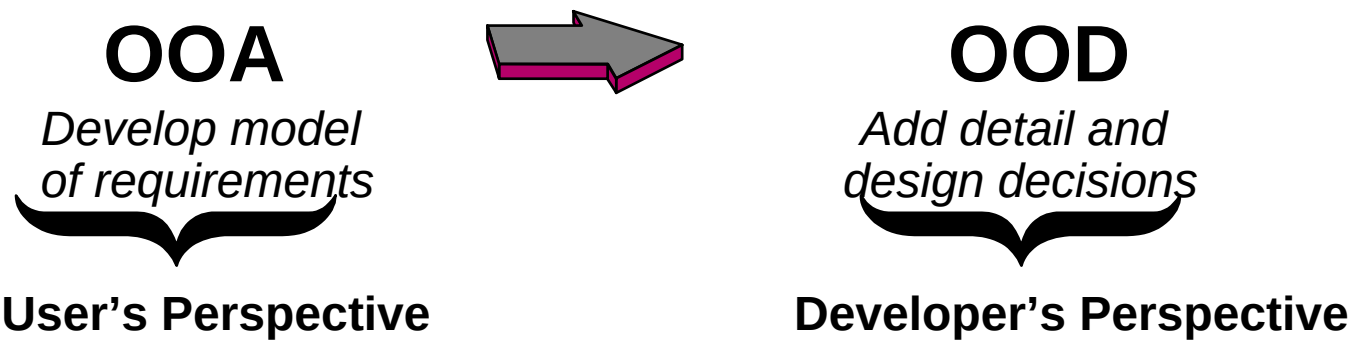
- Along with the architecture, we consider the macro design of the system
  - This deals with the larger issues and mechanisms of the system such as security, persistence, or distribution
  - This is not micro (detailed) design, but rather deals with the overall issues in the system

What is Software Design?

- Sometimes when people refer to design, they mean all of:
  - Architecture
    - the structures of the system
  - Macro design
    - system wide issues or mechanisms, also called tactical design
  - Micro design
    - design of classes and relationships, issues affecting a small part of the system
- Others just mean macro design.
  - In this class, design will refer to just macro design

What is Design?

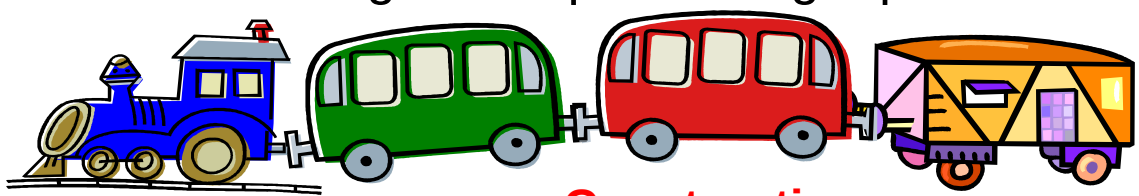
- Design is:
  - Figuring out HOW to build the system
  - Modifying the system for various constraints
  - Adding classes that supply system wide mechanisms



Wyyzzk, Inc.

Fitting design activities into lifecycle

- Architecture and Macro design issues are typically handled late in Elaboration
  - they affect the whole system
- What happens in Construction will depend on the size of the system.
  - For small systems, each Construction iteration will be mostly concerned with micro design, coding, and testing.
  - Larger systems will require a full life cycle of analysis, architecture, macro design, micro design, coding and testing for the software being developed during a particular iteration.



Wyyzzk, Inc.

For Example

- In an iteration, we decide to develop a whole subsystem
- From Elaboration you know:
  - The name of the subsystem
  - The interfaces supported by the subsystem
  - The responsibilities the subsystem supports
  - You also know about system wide macro issues
- Now you have to architect and design the subsystem before you can code it

Wyyzzk, Inc.

Summary

- Architecture describes the structure of the system you are developing
- You need to consider the system from many viewpoints to get a complete picture
  - logical
  - implementation
  - process
  - deployment
  - Use case

Wyyzzk, Inc.

Summary

- Subsystems and packages can be used to diagram the top level architecture views of your system
- Macro design is finding solutions for system wide issues such as security, persistence, and distribution.