


Slide 1




---

## Introduction to Software Architecture

Copyright © Wyzzk, Inc. 2004  
Version 5.0

slide 2




## 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

Copyright © Wyzzk, Inc. 2004  
Version 5.0

Design Patterns - 2

## Slide 3




### Lesson Outline

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

Copyright © Wyzzak, Inc. 2004  
Version 5.0 Design Patterns - 3

## Slide 4




### 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

Copyright © Wyzzak, Inc. 2004  
Version 5.0 Design Patterns - 4

## Slide 5



### 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**

Copyright © Wyzzak, Inc. 2004  
Version 5.0 Design Patterns - 5

## Slide 6



### Structure

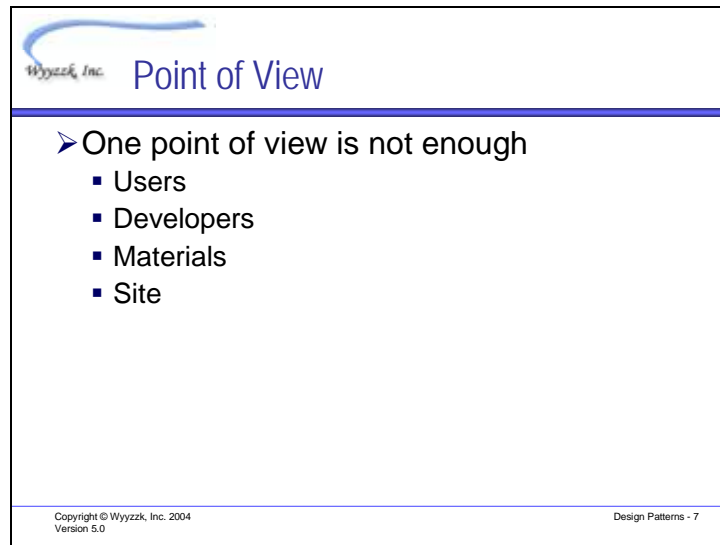
- Architecture deals with the structure and systems of something:
  - outside appearance
  - major subsystems
  - how subsystems communicate
  - the nature of the communication

Copyright © Wyzzak, Inc. 2004  
Version 5.0 Design Patterns - 6

The architecture of anything is concerned with structure, systems, and subsystems. Consider buildings – being physical it is easy to see the parts of the architecture. Subsystems are things like the room layout, plumbing, gas pipes, electric wiring, telephone wiring, network wiring, and sound system wiring.

In software we consider the user interface (the outside appearance), the subsystems, layers, tiers, components, and how they communicate to form a complete system.

## Slide 7



The slide features a blue logo for Wyzask, Inc. in the top left corner, consisting of a stylized blue wave above the text "Wyzask, Inc.". To the right of the logo is the title "Point of View" in a blue serif font. A horizontal blue line separates the header from the main content area. The main content area contains a bulleted list with a blue arrowhead pointing to the first item. The footer area is separated from the main content by another horizontal blue line and contains two lines of small black text: "Copyright © Wyzask, Inc. 2004 Version 5.0" on the left and "Design Patterns - 7" on the right.

Wyzask, Inc. Point of View

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

Copyright © Wyzask, Inc. 2004  
Version 5.0

Design Patterns - 7

In addition to the subsystems, you have to also consider different points of view of the system. For example, as a person using a building, you have a different point of view than the person who is constructing the building. The plumber has a different point of view than the electrician. The general contractor has a different point of view than the construction workers. Each point of view is concerned with a different part of the architecture, or you could say that each point of view requires its own architecture.

We need to consider our software product from the point of view of the users, marketers, developers, maintainers, interactions with other systems, run time system, and hardware

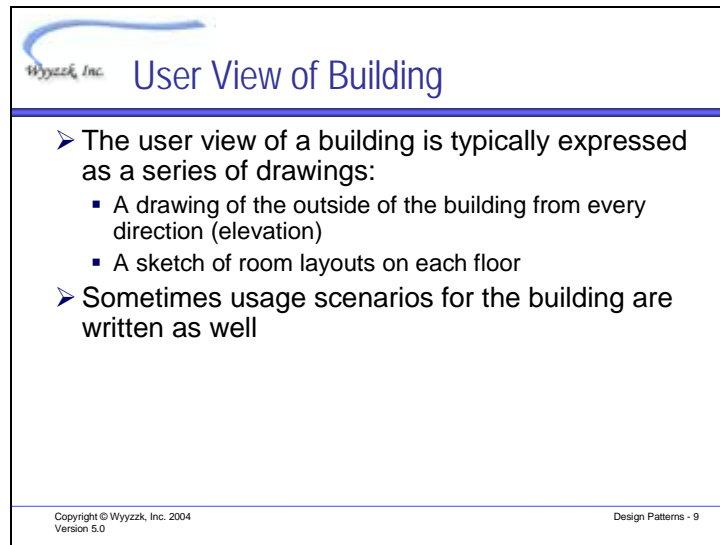
Slide 8

The slide features a blue header with the Wyyzzk, Inc. logo and the title 'Complexity'. Below the header, a blue arrow points to the text 'Complexity varies with:', followed by a bulleted list containing 'Time' and 'Hierarchy'. A diagram consists of three nested rectangles: an outer dark green rectangle labeled 'Enterprise Architecture', a middle light green rectangle labeled 'Line-of-business/Product Line Architecture', and an inner white rectangle labeled 'Project Architecture'. At the bottom, a blue footer contains the text 'Copyright © Wyyzzk, Inc. 2004 Version 5.0' on the left and 'Design Patterns - 8' on the right.

The amount of detail or complexity of an architecture changes in two ways:

Initial architectures are often not very detailed. Over time, as you learn more about the system you are developing, the architecture will become more detailed, and probably also more complex.

Architecture can be described in a hierarchical fashion, starting with a high level overview and continuing to more detailed descriptions of the individual subsystems. You can also consider architecture in the large – such as a whole city or all the systems in a corporate enterprise; architecture in the medium – such as a complex of buildings or a software line of business or product line; and architecture in the small – such as an individual building or the architecture of one software product. Notice that small may not be all that small! The architecture of a whole building is only small in relation to the architecture of a whole city.



The slide features a blue logo for 'Wyzzak, Inc.' in the top left corner. The title 'User View of Building' is centered at the top in a blue font. Below the title, a blue horizontal line separates it from the main content. The content consists of a bulleted list with blue arrowheads. The first bullet point is followed by two sub-bullets with square markers. The second main bullet point is followed by a single line of text. At the bottom of the slide, there is a thin blue horizontal line, and below it, the copyright information 'Copyright © Wyzzak, Inc. 2004 Version 5.0' is on the left and 'Design Patterns - 9' is on the right.

Wyzzak, 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

Copyright © Wyzzak, Inc. 2004  
Version 5.0 Design Patterns - 9

One way to look at architecture is by considering the different points of view and the information that is needed to develop a system from that point of view. Because it is the most obvious, we first consider the user view of the system.

It is common for an architect to ask questions about how a building will be used. The answers may or may not be written as formal usage scenarios, but the basic answers lead the architect to start narrowing the choices for the building. For example, if the building is to be used as a house, what kind of a home, what is important to the homeowners? Is the purpose of the home to raise children or is the home for middle-aged people whose children are grown and who like to entertain or is the home for an elderly couple who have a lot of books, like to sit and read in front of a fireplace, and frequently have grandchildren visiting. The answers to how the building will be used cause the architect to mentally select some set of appropriate patterns for the building as a starting point of the architecture.

## Slide 10




### 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

Copyright © Wyzzk, Inc. 2004  
Version 5.0 Design Patterns - 10

## Slide 11



### 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

Copyright © Wyzzk, Inc. 2004  
Version 5.0 Design Patterns - 11

You will unfortunately find people who claim that the outside appearance of the software (the user interface) is the architecture. They are only partially correct. The user interface or user view of the system is only one aspect of the architecture of the system.

In a relatively small system, if the project team has a lot of experience in the domain, the description of the user view may be enough to construct the system. For example, if you have some experience building things, you could build a simple one room dog house without any more information than what the outside looks like and the size of the dog. You can do this because you have the rest of the knowledge required in your head. With just a few notes and simple sketches, you would be able to do everything necessary to build the dog house based on your own experience of dog houses and construction. You would not build a large house or office building that way. It is too complicated.

Similarly with software. With some experience, you could put together a simple website with a blog (for example) just by knowing some basic information about the appearance of the website. You would not create mission control software for NASA just by knowing the user interface.

Even in these two examples, we actually knew more about the project than just the appearance. From experience we had some idea of what the system had to do and the major subsystems we had to consider. So we knew the user interface, the major subsystems, the communication between the subsystems, the main usage scenarios for the project, appropriate materials to use for the project (or hardware and software to use), and construction techniques. Also from experience we can estimate cost and time to completion.



Slide 12

Wyzzk, Inc. 4+1 Views

➤ To completely describe a software architecture, four views are needed:

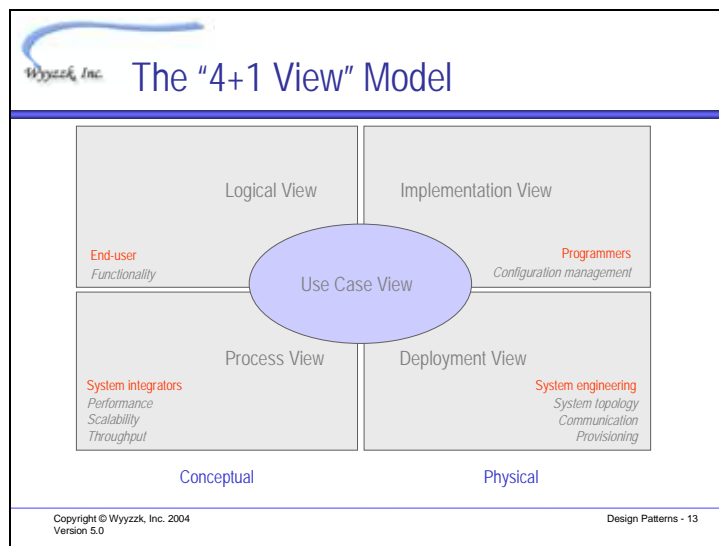
- The logical view to provide a static picture of the primary abstractions and their relationships
- The development view to show how the code is organized into subsystems and libraries and the use of commercial off-the-shelf (COTS) software
- The process view to show the processes and tasks
- The physical 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

Copyright © Wyzzk, Inc. 2004 Version 5.0 Design Patterns - 12

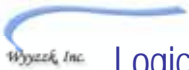
There are a variety of different ways to describe the views of an architecture. For this class, we have chosen to use the 4+1 architecture views developed by Philippe Krutchen.

Slide 13



Philippe Krutchen

## Slide 14

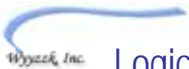


### Logical View of Building

- For the building, let's 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

Copyright © Wyzak, Inc. 2004  
Version 5.0 Design Patterns - 14

## Slide 15

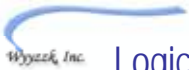


### Logical View of Building (cont.)

- 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

Copyright © Wyzak, Inc. 2004  
Version 5.0 Design Patterns - 15

## Slide 16

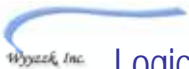


### Logical View of Software

- 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

Copyright © Wyzzk, Inc. 2004  
Version 5.0 Design Patterns - 16

## Slide 17




### 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

Copyright © Wyzzk, Inc. 2004  
Version 5.0 Design Patterns - 17

## Slide 18




### 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

Copyright © Wyzzk, Inc. 2004  
Version 5.0

Design Patterns - 18

## Slide 19




### 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

Copyright © Wyzzk, Inc. 2004  
Version 5.0

Design Patterns - 19

## Slide 20




### 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

Copyright © Wyzzk, Inc. 2004  
Version 5.0

Design Patterns - 20

## Slide 21




### 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

Copyright © Wyzzk, Inc. 2004  
Version 5.0

Design Patterns - 21

Slide 22




## 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

Copyright © Wyzzk, Inc. 2004  
Version 5.0 Design Patterns - 22

Slide 23




## 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.

Copyright © Wyzzk, Inc. 2004  
Version 5.0 Design Patterns - 23

## Slide 24




### 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

Copyright © Wyzzk, Inc. 2004  
Version 5.0 Design Patterns - 24

## Slide 25




### 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

Copyright © Wyzzk, Inc. 2004  
Version 5.0 Design Patterns - 25

## Slide 26




### 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

Copyright © Wyzak, Inc. 2004  
Version 5.0 Design Patterns - 26

## Slide 27




### 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

Copyright © Wyzak, Inc. 2004  
Version 5.0 Design Patterns - 27



Slide 28




## 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

Copyright © Wyzzk, Inc. 2004  
Version 5.0 Design Patterns - 28

Slide 29




## Examples

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

Copyright © Wyzzk, Inc. 2004  
Version 5.0 Design Patterns - 29

Slide 30




## Examples

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

Copyright © Wyzak, Inc. 2004  
Version 5.0 Design Patterns - 30

Slide 31




## 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)

Copyright © Wyzak, Inc. 2004  
Version 5.0 Design Patterns - 31

## Slide 32




### 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

Copyright © Wyzak, Inc. 2004  
Version 5.0 Design Patterns - 32

## Slide 33



### 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

Copyright © Wyzak, Inc. 2004  
Version 5.0 Design Patterns - 33

Some people consider this to be a part of the architecture, some consider it to be a part of the design. This macro design is closely related to the structures of the architecture, but is somewhat different from the architecture. Different designs of persistence (for example) could be applied to the same architecture.

Slide 34

Wyzzk, Inc. **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

Copyright © Wyzzk, Inc. 2004  
Version 5.0 Design Patterns - 34

Slide 35

Wyzzk, Inc. **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

**OOA**  
*Develop model of requirements*  
User's Perspective

➔

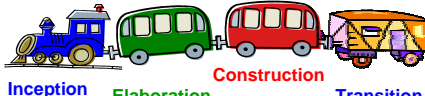
**OOD**  
*Add detail and design decisions*  
Developer's Perspective

Copyright © Wyzzk, Inc. 2004  
Version 5.0 Design Patterns - 35

Slide 36

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.



Inception Elaboration Construction Transition

Copyright © Wyyzzk, Inc. 2004  
Version 5.0 Design Patterns - 36


Slide 37

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

Copyright © Wyyzzk, Inc. 2004  
Version 5.0 Design Patterns - 37

## Slide 38




### 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

Copyright © Wyzzk, Inc. 2004  
Version 5.0 Design Patterns - 38

## Slide 39



### 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.

Copyright © Wyzzk, Inc. 2004  
Version 5.0 Design Patterns - 39