Developer to Architect

The Software Architect Role in the Enterprise

Chris Simmons
www.avidsoftware.com
chris.simmons@avidsoftware.com

pluralsight
hardcore developer training

Designing the Solution

Chris Simmons
www.avidsoftware.com
chris.simmons@avidsoftware.com

pluralsight
hardcore developer training

Designing the Solution

- **Design phase**
  - Design
  - Document
- **Two halves**
  - Design
  - Communicating the design
- **Software architecture**
  - What are software architecture & detailed design?
  - Goals
  - How are they different?
  - Who should perform them?
  - The role of architecture in an agile world

Designing the Solution

- **Approach to Architectural Design**
  - Two fundamental approaches
    - Top-down
    - Bottom-up
    - Which is the right choice?
- **Architectural Design Process**
  - Design Considerations
  - Design process step by step
  - Prototypes
  - Architectural Patterns
  - When to use them

Designing the Solution

- **Communicating the Solution**
  - 3 Main Objectives
  - Documentation standards
- **What are views?**
  - 4+1 architectural view model
  - Views and Beyond
- **Views & Beyond in practice**
  - Examples
    - Module View
    - Component-and-Connector View
    - Allocation Views

What is Software Architecture?

"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 in Practice (3rd Edition) –

Software Architecture

- **What is Software Architecture?**
  - A set of structures that communicate the goals of the solution

What is Software Architecture?

"A structure is simply a set of elements held together by a relation"

-- Software Engineering in Practice (3rd Edition) –

Software Architecture

- **What is Software Architecture?**
  - A set of structures that communicate the goals of the solution
  - Singular task: to identify the structures, their elements, and the relationships between them
  - Properties and behaviors must be identified
  - Architecture must define publicly visible properties that identify how elements relate and interact with each other
  - Architecture must have 4 attributes
    - Elements
    - Relations
    - Properties
    - Behavior

What is Software Architecture?

"A structure is architectural if it supports reasoning about the system and the system's properties."

-- Software Engineering in Practice (3rd Edition) –

Software Architecture

- **What is Software Architecture?**
  - A set of structures that communicate the goals of the solution
  - Singular task: to identify the structures, their elements, and the relationships between them
  - Properties and behaviors must be identified
  - Architecture must define publicly visible properties that identify how elements relate and interact with each other
  - Architecture must have 4 attributes
    - Elements
    - Relations
    - Properties
    - Behavior
  - The architecture need only contain the portions of the system that are architecturally significant
  - Design only enough for the team to begin coding or to perform detailed design

Software Architecture

- **What is Software Architecture?**
  - The architecture not only defines the structures of the solution but provides structure to your project
    - Guiding the project through all phases
    - Providing constraints for the team to work within
    - Directing toward successful completion
  - Architects Primary Objective
    - Provide architecturally significant structures which are made up of: elements, relations, properties and behaviors
    - Create a blueprint that guides and shapes the implementation with regard to the pieces that are the most difficult and costly to change

Software Architecture

- **What is detailed design?**
  - No clear delineation as to where architectural design ends and detailed design begins
  - Lines are not only blurred they move depending on many factors
  - Architecturally significant design choices
    - System wide impact
    - Meet the non-functional requirements
    - Guide downstream design activities
  - Design choices that focus on implementation are non-architectural
  - Design is the process of serially decomposing the whole into its many constituent parts
  - Architectural design is performed until it reaches the point where the development team can begin their work
    - Development team can design individual modules, classes or even begin coding
    - Granularity is dependent not on the architect, or even the project, but on the development team

Software Architecture

- **What is detailed design?**
  - It is not the job of the architect to design every aspect of the solution
  - Development team often has more experience in certain technical domains
  - It is the architects job to:
    - Understand the problem area and identify where the solution should fit within the organization
    - Understand what technologies are best suited to the project
    - Provide structure to the development team with an eye toward meeting the functional, non-functional and business goals of the project
    - Decide when enough is enough – the rest is non-architectural/detailed design

Software Architecture

- **What is detailed design?**
  - Design is evolutionary and iterative
  - Architectural design focuses on the pieces of the solution that are difficult and costly to change
  - Design continues long after architectural design is complete
  - Design is only completed when the project is delivered
  - Development team will continue to make implementation decisions throughout the project life cycle
  - Guided by the architectural design that preceded it

Software Architecture

- **How are they different?**
  - Design process is the same
  - Context or scope is the differentiator
  - When creating your architectural design it should contain :
    - Structures and elements, that once defined, should rarely change
    - When they do it is extremely painful
  - High-level architecture identifies all the top level elements and how they interact
  - Detailed design begins with the high-level architecture and continues to fill in the implementation details within each of these elements
  - Design process continues throughout the life of the project and only ends with the project's completion

Software Architecture

- **So who does the detailed system design?**
  - It depends…
  - Solutions architect is responsible for creating the overall architectural design
  - Application architect is responsible for the more detailed design and implementation choices
  - You may be both the solution and the application architect
  - When separate roles, the delineation point will be more explicit
  - Where you are both the solution architect and the application architect, it will get blurred
  - Distinction is important because each task addresses separate concerns of the project

Software Architecture

- **So who does the detailed system design?**
  - Must force yourself to wear two hats and approach each task as discrete units of work, each with different goals
  - Application architect or lead developer performs detailed design
  - Sometimes that is you

Software Architecture

- **The role of architecture in an agile world**
  - Most, if not all, of the concepts I have outlined still apply
  - On agile teams the architect is most likely a member of the development team
  - Architect's qualities are usually present in someone on your team, regardless of title
  - When forming teams only a few individuals emerge as leaders
    - Architect asserting his or her role on the project
    - Architect is a role not necessarily a title
  - Each phase that I outlined happens on every project no matter what methodology is used
  - Iterative and evolving architecture IS an agile approach
  - Architects must design solutions that insulate design choices from change
  - Approaching design as iterative and evolving helps us to react when we need to

Approach to Architectural Design

- **Large systems are extremely complex and difficult to comprehend in their entirety**
- **Separate large problem domains into manageable pieces**
- **Abstract and encapsulate the complexities of the problem at hand**
- **Many approaches to design**
- **Two fundamental approaches to design and their benefits**
  - Top-down
  - Bottom-up
- **Referred to by many names**
  - Up-front vs evolutionary design
  - Decomposition vs composition
  - Planned design vs emergent design

Approach to Architectural Design

- **Top-down**
  - Highest level of abstraction, progressively work downward
- **Bottom-up**
  - Focus on the components that makeup the solution, working upward
- **All have common objectives**
  - Create a set of common terms that can be used to facilitate communications
  - Provide a representation of functional components
  - Create a model that facilitates the partitioning of the problem domain

Approach to Architectural Design

- **Top-down**
  - Traditional approach
  - Break down the system into a series of components
  - Begins at the highest level of detail
  - Performed iteratively
  - Series of sequential decomposition exercises
  - Series of black boxes, interfaces and relationships
  - Basis for implementation choices
  - Common in the enterprise
  - Most effective when the problem domain is well understood
  - Architect focuses on the larger issues up front

Approach to Architectural Design

- **Top-down benefits**
  - Effective on both large and small projects
  - Provides a logical and systematic approach
  - Lends itself to system partitioning
  - Helps to reduces size, scope and complexity of each module
  - Works for both functional and object oriented design

Approach to Architectural Design

- **Top-down drawbacks**
  - Requires an in depth understanding of problem domain
  - Partitioning doesn't facilitate reuse
  - Sometimes leads to ivory tower architecture
  - Design flaws can sometimes ripple up to the highest layers

Approach to Architectural Design

- **Bottom-up**
  - Process of defining the system in small parts
  - Like assembling Legos
  - Where concept of emergent architecture is founded
  - Typically encountered when using an agile development methodology
  - More common than top-down?
  - Is there an architecture when you chose bottom-up?
  - As the project progresses the architecture really does emerge…eventually

Approach to Architectural Design

- **Bottom-up advantages**
  - Allows a team to begin coding and testing early
  - Simplicity
  - Promotes code reuse
  - Promotes the use of continuous integration and unit testing

Approach to Architectural Design

- **Bottom-up disadvantages**
  - Can become difficult to maintain
  - Benefits of code reuse are eliminated or at least delayed as the team grows
  - Design flaws can ripple throughout the entire solution

Approach to Architectural Design

- **What is the right choice?**
  - They both are
  - It is up to you as the architect to decide which approach is best for your particular project, team and organization
  - Guidance:
    - Top-down will serve you well in the enterprise
    - Much of your management will be familiar with top-down
    - Top-down is useful when high-level estimates are required
    - Top-down is often a better fit with big projects and large teams
    - Bottom-up is extremely effective on small projects with just a handful of developers

Approach to Architectural Design

- **Personal recommendation and insights**
  - I like using both approaches
  - Benefits of each approach can be realized when using the right mix
  - Great number of benefits come with embracing a top-down at inception
  - Typically will spend some time creating a high-level architecture
  - Provides just enough structure
  - Can be used by the development team to create more detailed designs or to help organize sprints
  - Hybrid approach introduces some structure to the project at the inception but allows for evolutionary design to occur

Architectural Design Process

- One technique to design your solution
- Top-down functional decomposition
- End result will be a high-level architecture
- High-level architecture used to create more detailed design
- High-level architecture could be used by an agile team practicing emergent architecture
- Introduction to design process

Architectural Design Process

- **Architectural design shares many things in common with detailed design**
- **Scope**
  - Most difficult and costly to change
  - Architecturally significant elements
- **Structures, behaviors and the relationship**
- **Iterative**
- **Layered**
- **Begins by defining the highest level of detail**
- **Make sure everyone is speaking the same language**
- **Define a common set of terms**
  - Keep them simple and be consistent

Architectural Design Process

- **System**
  - Highest level of detail
  - Defines the entire scope of the project
  - Made up of one or many subsystems
- **Subsystem**
  - Logical separations of responsibility
  - Groupings of related elements
  - Sometimes a standalone application

Architectural Design Process

- **Module**
  - Logical separations of responsibility
  - High-level groupings of other modules or components
  - Responsibilities are easily understood by the entire team
- **Components**
  - Execution units in our solution
  - Designed to be pluggable
  - Defined by their interfaces and behaviors

Architectural Design Process

- Each terms defines an element and a hierarchy
- Systems will include one or many subsystems
- Subsystems will include one or many modules
- Use of these exact terms is not important
- Defining YOUR terms is what's important
- Don't get lost in the details
- Use a horizontal approach

Architectural Design Process

- **Design Considerations**
  - What are the goals & objectives of your architecture?
  - What style of application will best meet your needs?
  - What are the architectural significant requirements that should be considered?
  - How does this design account for them?
  - What are the important system, run-time, design and user quality attributes of the solution?
  - How are these addresses in the architecture?
  - Do any of these attributes have competing goals?
  - Which attributes are more important and what are the tradeoffs that should be considered?

Architectural Design Process

- **Design Considerations**
  - What are the technical concerns of the solution?
  - Have you accounted for cross cutting concerns?
  - Will you utilize existing frameworks or create some as part of your project?
  - What patterns should be considered?
  - Have you considered technology, team, deployment and time constraints?
  - Are there alternative architectures that should be considered?
- **All of these are important considerations that will impact and guide your design choices**

Architectural Design Process

- **Step 1: Start with the big picture & know your boundaries**
  - Where does your solution fit into your organization's ecosystem?
  - Will this solution interoperate with other systems within your organization?
  - Will this solution be a standalone application?
  - Will other applications use the database you are creating?
  - Will your application provide or consume any services?
  - Will this application access database that are not part of your solution?
  - Determine the solution's responsibilities
  - Determine how it will function as a part of your organizations ecosystem
  - Take note of all systems that your solution will interact with and how they will communicate
  - Boundaries create context

Architectural Design Process

- **Step 2: Define your subsystems**
  - Identify subsystems that comprise the solution
  - Subsystems are logical separations of responsibility within the system boundary
  - Subsystems are groupings of related elements that comprise the entirety of your solution
  - Subsystems may be partitioned by application type
  - If subsystems interact with each other, how and what they communicate, must be considered
  - Inventory of all subsystems, an understanding of their responsibilities and how they interact with each other

Architectural Design Process

- **Step 3: Define your modules**
  - Identify all modules that make up each subsystem
  - Subsystems contain one or many modules
  - Modules are dedicated to a single logical area of responsibility
  - Modules are high-level groupings of other modules
  - Modules will be defined for every subsystem proceeding horizontally from one subsystem to the next
  - Interoperation between modules is important

Architectural Design Process

- **Solution decomposed into 3 levels**
  - System
  - Subsystems
  - Modules
- **High-level architecture or HLA**
  - System boundaries
  - External interoperation points
  - Subsystems
  - Subsystem types
  - Subsystem interoperation points
  - Modules
  - Validates, constrains and provides structure
  - Used as the basis for team organization
  - Serves as an architectural constraint for more detailed design

Architectural Design Process

- **Step 4: Define your components**
  - Define the components that make up or map to each module
  - Independently deployable software elements
  - Designed to be pluggable
  - Defined by their interfaces and behaviors
  - Responsible for a single function
  - Provide replaceable, independent and encapsulated elements
  - Should be loosely coupled
  - Provide much flexibility in deployment
  - Lower levels of design should be performed by development team
  - Level of decomposition required is up to the architect

Architectural Design Process

- **When to use prototypes**
  - Prototypes help to reduce risk by proving or validate design concepts
  - Prototyping is often a necessary tool for the more risky and lesser understood parts of the solution
  - Prototypes are often created by the architect
  - May be assigned to individuals on the development team if desired
  - Prototypes prove concepts to be used as examples during construction
  - Providing time constraints will help to keep the task focused

Architectural Design Process

- **Architectural patterns**
  - Architectural patterns help to solve recurring and well understood architectural problems
  - Architectural patterns deal with the structure and organization of the entire system or subsystem
  - Design patterns solve well understood problems encountered during construction
- **Benefits of architectural patterns**
  - Solve recurring problems
  - Provide a common language or short hand
  - Promote high level discussions
  - Simplify the design process

Architectural Design Process

- **Architectural patterns**
  - May be more than one architectural pattern utilized in your solution
  - Should be used as a starting point
  - Don't be afraid to modify to solve your particular problem
  - Patterns are descriptions of solutions, not implementations
  - Don't get too hung up on applying patterns
  - Intended to provide guidance when encountering common problems
  - Patterns must be adapted to address YOUR particular problem domain

Architectural Design Process

- **What is the right solution?**
  - No single right solution to every problem
  - Almost infinite number of ways to solve any particular problem
  - Don’t get too hung up on making sure your ideas are always the best
  - Primary goal: find the best solution
  - Best meets the functional, non-functional and business goals
  - Sometimes we need to check our egos at the door

Communicating the Solution

- Communicating the design to your stakeholders
- Design and documentation are two distinct processes
- Design identifies all the structures that make up the solution
- Documentation communicates design structures
- Artifacts are representations of the architecture
- Created to communicate information to a particular audience
- No right or wrong number of artifacts
- Size and makeup of your team determines the number and detail
- On small teams a high level architecture document may be sufficient
- When the team grows so do the benefits of documentation

Communicating the Solution

- **The more complex the design, the more the need to create an artifact**
- **Don't forget about your non-technical stakeholders**
- **Small teams balloon quickly**
- **Time spent documenting pales in comparison to the time spent**
  - Communicating the design
  - Educating new team members
  - Justifying and explaining design decisions
- **Artifacts can do much of this for you**
- **Artifacts help you scale**
- **Blueprint upon which to construct the solution**
- **Why decisions were made**
- **What should be considered in the future**

Communicating the Solution

- Value provided is much greater than cost
- You will never be able to scale as well as your artifacts

Communicating the Solution

- **Three main objectives for creating architectural artifacts**
- **Objective 1: Facilitate communication between stakeholders**
  - Communicate what is being built
  - How functional and quality related requirements are being achieved
  - Communicate system constraints
  - Blueprint serves as the interface
  - Define the structures
  - Define the terms upon which the team will communicate
  - Communicate meaningfully within and between teams

Communicating the Solution

- **Objective 2: Basis for detailed design and construction efforts**
  - Provide a foundation
  - Provide guidance, structure and constraints

Communicating the Solution

- **Objective 3: Educate the team, both business and technical**
  - Educate both technical and non-technical stakeholders
  - Facilitate discussions
  - Provide background
  - Explain choices
  - Document decisions
  - Provide context and abstraction
  - Provide a basis for educating

Communicating the Solution

- **How do we know we are meeting these objectives?**
  - Ask yourself two questions
  - Does this document provide value?
  - Does this level of detail communicate enough?
  - Second question tells us when to stop
  - Is there enough detail for our business users to understand how we are meeting their needs?
  - Is there enough detail for our development team to build a solution?
  - When both of these questions are answered then you have provided enough detail

Communicating the Solution

- **Documentation Standards**
  - Focus on diagrams
  - Diagrams are more effective for communicating complex concepts
  - 3 Categories: Formal, Informal and Hybrid

Communicating the Solution

- **Formal**
  - Unified Modeling Language or UML
  - Industry standard language for modeling
  - 14 types of diagrams
  - Two categories
    - Structural
    - Behavioral
  - Structural diagrams define the structures that makeup the application
  - Behavioral diagrams represent the behavior and functionality
  - Interaction diagrams represent the control flow between components
  - Adoption is not widespread
  - Usage is inconsistent
  - Very good at low level documentation
  - Not as straightforward for higher levels

Communicating the Solution

- **Informal**
  - Box and line diagrams
  - Most common type of diagram
  - Barrier of entry is low
  - No constraints
  - Easily expressed
  - No rules to break
  - Adoption is easy and widespread
  - Ambiguous, imprecise and disorganized

Communicating the Solution

- **Hybrid**
  - Non-standard UML diagrams in conjunction with box and line drawings
  - Martin Fowler - 3 modes of UML diagrams
    - Sketch
    - Blueprints
    - Programming language
  - Sketch is most common
  - Sketch doesn't necessarily conform to all the rules of UML
  - Selectively communicating concepts and ideas
  - Non-standard usage of UML is hybrid approach
  - Natural progression from sketches to blueprints
  - Blueprints are the architectural diagrams that are the focus of this section

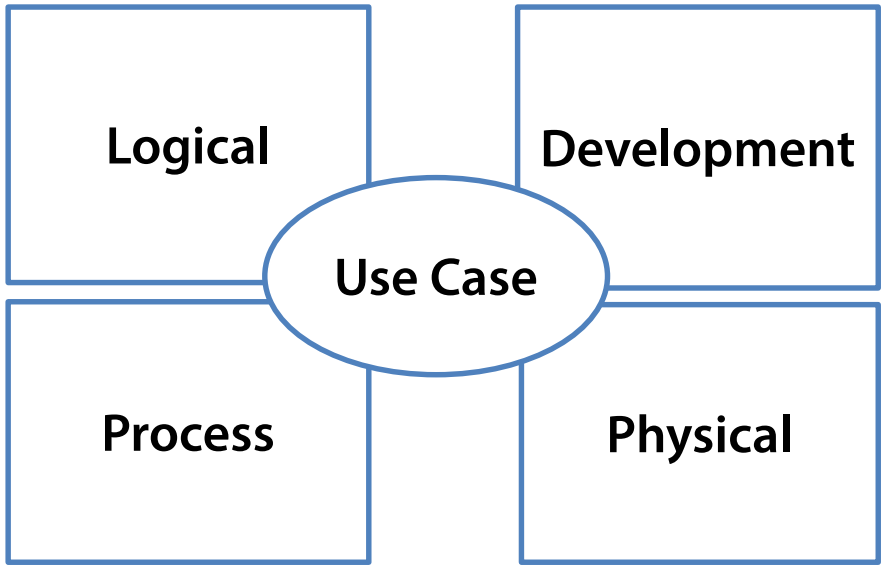
Communicating the Solution

- **Personal recommendations**
  - Use hybrid approach
  - Use UML selectively
  - Hybrid approach provides best of both worlds
  - Martin Fowlers “UML distilled” is a good starting point

What are views?

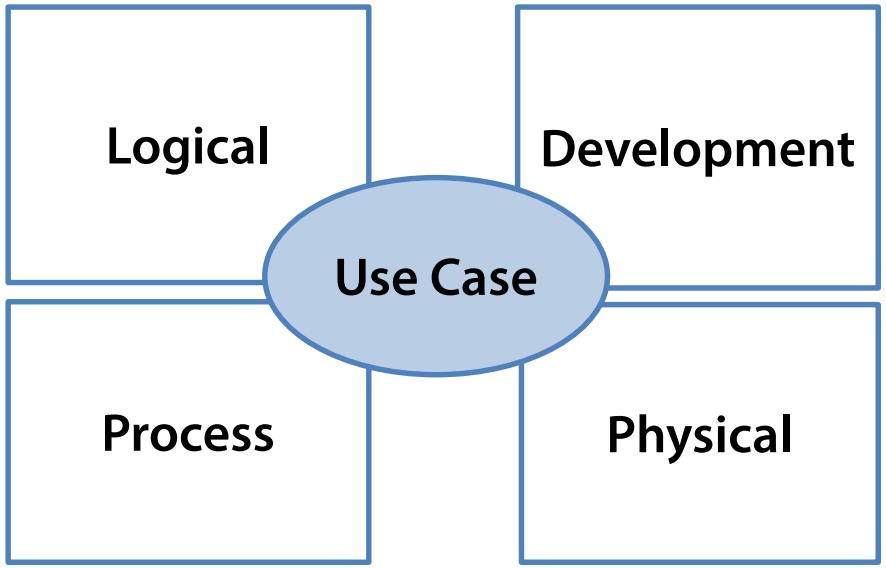
- A window into the architecture
- Single viewpoint targeted to a particular audience
- No single view
- No set number and types of views
- The architecture is comprised of all the views
- Several well-known approaches rely on of views

4+1 architectural view model



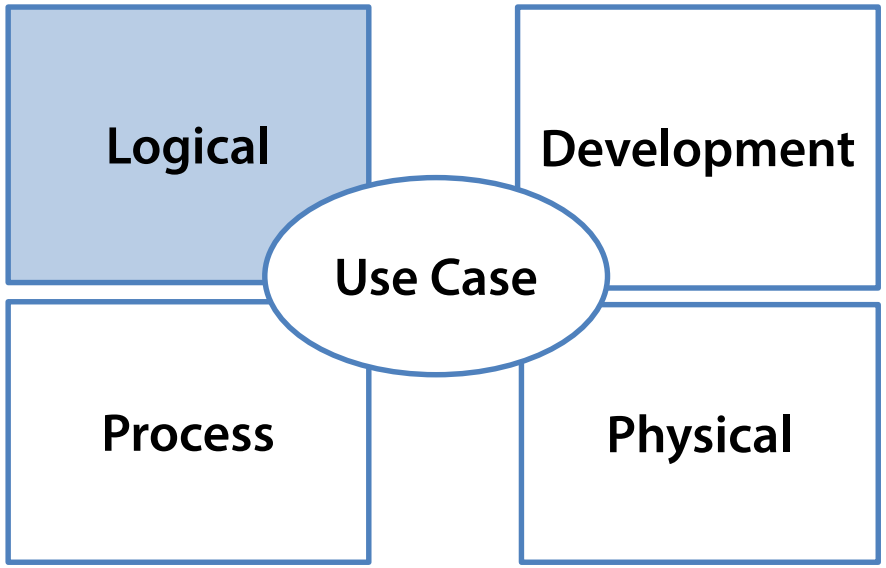
- 5 views
- Each address a separate set of concerns and audience

4+1 architectural view model



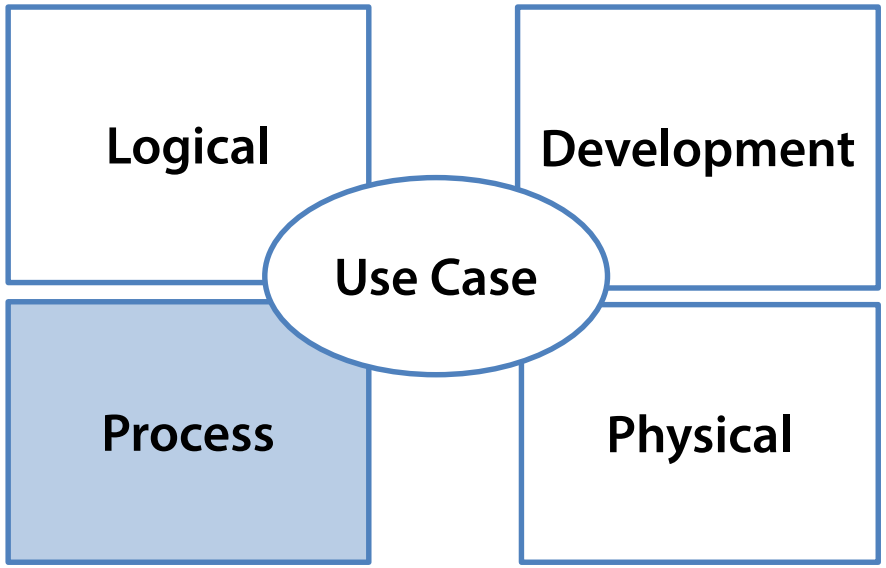
- Ties all of the other views together
- User requirements
- System functionality
- Internal and external actors
- Represented using use UML case diagrams

4+1 architectural view model



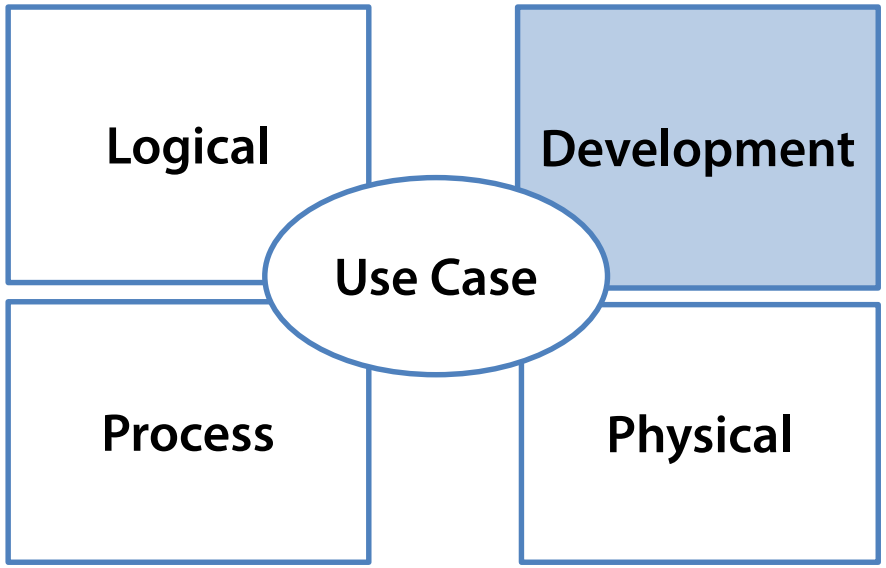
- End user functionality viewpoint
- Structures of the architecture that implement functional requirements
- Classes and their relationships
- Represented using UML class diagrams

4+1 architectural view model



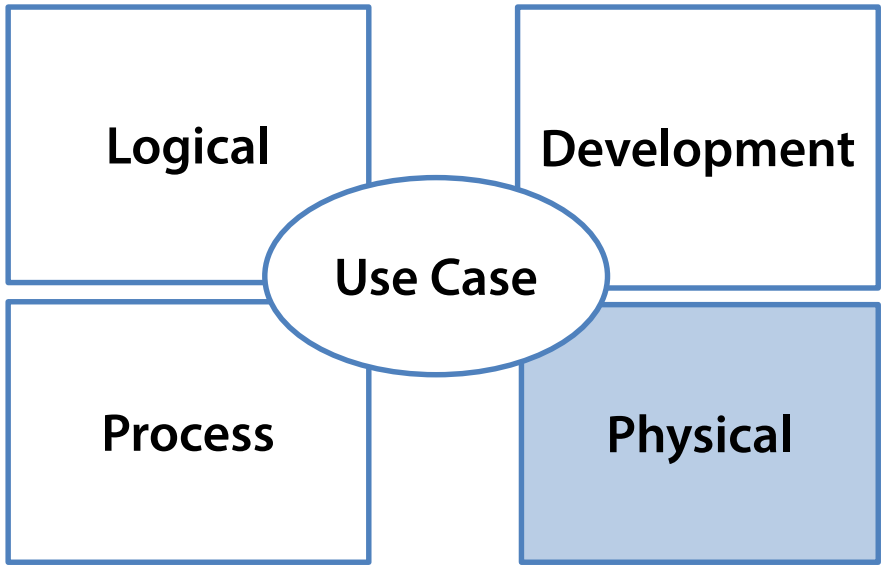
- Run-time viewpoint
- Performance
- Reliability
- Scalability
- Interaction and communication
- Represented using UML activity diagrams

4+1 architectural view model



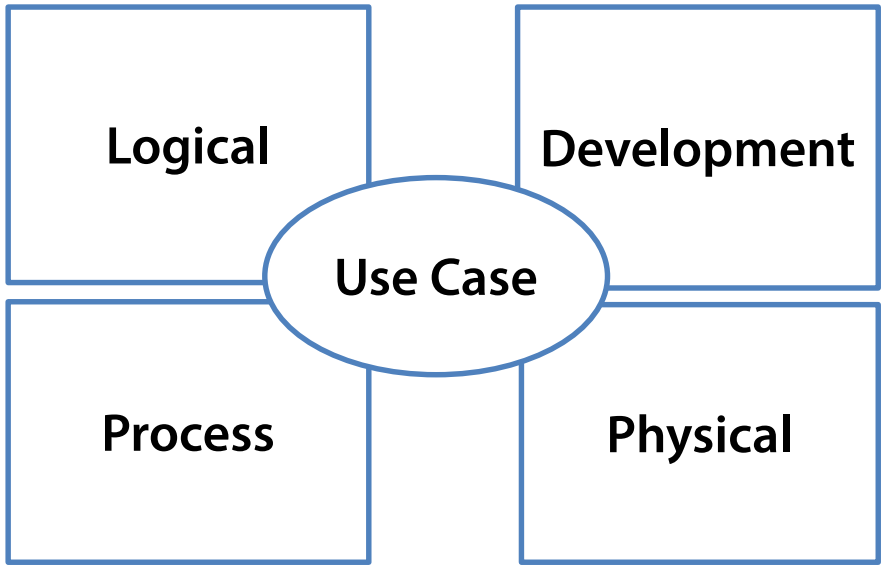
- Structure & organizational viewpoint
- Modules are organized
- Module interaction
- Represented with a UML package and component diagram

4+1 architectural view model



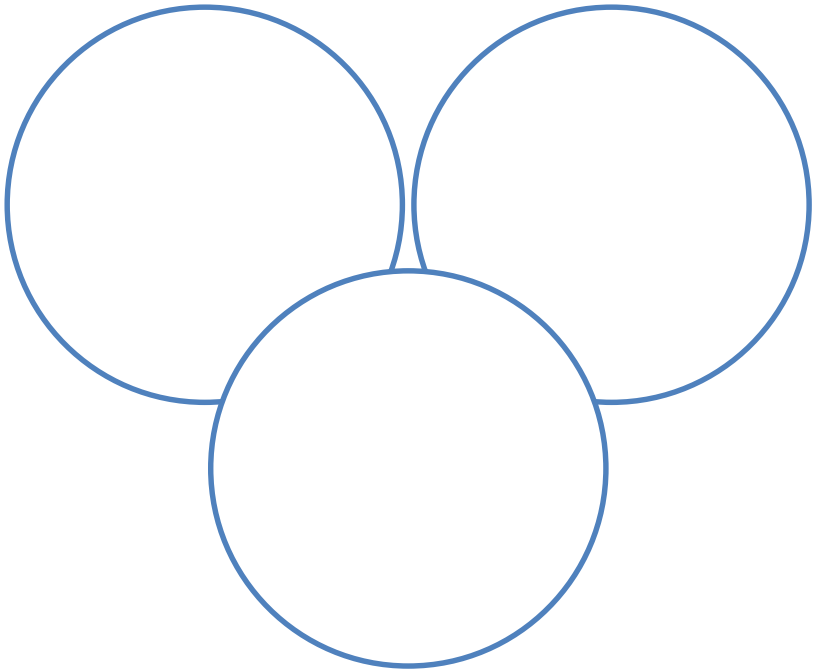
- Infrastructure viewpoint
- Deployment
- Communications between physical tiers
- Represented with a UML deployment diagram

4+1 architectural view model



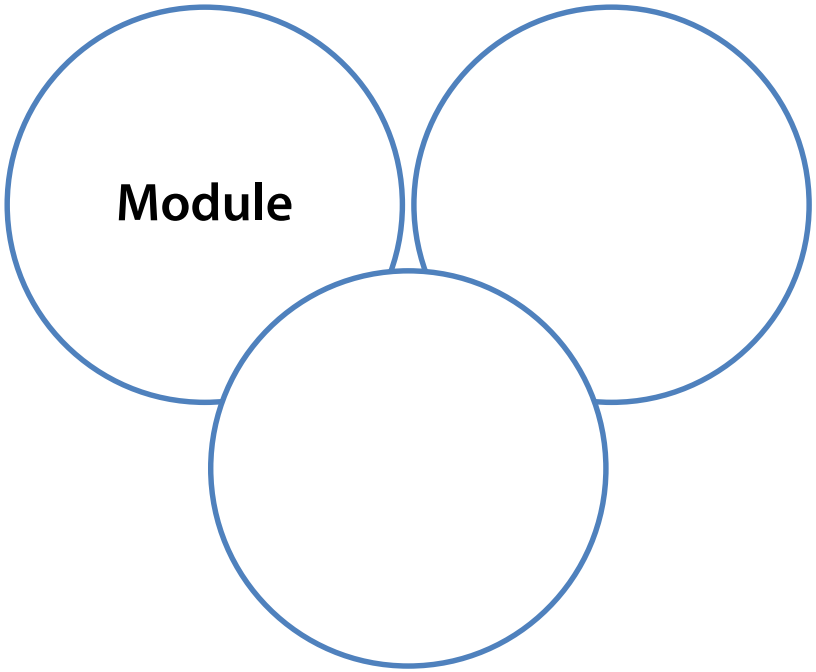
- Each view targets a specific set of stakeholders and concerns
- 5 view categories represent the whole of the architecture

Views & Beyond



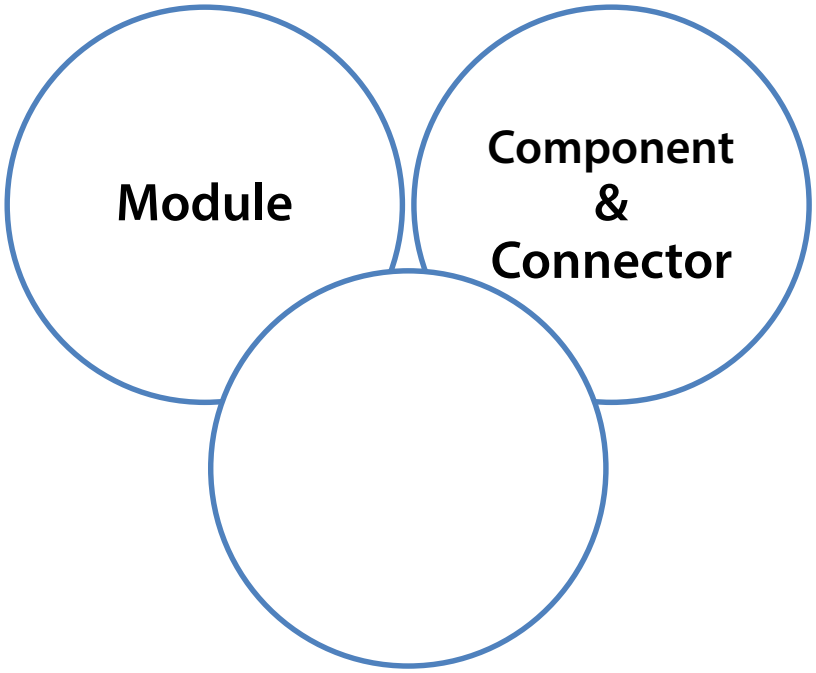
- Documenting Software Architectures: Views and Beyond (2nd Edition): Paul Clements, Felix Bachmann, Len Bass, David Garlan, James Ivers, Reed Little, Paulo Merson, Robert Nord, Judith Stafford
- 3 view categories
- Each represent a distinct set of styles

Views & Beyond



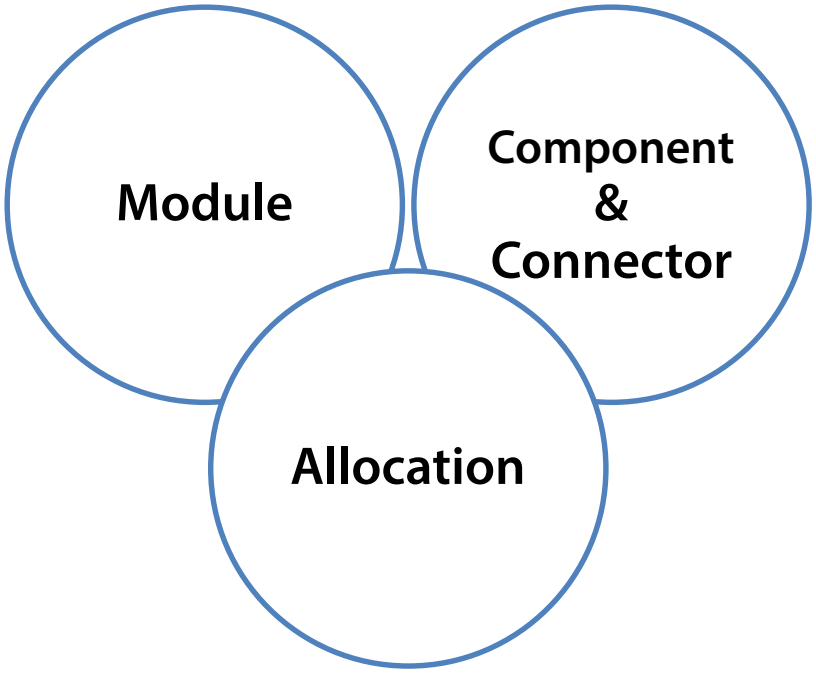
- Documenting Software Architectures: Views and Beyond (2nd Edition): Paul Clements, Felix Bachmann, Len Bass, David Garlan, James Ivers, Reed Little, Paulo Merson, Robert Nord, Judith Stafford
- 3 view categories
- Each represent a distinct set of styles

Views & Beyond



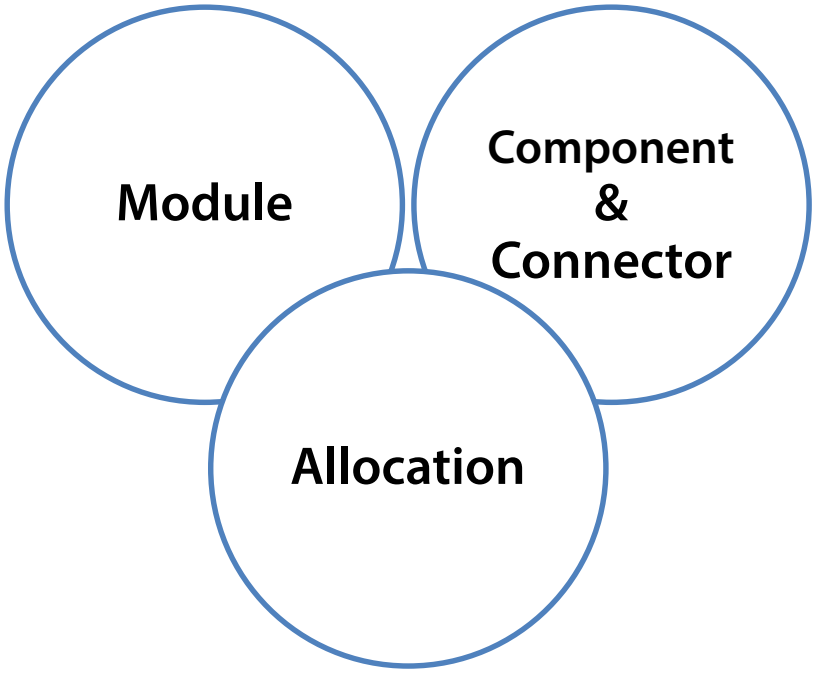
- Documenting Software Architectures: Views and Beyond (2nd Edition): Paul Clements, Felix Bachmann, Len Bass, David Garlan, James Ivers, Reed Little, Paulo Merson, Robert Nord, Judith Stafford
- 3 view categories
- Each represent a distinct set of styles

Views & Beyond



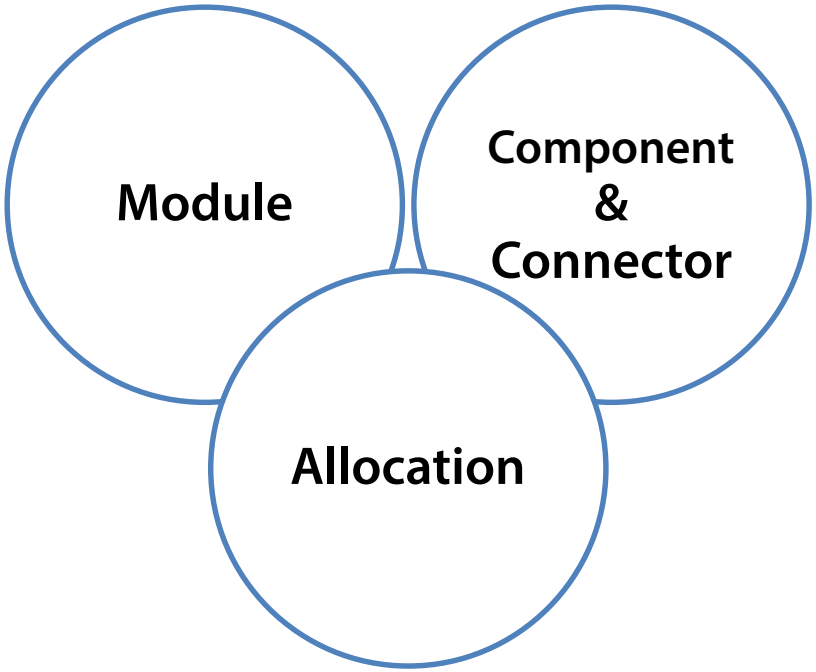
- Documenting Software Architectures: Views and Beyond (2nd Edition): Paul Clements, Felix Bachmann, Len Bass, David Garlan, James Ivers, Reed Little, Paulo Merson, Robert Nord, Judith Stafford
- 3 view categories
- Each represent a distinct set of styles

Views & Beyond



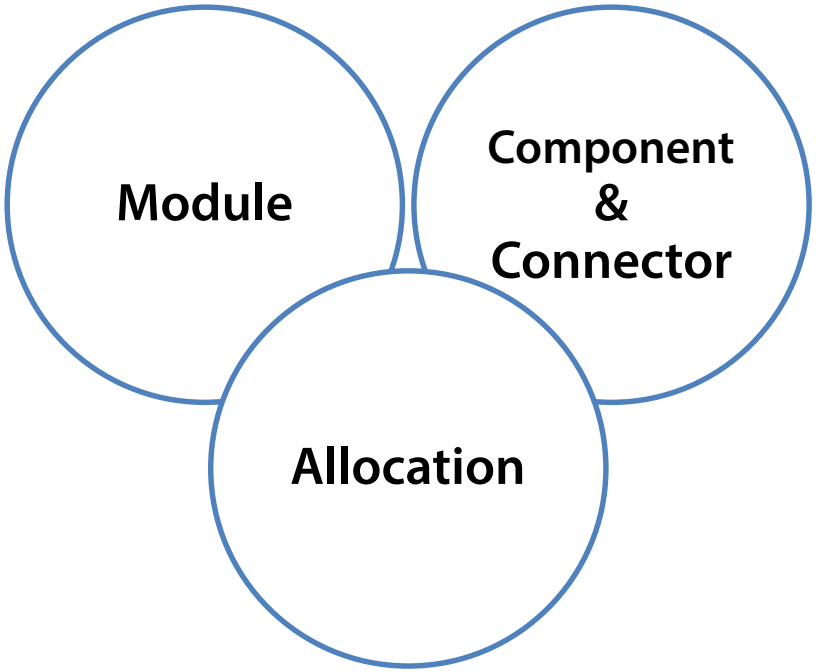
- Multiple views styles
- Enumerate, organize and describe common view styles
- Prescribe notation used for each style
- View styles are often combined

Views & Beyond



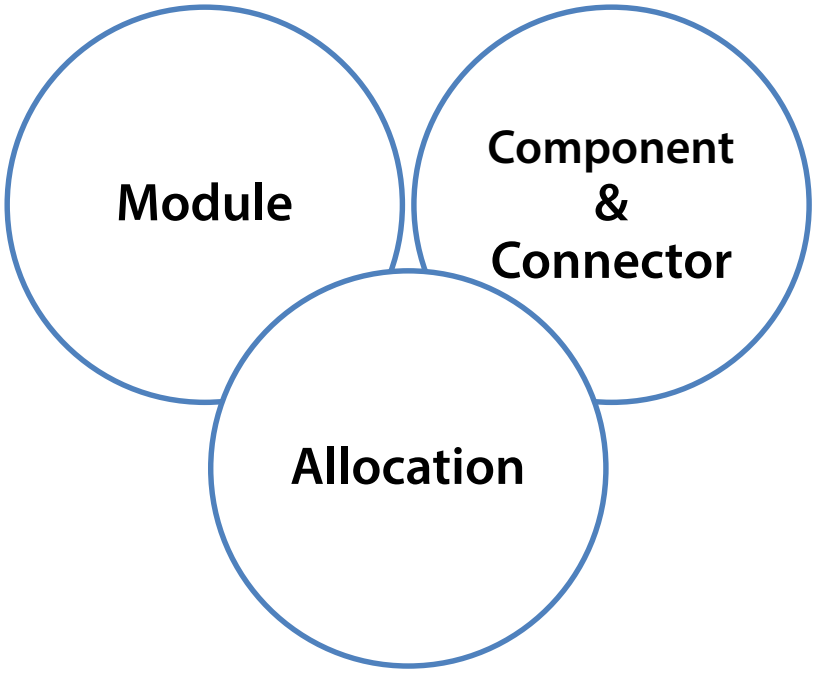
- Provide a style guide that describes notation
- Concise, understood and consistent
- UML only solves part of the problem
- Still a need to use informal and hybrid notation

Views & Beyond



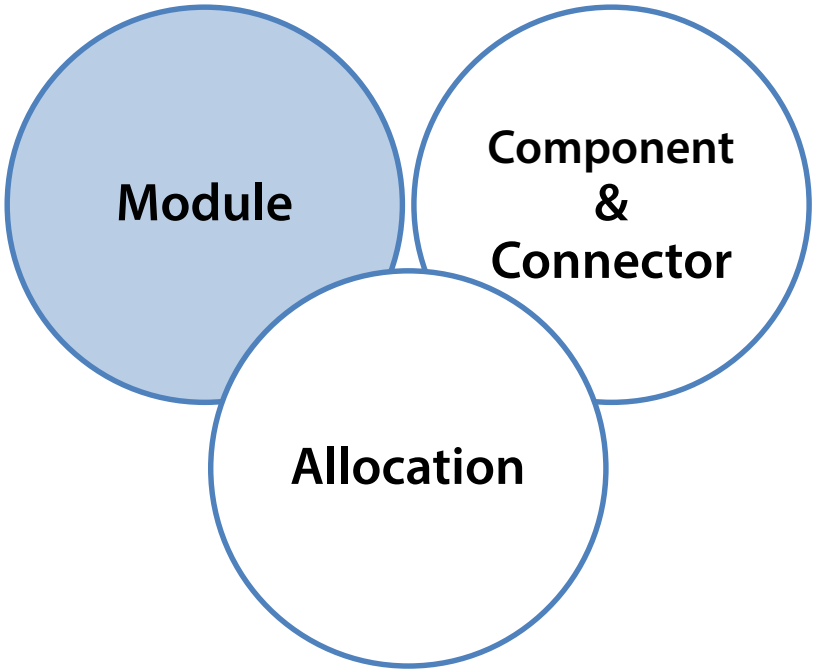
- No single industry standard approach to architectural documentation
- Offer their text as a guide
- Other valid approaches
- Documentation must be standardized

Views & Beyond



- 3 categories of views
- Each category contains many view styles
- Each categories does not prescribe number of views
- Standard sets of styles that should considered
- Style guide prescribes the notation style that will be utilized

Views & Beyond

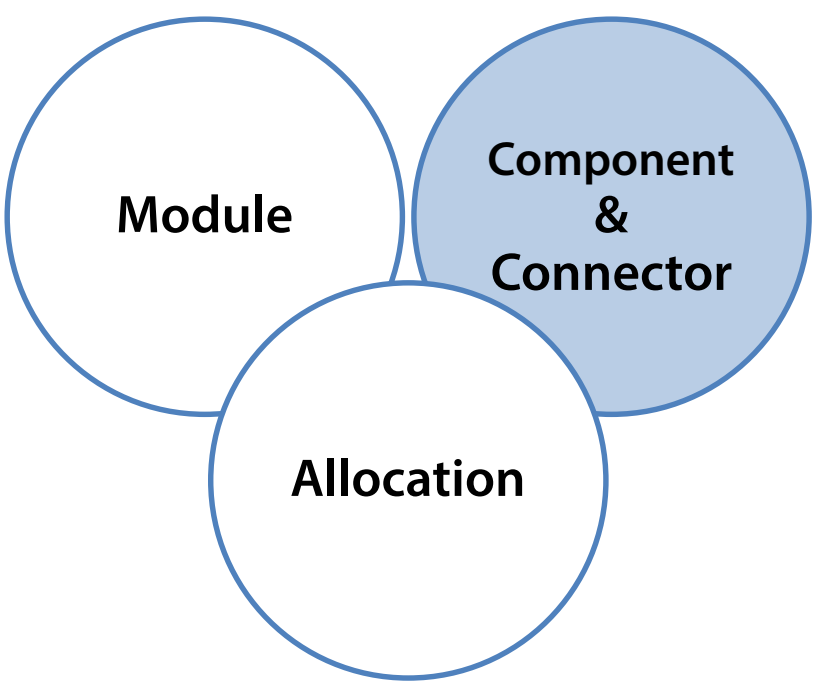


Styles

- Decomposition
- Uses
- Generalization
- Layered
- Aspects
- Data model

- Structural view of the architecture
- How modules are organized and how they interact
- Blueprint of the system
- Show dependencies
- Styles may be combined on any single view

Views & Beyond

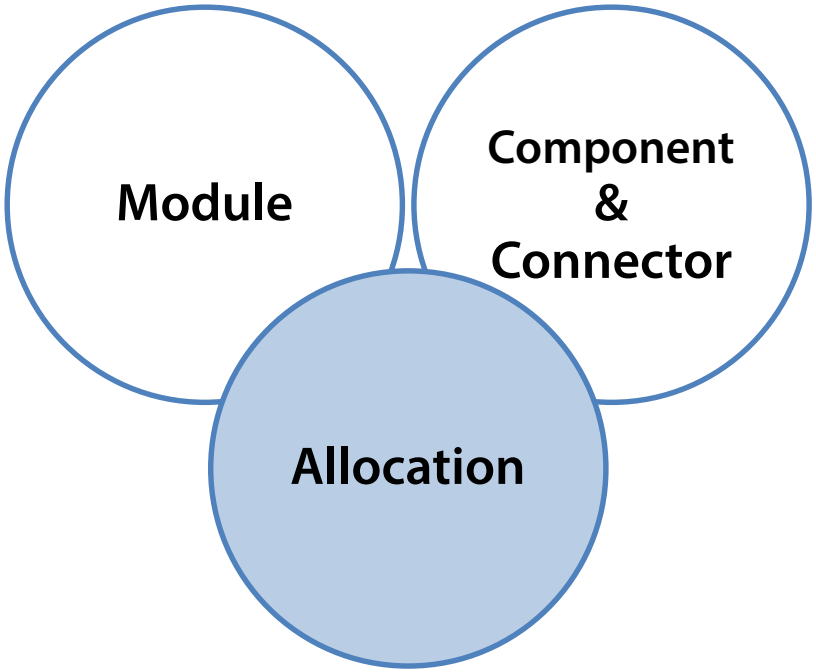


Styles

- Data flow
- Call-return
- Event-based
- Repository

- Behavioral view of the architecture
- How the elements of the system work together at run-time
- How they meet performance, reliability and availability quality attributes

Views & Beyond

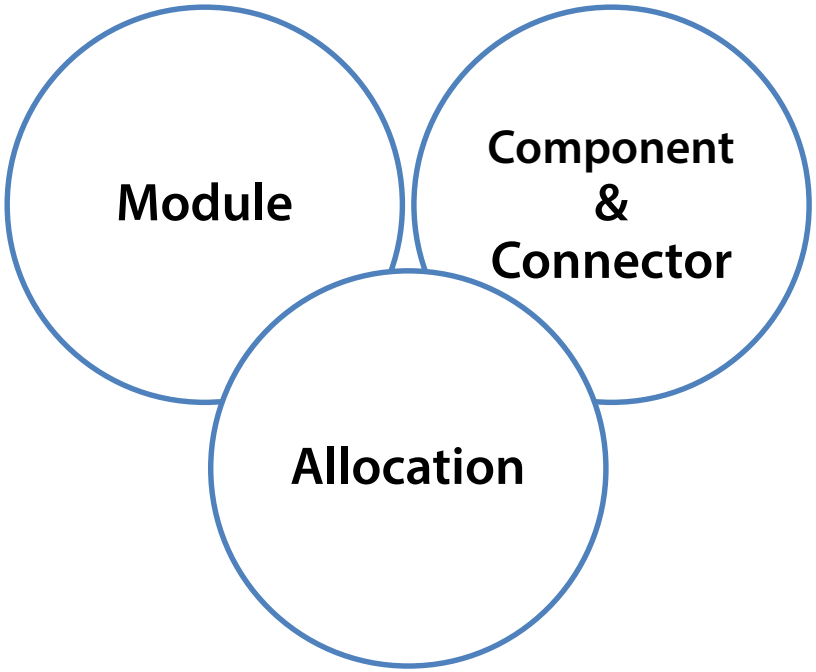


Styles

- Deployment
- Install
- Work assignment

- Development, deployment and execution views of the architecture
- Allocated to infrastructure and work teams
- Map elements to hardware and work teams

Views & Beyond

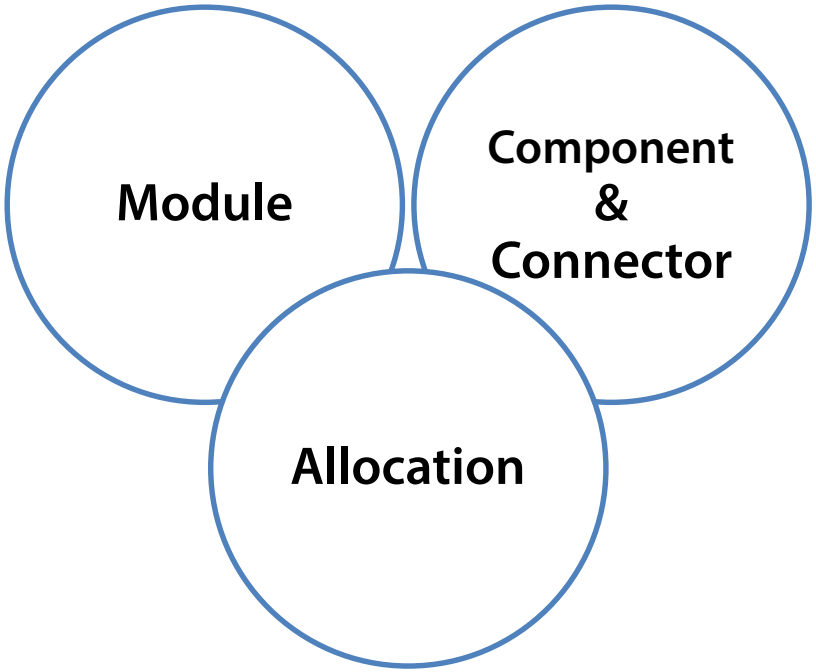


- Views are not just made up of the architectural diagrams
- Supporting information must be presented
- Element catalog
- Variability guide
- Rationale

What are views?

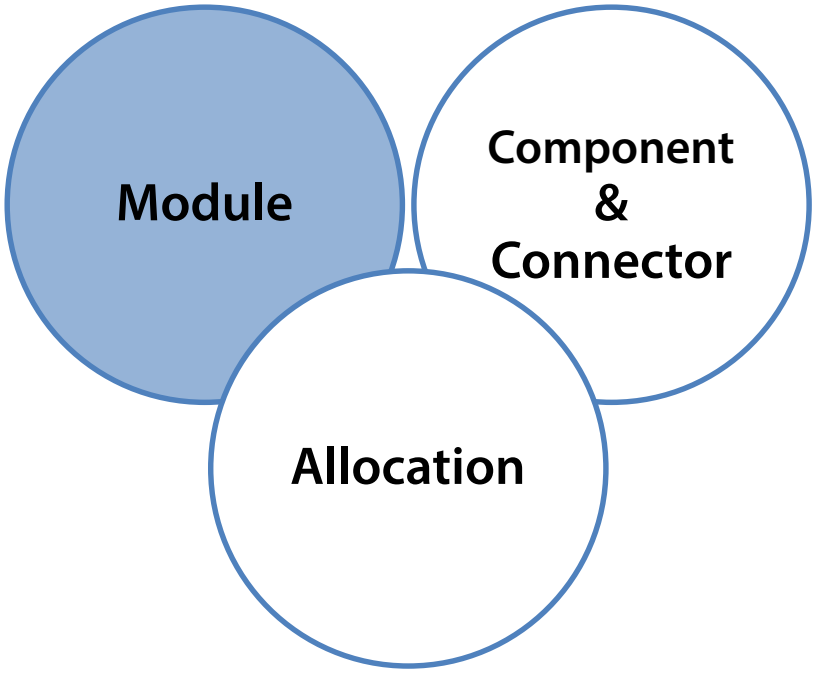
- Many approaches to architectural documentation
- Mainstream approaches have adopted a view based approach
- No single document that can represent complexity
- Abstraction
- Decomposition
- Different views of the architecture
- Address a certain set of concerns
- Target a specific audience
- Not advocating the wholesale adoption of either approach
- Continue your educating with either approach
- No one size fits all approach
- Take pieces from each that best suit you, your projects and your organization

Views & Beyond in practice



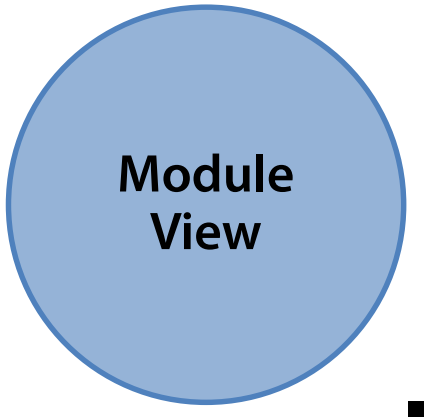
Examples

Views & Beyond in practice

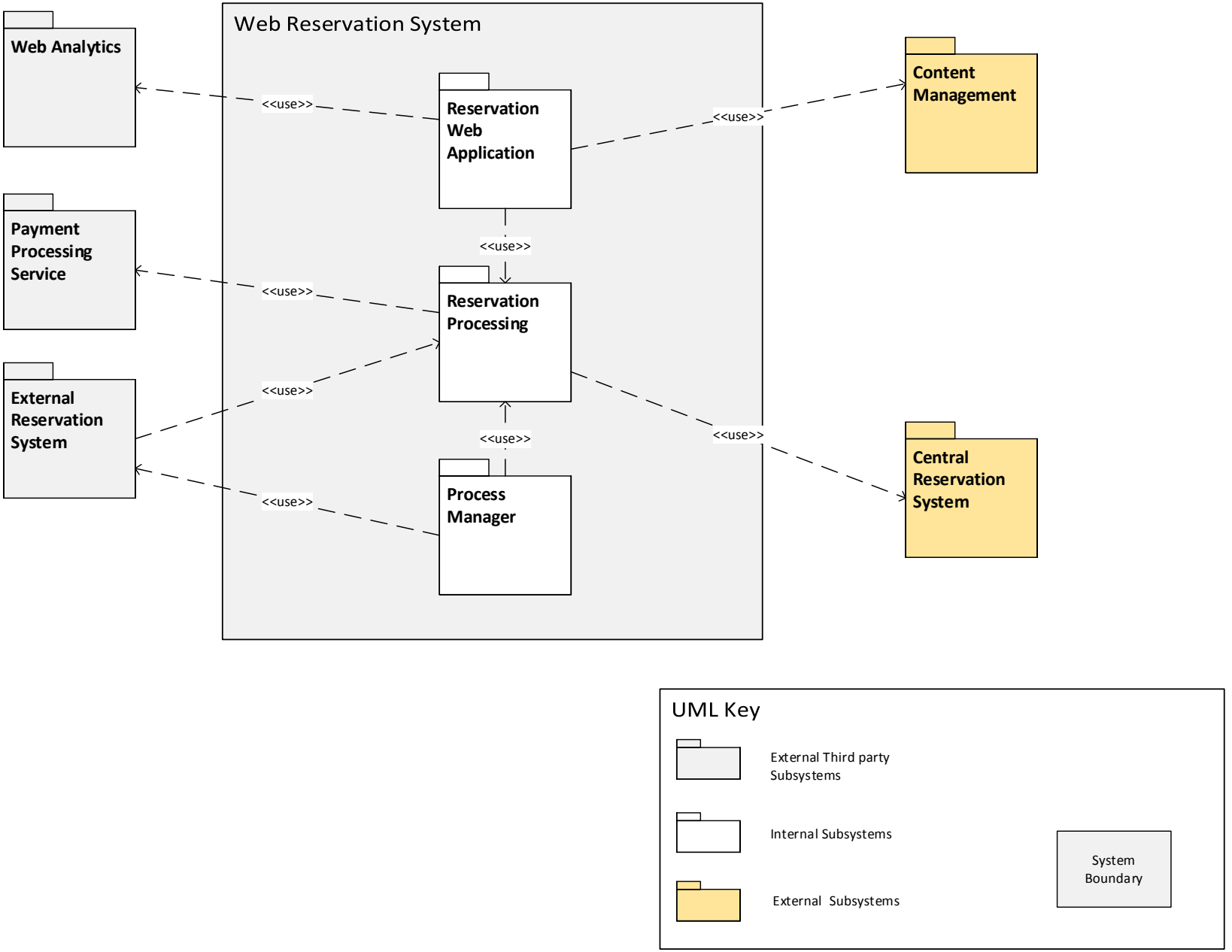


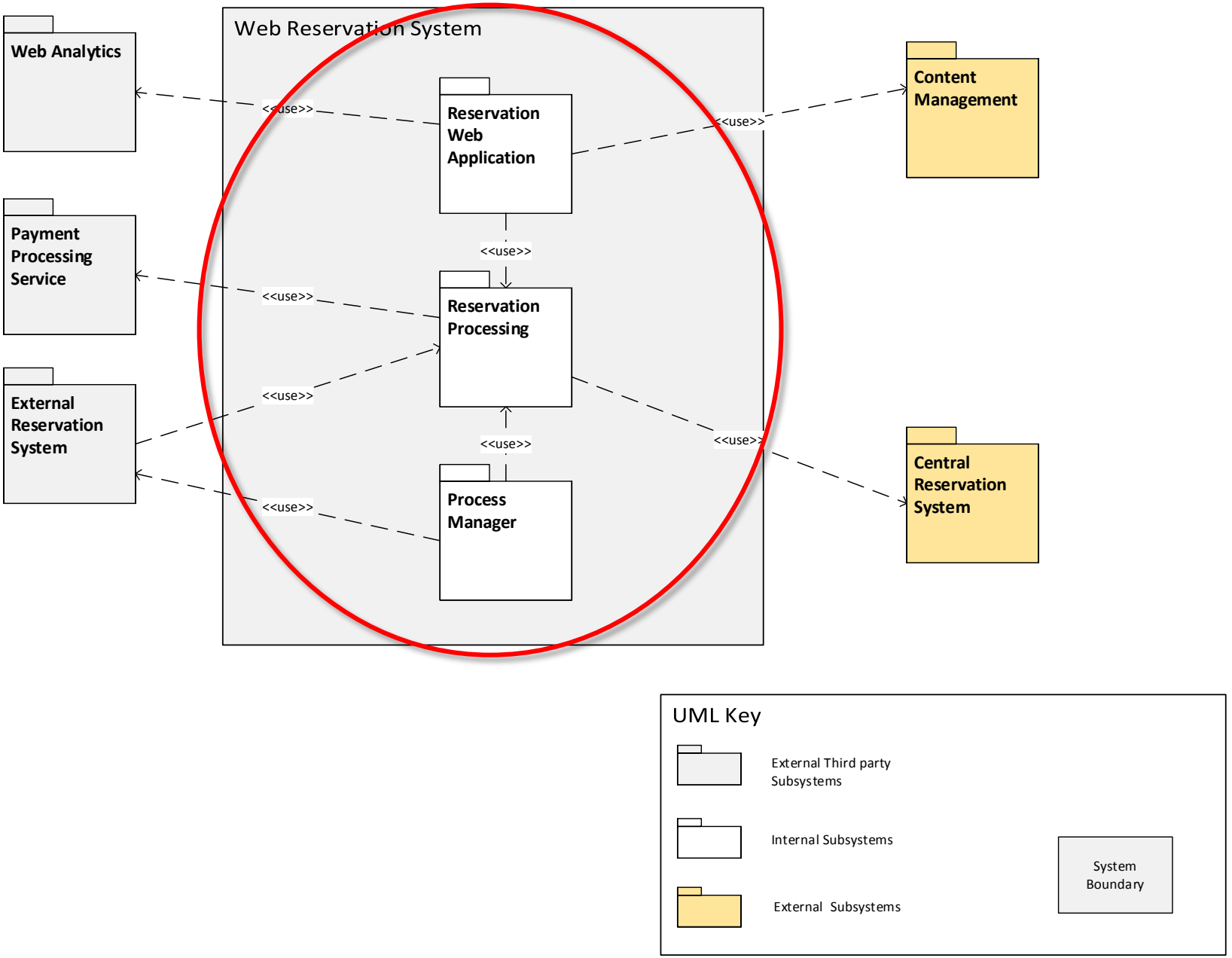
- Structural view of the architecture
- How modules are organized and how they interact
- Blueprint
- Identify all the major subsystems in the solution
- Module decomposition diagram
- Boundaries of the solution, all the subsystems and their interactions

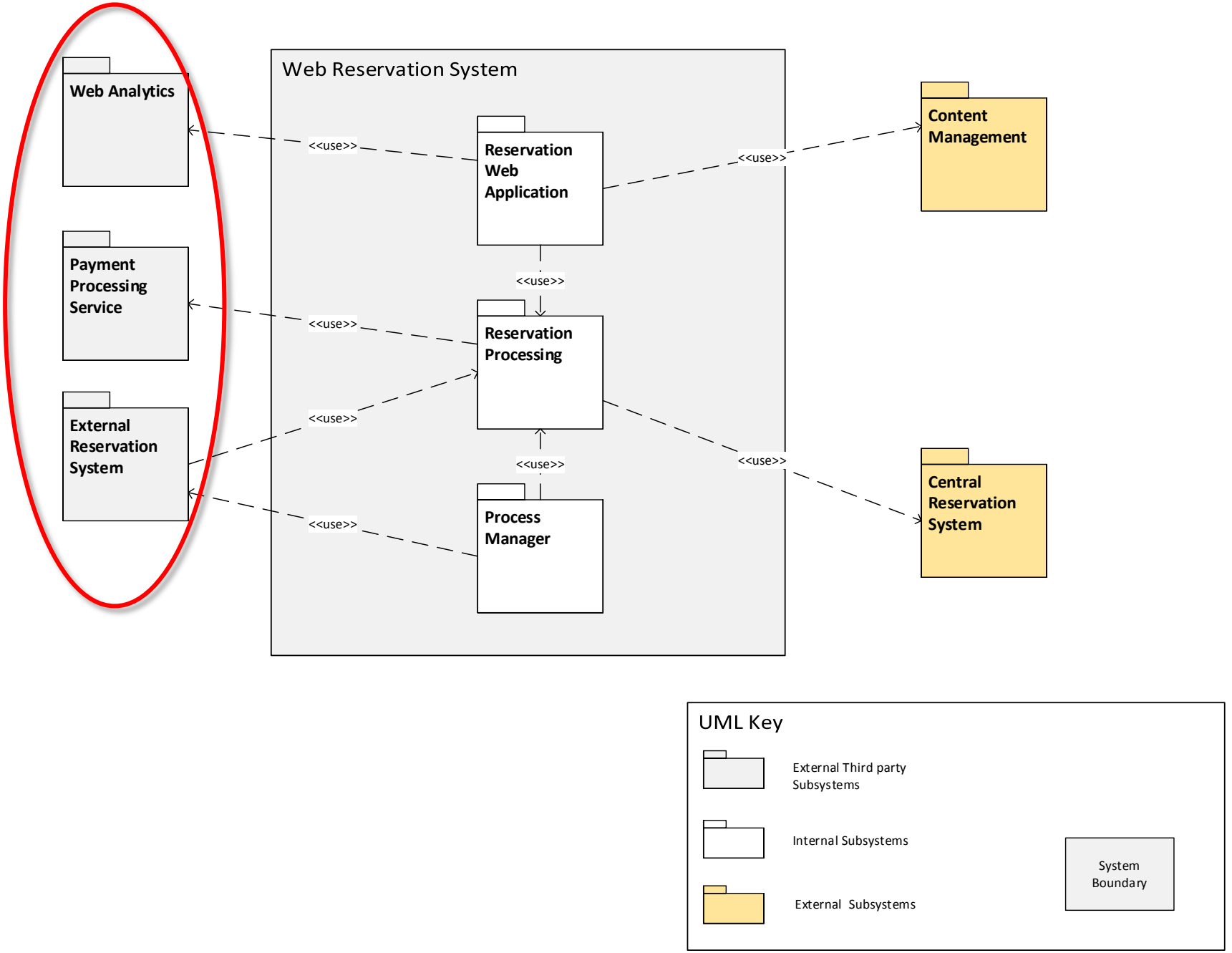
Views & Beyond in practice

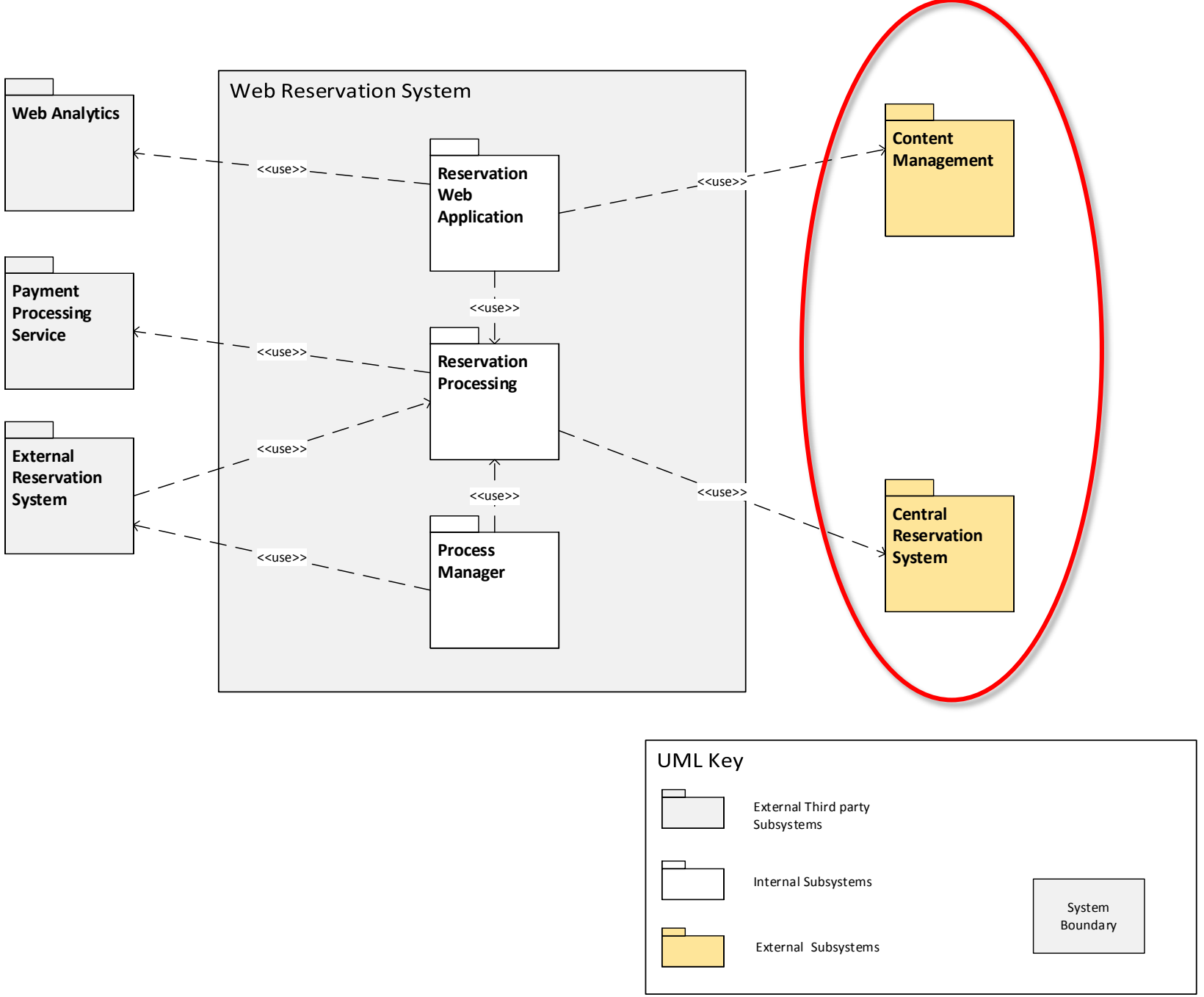


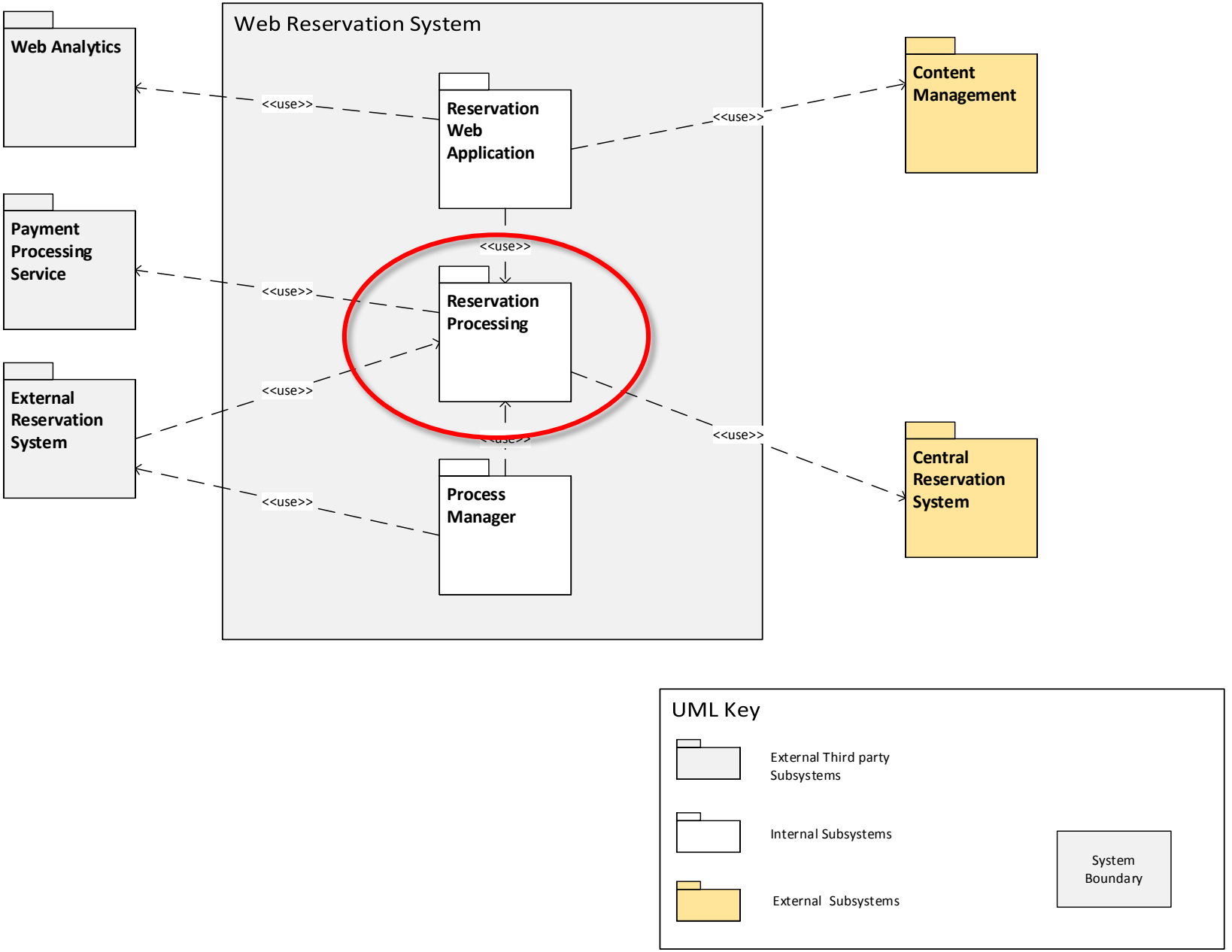
- **Module decomposition - Level 1**
  - View of the entire solution from a very high level
  - Technical and non-technical team members
  - Identify all system boundaries & subsystems

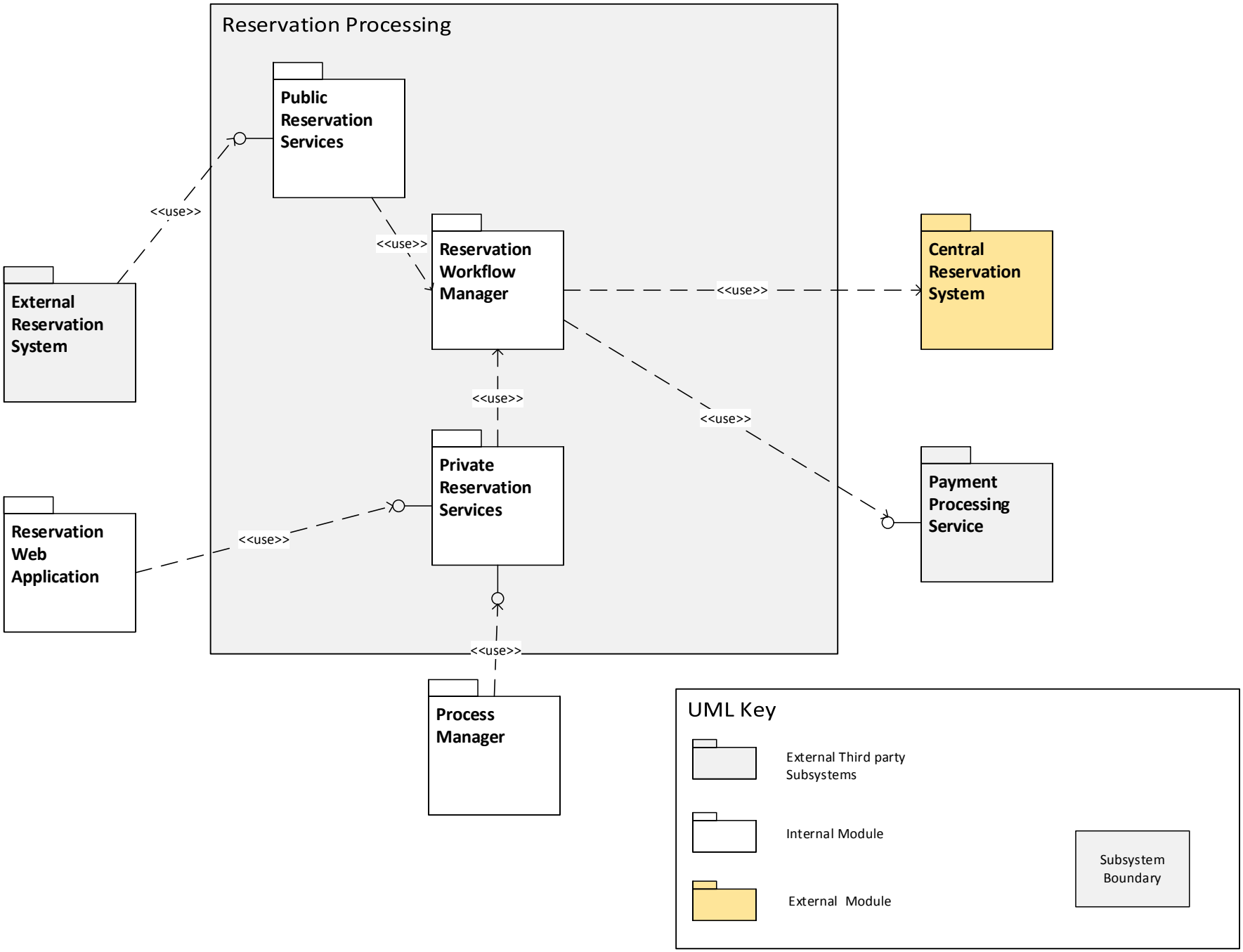

Web Reservation System
Web Analytics
Payment Processing Service
External Reservation System
Reservation Web Application
Reservation Processing
Process Manager
Content Management
Central Reservation System
<<use>>
<<use>>
<<use>>
<<use>>
<<use>>
<<use>>
<<use>>
UML Key
External Third party Subsystems
Internal Subsystems
External Subsystems
System Boundary


Web Reservation System
Web Analytics
Payment Processing Service
External Reservation System
Reservation Web Application
Reservation Processing
Process Manager
Content Management
Central Reservation System
<<use>>
<<use>>
<<use>>
<<use>>
<<use>>
<<use>>
<<use>>
UML Key
External Third party Subsystems
Internal Subsystems
External Subsystems
System Boundary

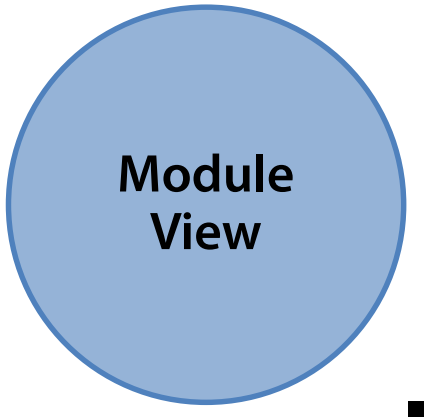

Web Reservation System
Web Analytics
Payment Processing Service
External Reservation System
Reservation Web Application
Reservation Processing
Process Manager
Content Management
Central Reservation System
<<use>>
<<use>>
<<use>>
<<use>>
<<use>>
<<use>>
<<use>>
UML Key
External Third party Subsystems
Internal Subsystems
External Subsystems
System Boundary


Web Reservation System
Web Analytics
Payment Processing Service
External Reservation System
Reservation Web Application
Reservation Processing
Process Manager
Content Management
Central Reservation System
<<use>>
<<use>>
<<use>>
<<use>>
<<use>>
<<use>>
<<use>>
UML Key
External Third party Subsystems
Internal Subsystems
External Subsystems
System Boundary

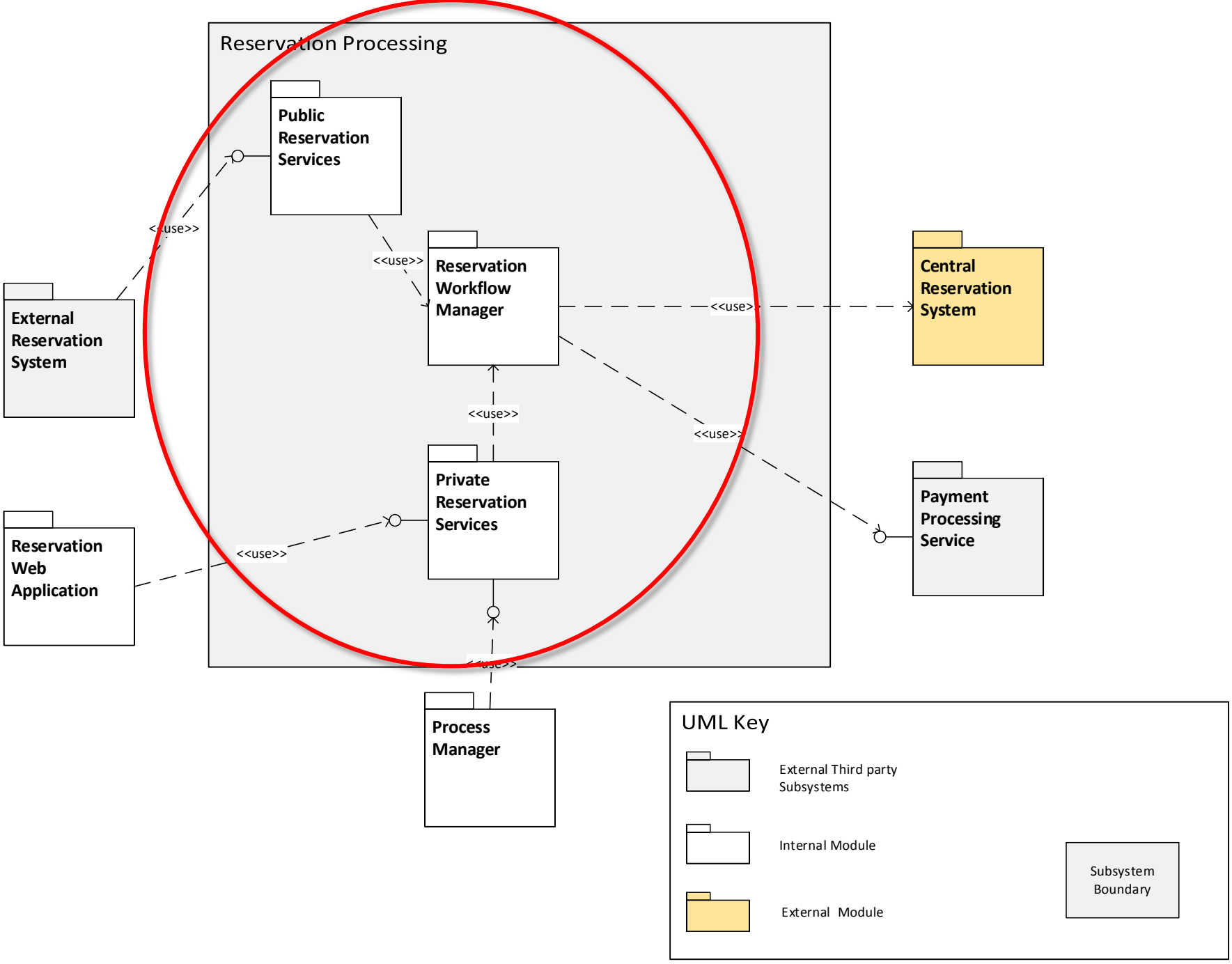

Web Reservation System
Web Analytics
Payment Processing Service
External Reservation System
Reservation Web Application
Reservation Processing
Process Manager
Content Management
Central Reservation System
<<use>>
<<use>>
<<use>>
<<use>>
<<use>>
<<use>>
<<use>>
UML Key
External Third party Subsystems
Internal Subsystems
External Subsystems
System Boundary

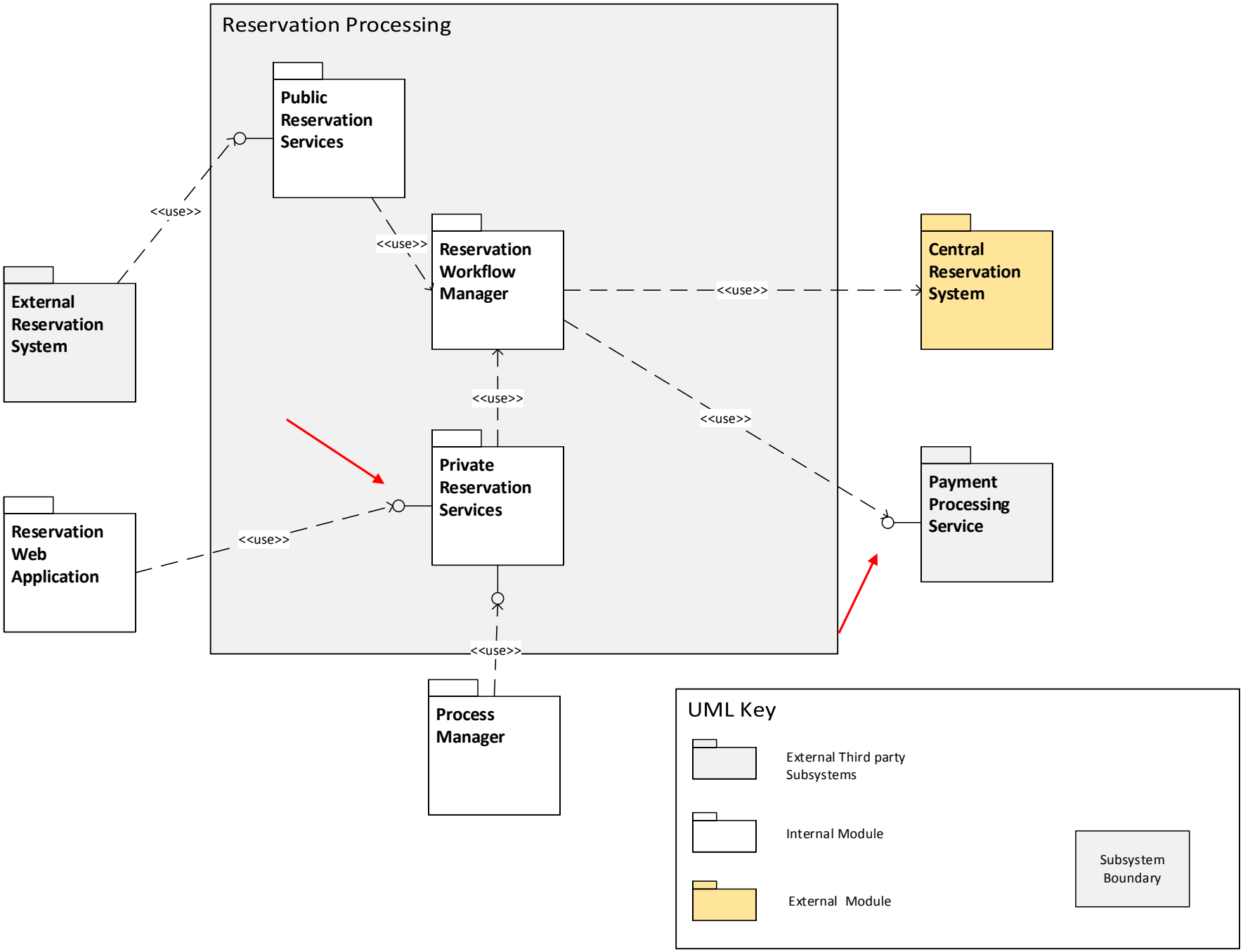

Reservation Processing
Public Reservation Services
Reservation Workflow Manager
Private Reservation Services
Process Manager
External Reservation System
Reservation Web Application
Central Reservation System
Payment Processing Service
<<use>>
<<use>>
<<use>>
<<use>>
<<use>>
<<use>>
UML Key
External Third party Subsystems
Internal Module
External Module
Subsystem Boundary

Views & Beyond in practice

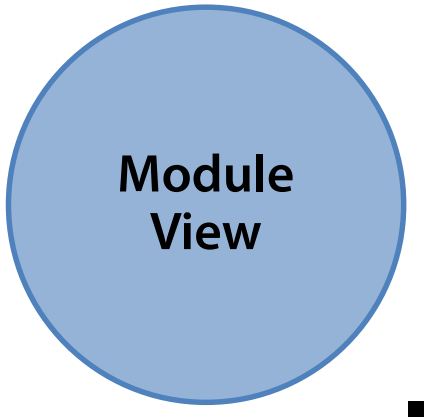


- **Module decomposition - Level 2**
  - Individual subsystems that were identified in the top level diagram
  - Define all modules that make up each subsystem
  - Subsystems contain one or many modules
  - Single logical area of responsibility
  - High-level groupings of other modules
  - Interoperation between modules is important

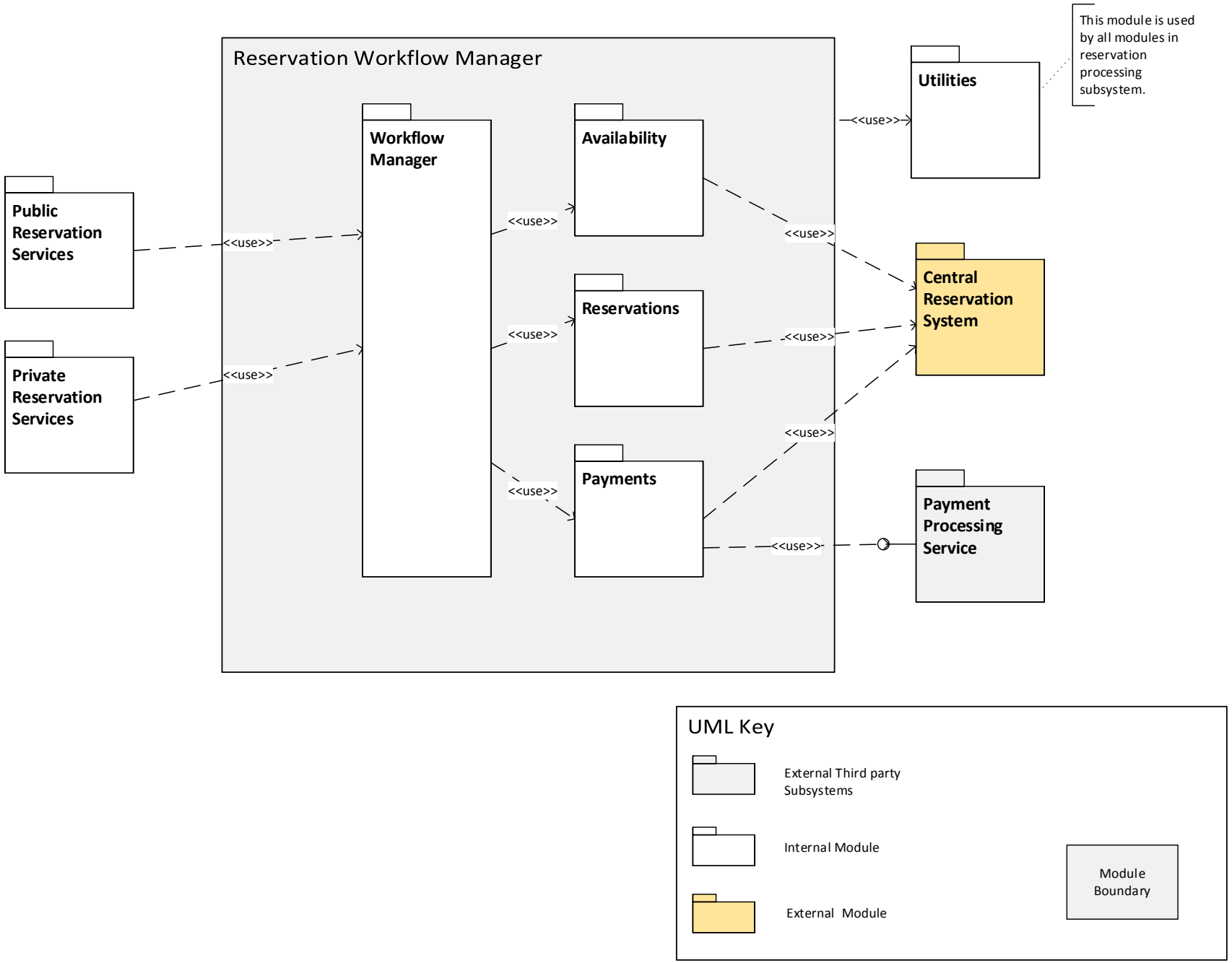

Reservation Processing
Public Reservation Services
External Reservation System
Reservation Workflow Manager
Central Reservation System
Private Reservation Services
Payment Processing Service
Reservation Web Application
Process Manager
<<use>>
UML Key
External Third party Subsystems
Internal Module
External Module
Subsystem Boundary

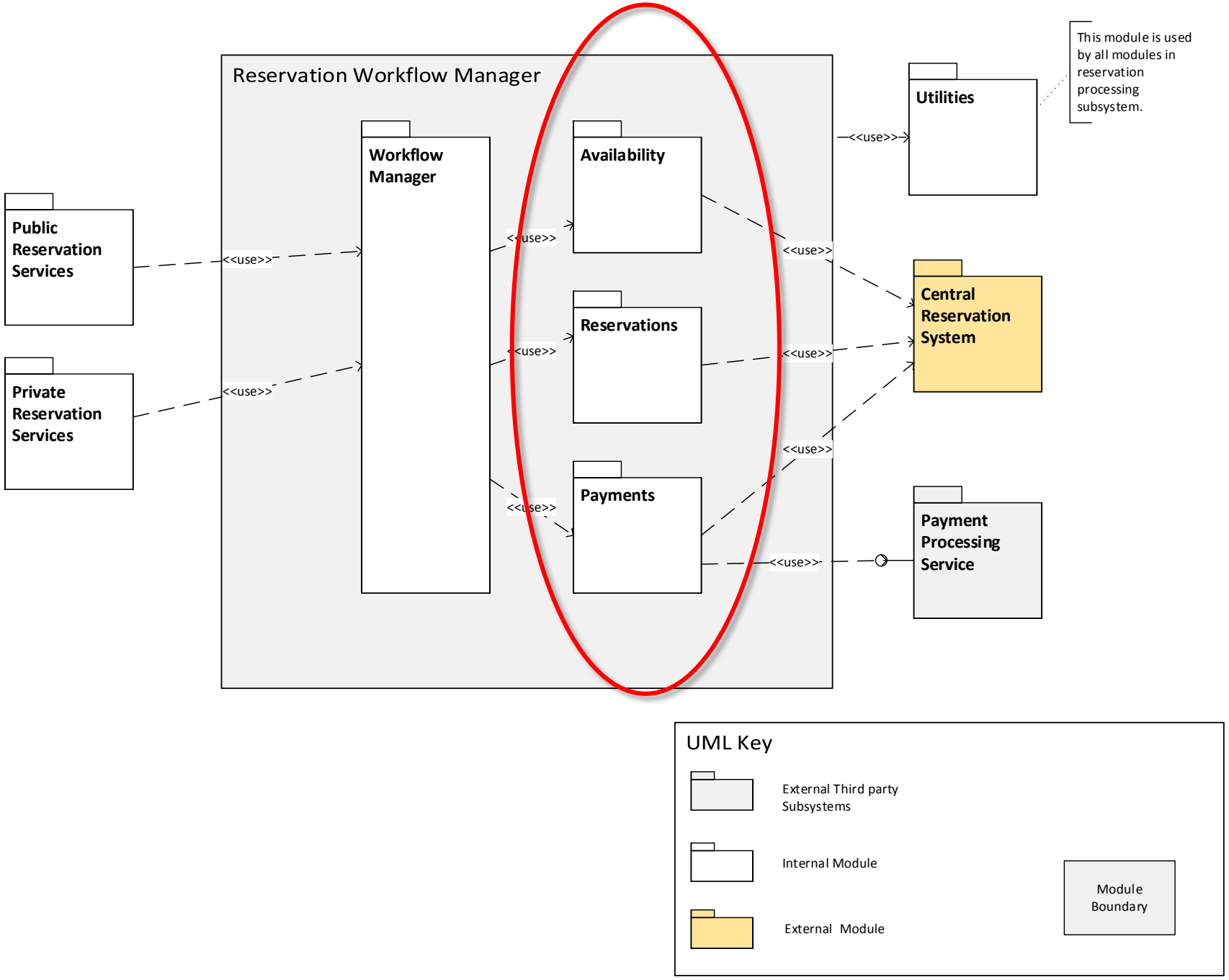

Reservation Processing
Public Reservation Services
Reservation Workflow Manager
Private Reservation Services
Process Manager
External Reservation System
Reservation Web Application
Central Reservation System
Payment Processing Service
<<use>>
<<use>>
<<use>>
<<use>>
<<use>>
<<use>>
UML Key
External Third party Subsystems
Internal Module
External Module
Subsystem Boundary

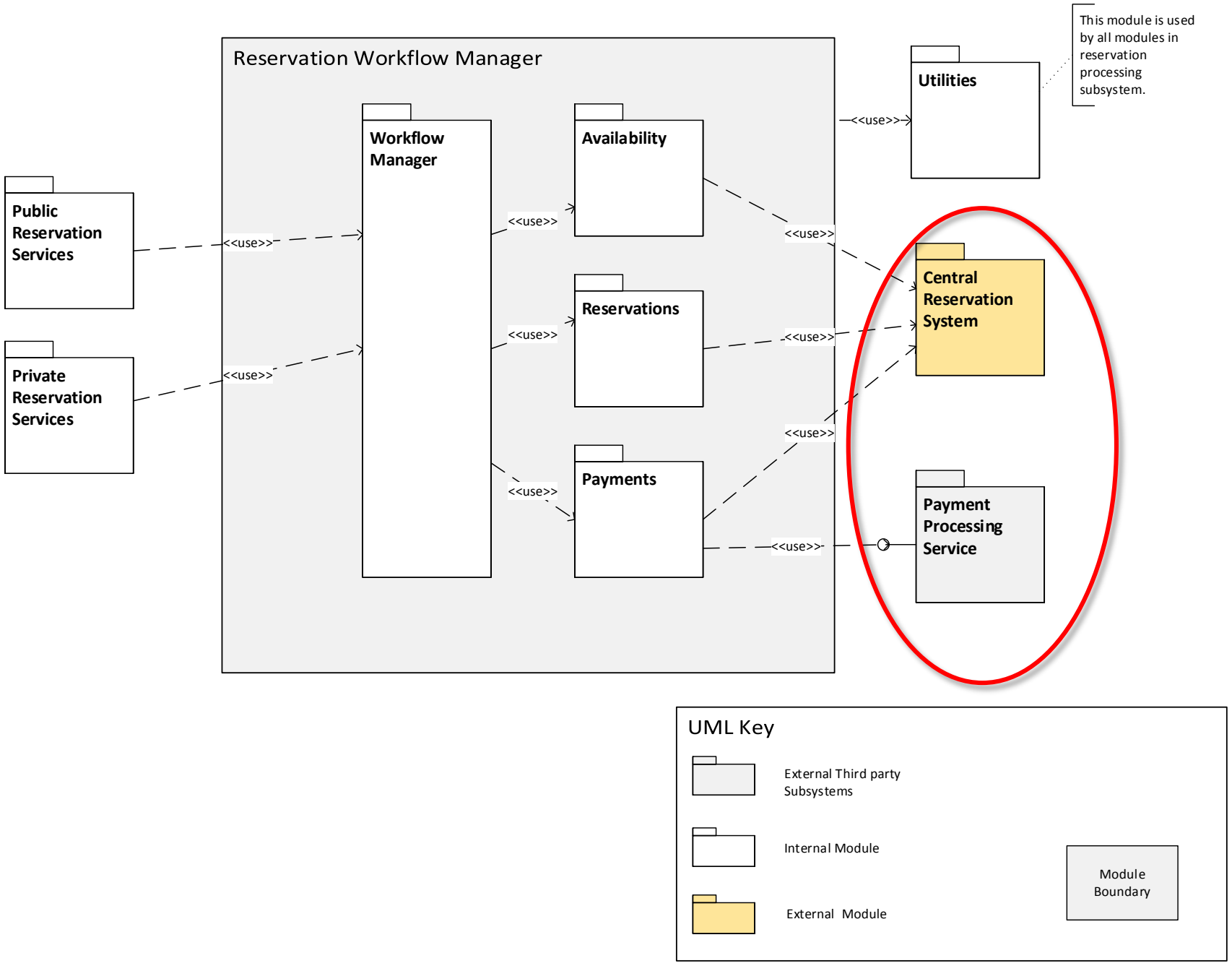
Views & Beyond in practice



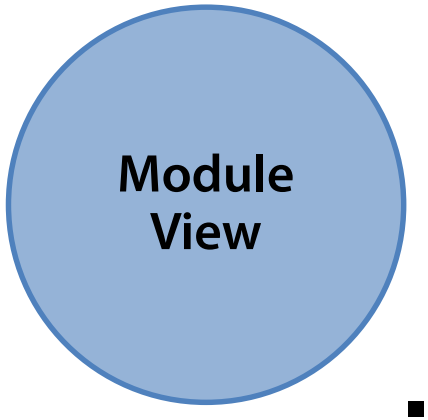
- **Module decomposition - Level 3**
  - No hard and fast rule that defines how many levels
  - Decompose your system to as many levels as needed
  - Decomposition is performed iteratively


Reservation Workflow Manager
Workflow Manager
Availability
Reservations
Payments
Utilities
This module is used by all modules in reservation processing subsystem.
Public Reservation Services
Private Reservation Services
Central Reservation System
Payment Processing Service
<<use>>
UML Key
External Third party Subsystems
Internal Module
External Module
Module Boundary


Reservation Workflow Manager
Workflow Manager
Availability
Reservations
Payments
Public Reservation Services
Private Reservation Services
Utilities
This module is used by all modules in reservation processing subsystem.
Central Reservation System
Payment Processing Service
<<use>>
UML Key
External Third party Subsystems
Internal Module
External Module
Module Boundary

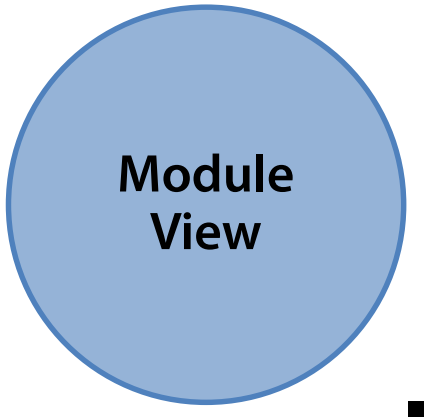

Reservation Workflow Manager
Workflow Manager
Availability
Reservations
Payments
Utilities
This module is used by all modules in reservation processing subsystem.
Public Reservation Services
Private Reservation Services
Central Reservation System
Payment Processing Service
<<use>>
UML Key
External Third party Subsystems
Internal Module
External Module
Module Boundary

Views & Beyond in practice

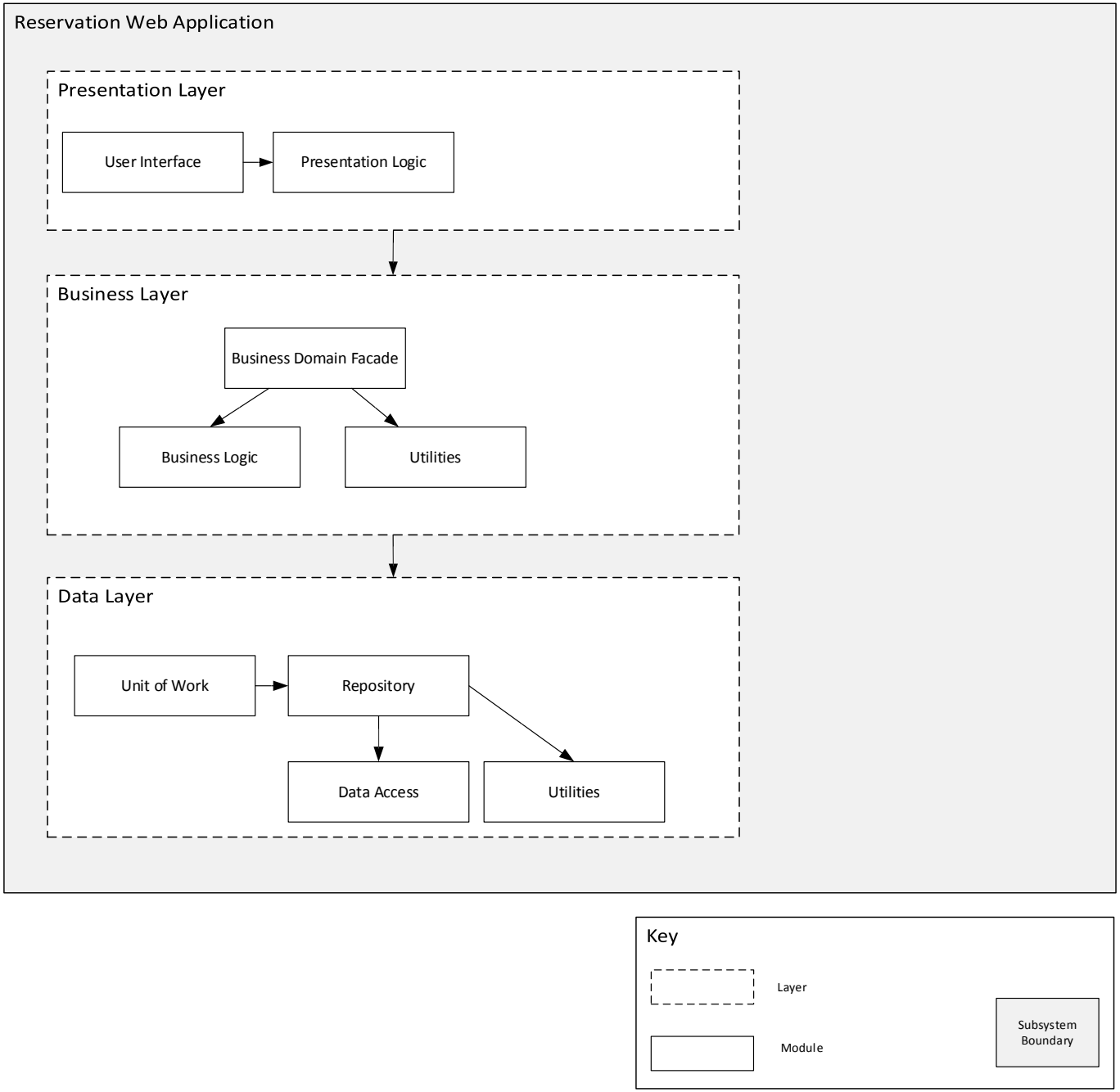


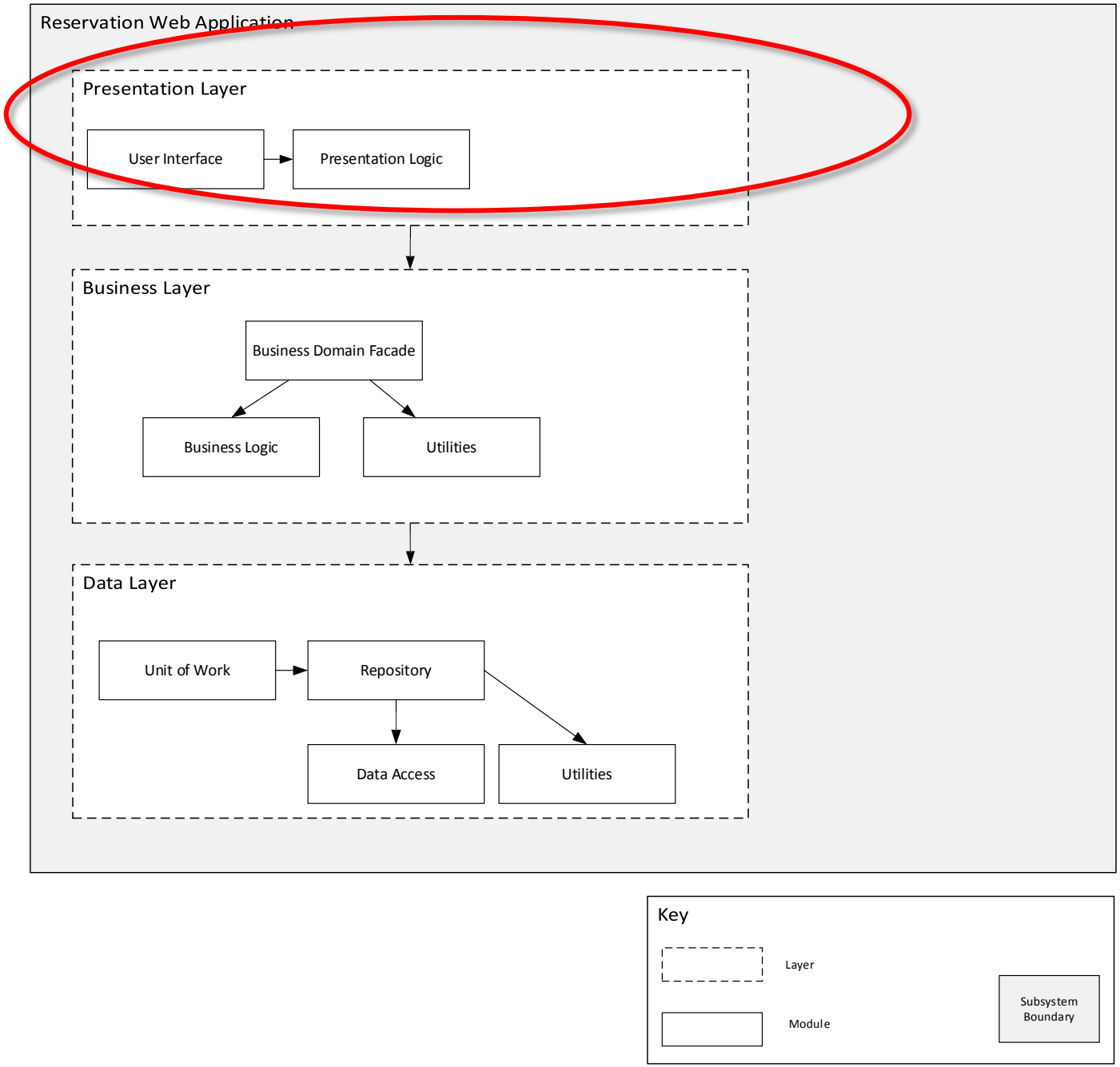
- **Module decomposition**
  - Continues to lowest level of detail that provides value
  - Not necessary to decompose every subsystem and every module
  - Guided by practicality

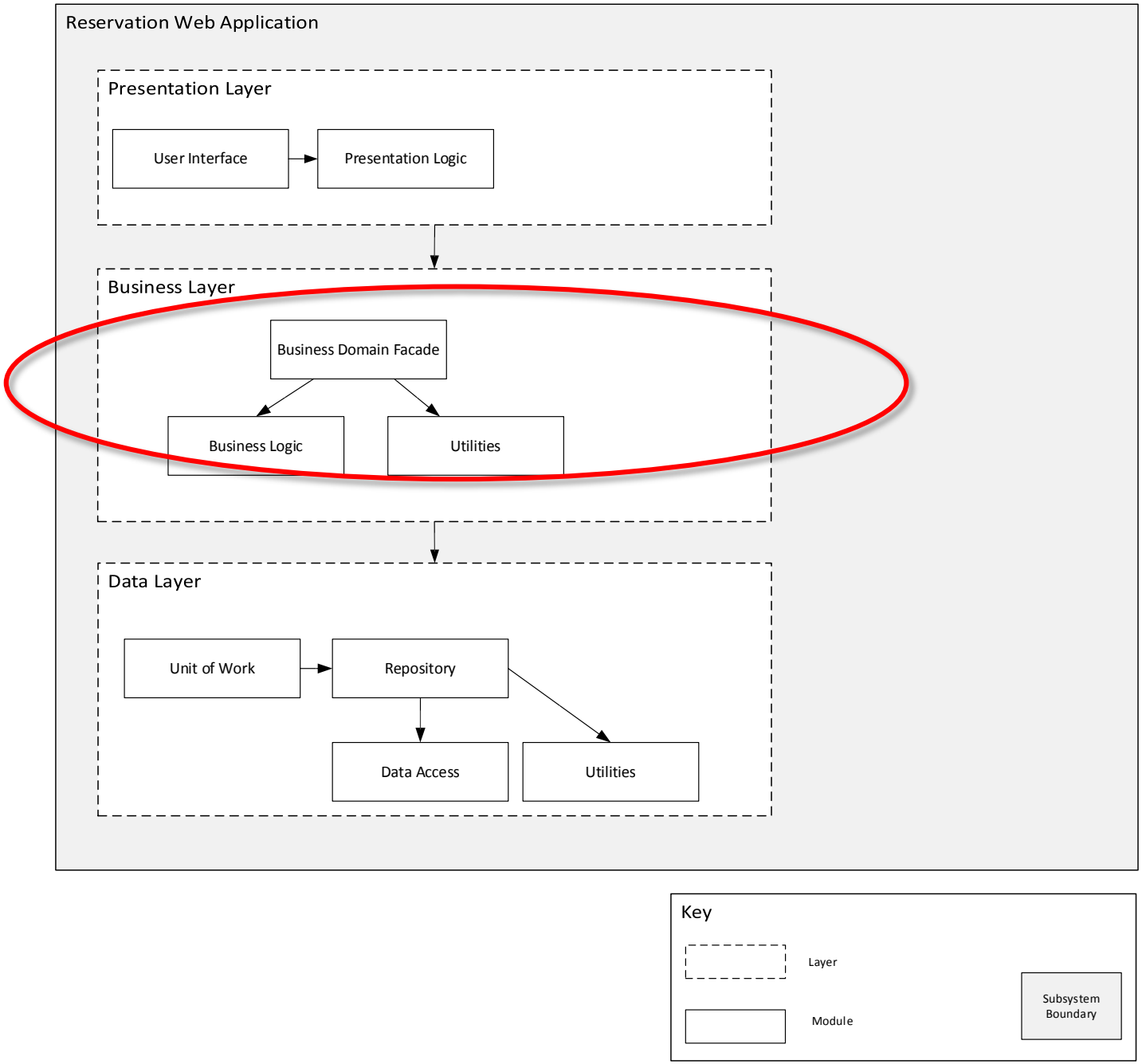
Views & Beyond in practice

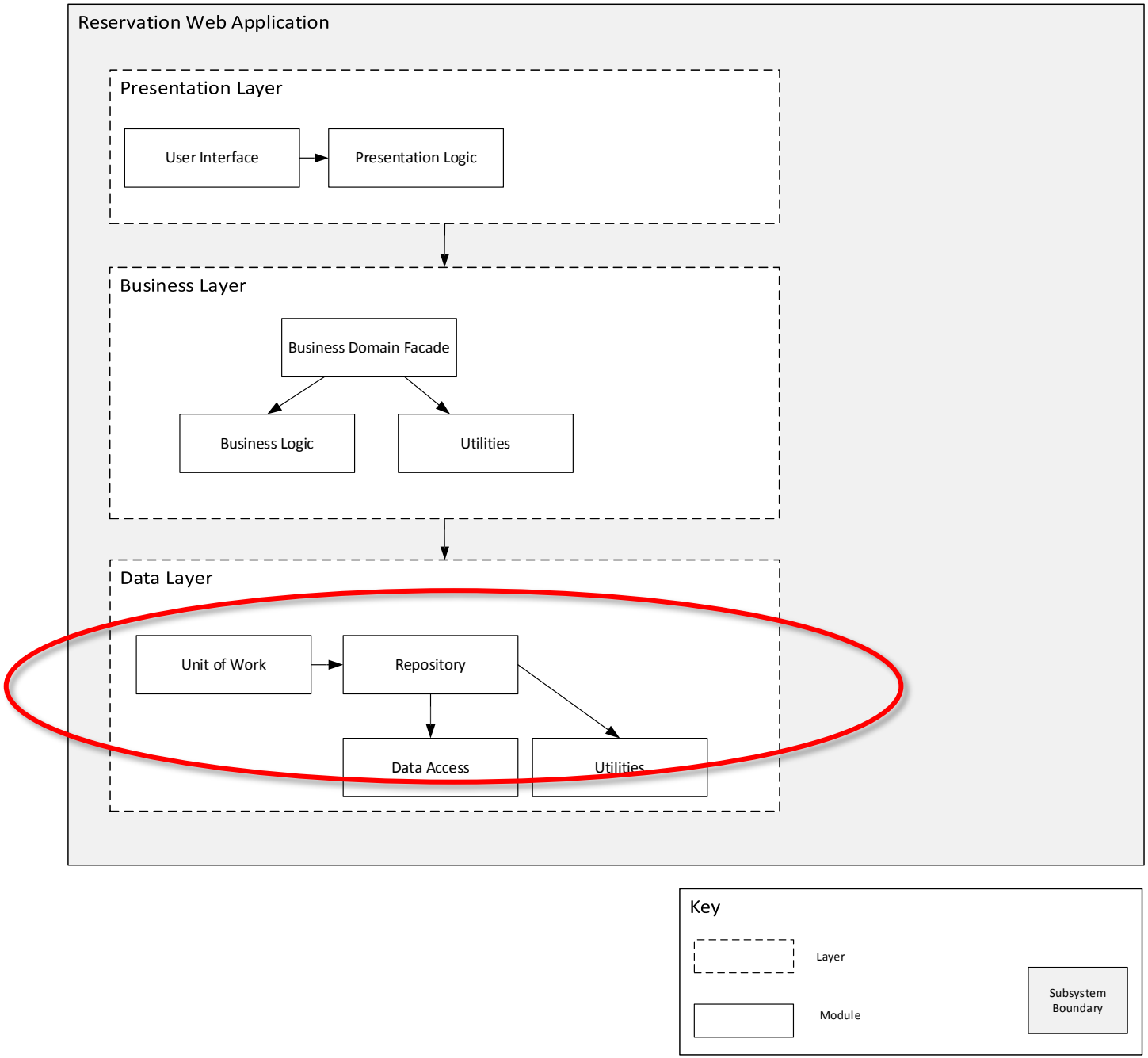


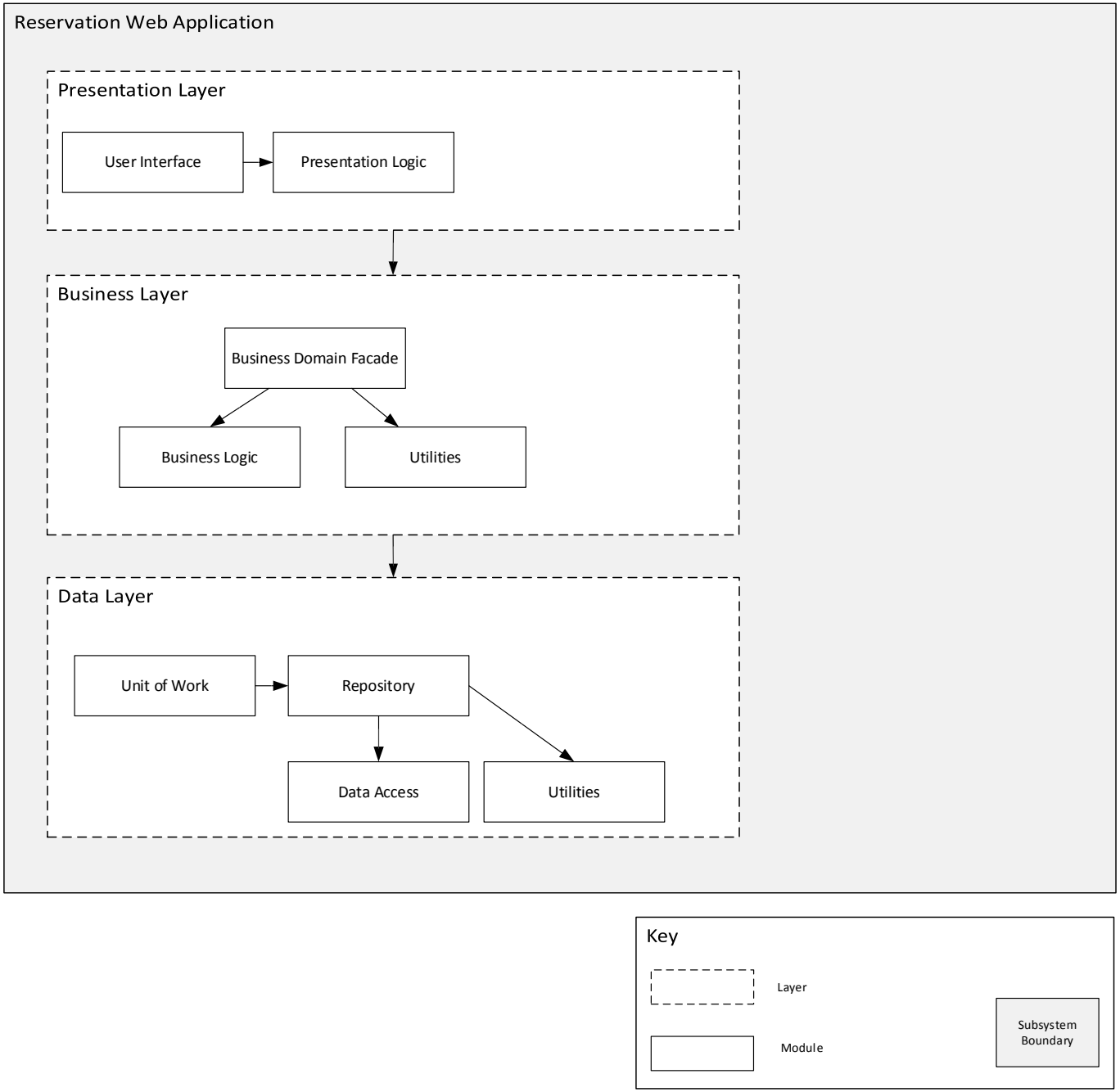
- **Layered**
  - Separated logically by responsibility
  - Contain modules that make up each layer
  - Loosely coupled
  - Cohesive
  - Manage dependencies
  - Flexibility
  - Maintainability
  - Layers are logical
  - Tiers are physical

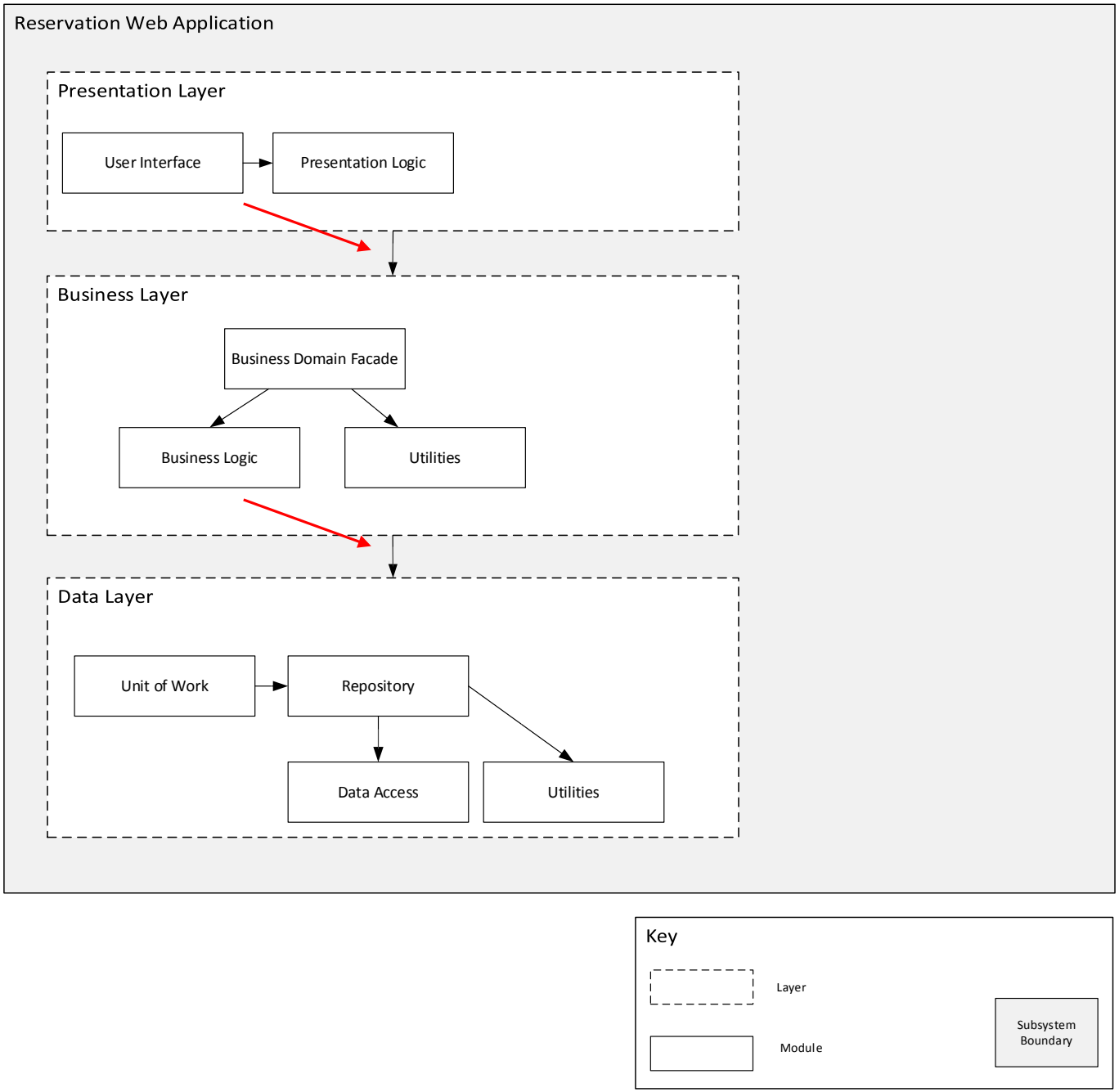

Reservation Web Application
Presentation Layer
User Interface
Presentation Logic
Business Layer
Business Domain Facade
Business Logic
Utilities
Data Layer
Unit of Work
Repository
Data Access
Utilities
Key
Layer
Module
Subsystem Boundary

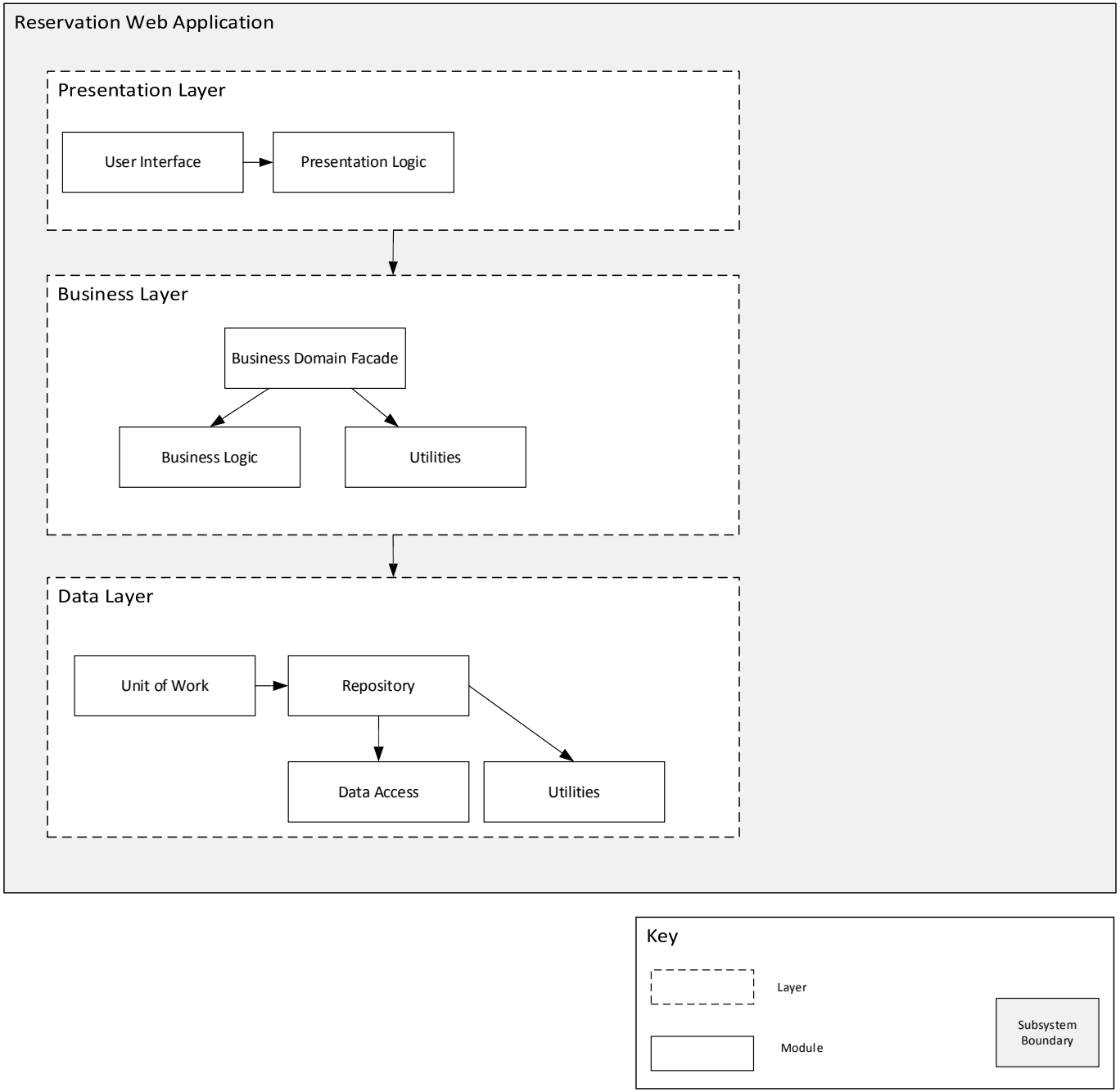

Reservation Web Application
Presentation Layer
User Interface
Presentation Logic
Business Layer
Business Domain Facade
Business Logic
Utilities
Data Layer
Unit of Work
Repository
Data Access
Utilities
Key
Layer
Module
Subsystem Boundary

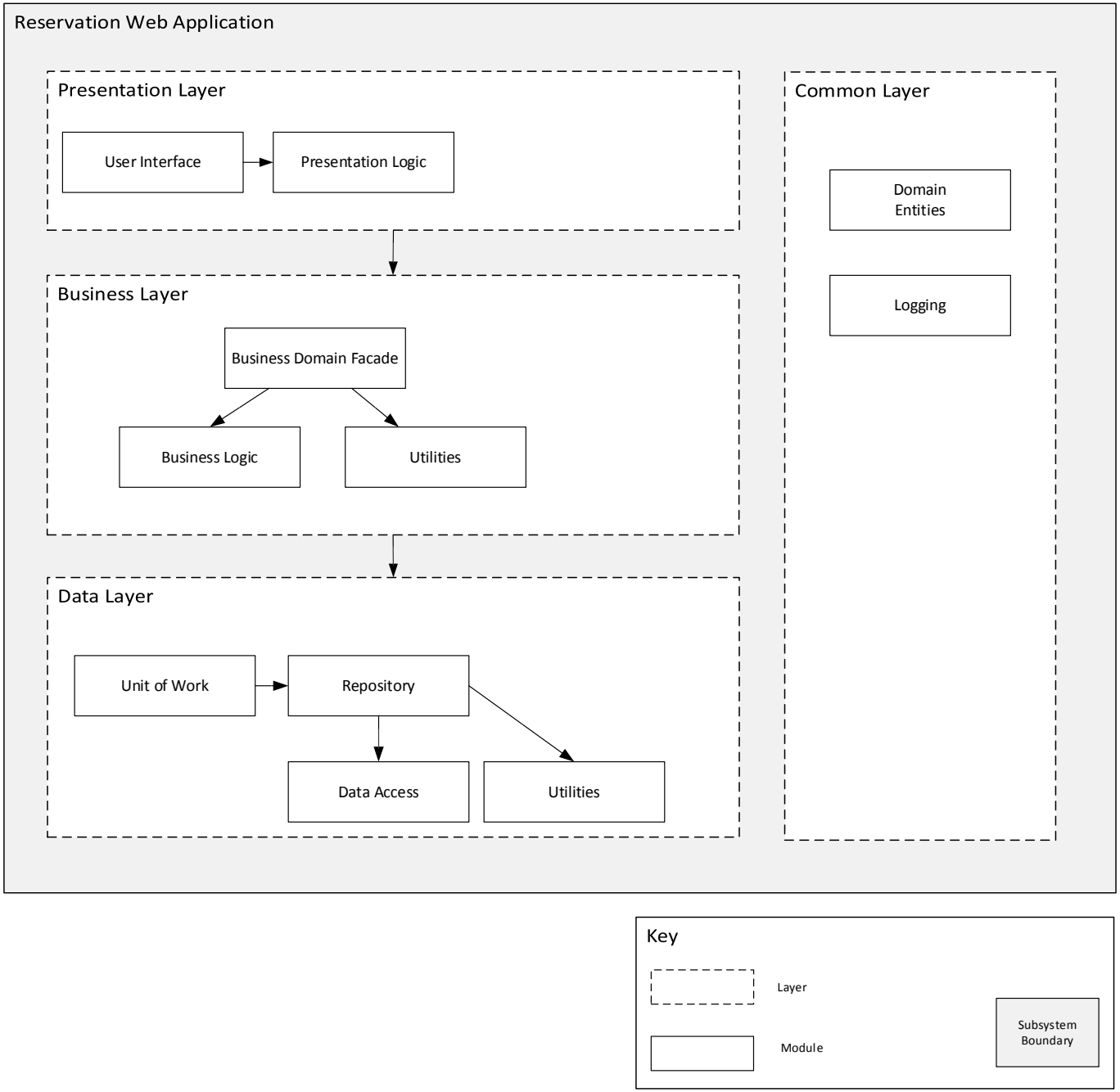

Reservation Web Application
Presentation Layer
User Interface
Presentation Logic
Business Layer
Business Domain Facade
Business Logic
Utilities
Data Layer
Unit of Work
Repository
Data Access
Utilities
Key
Layer
Module
Subsystem Boundary

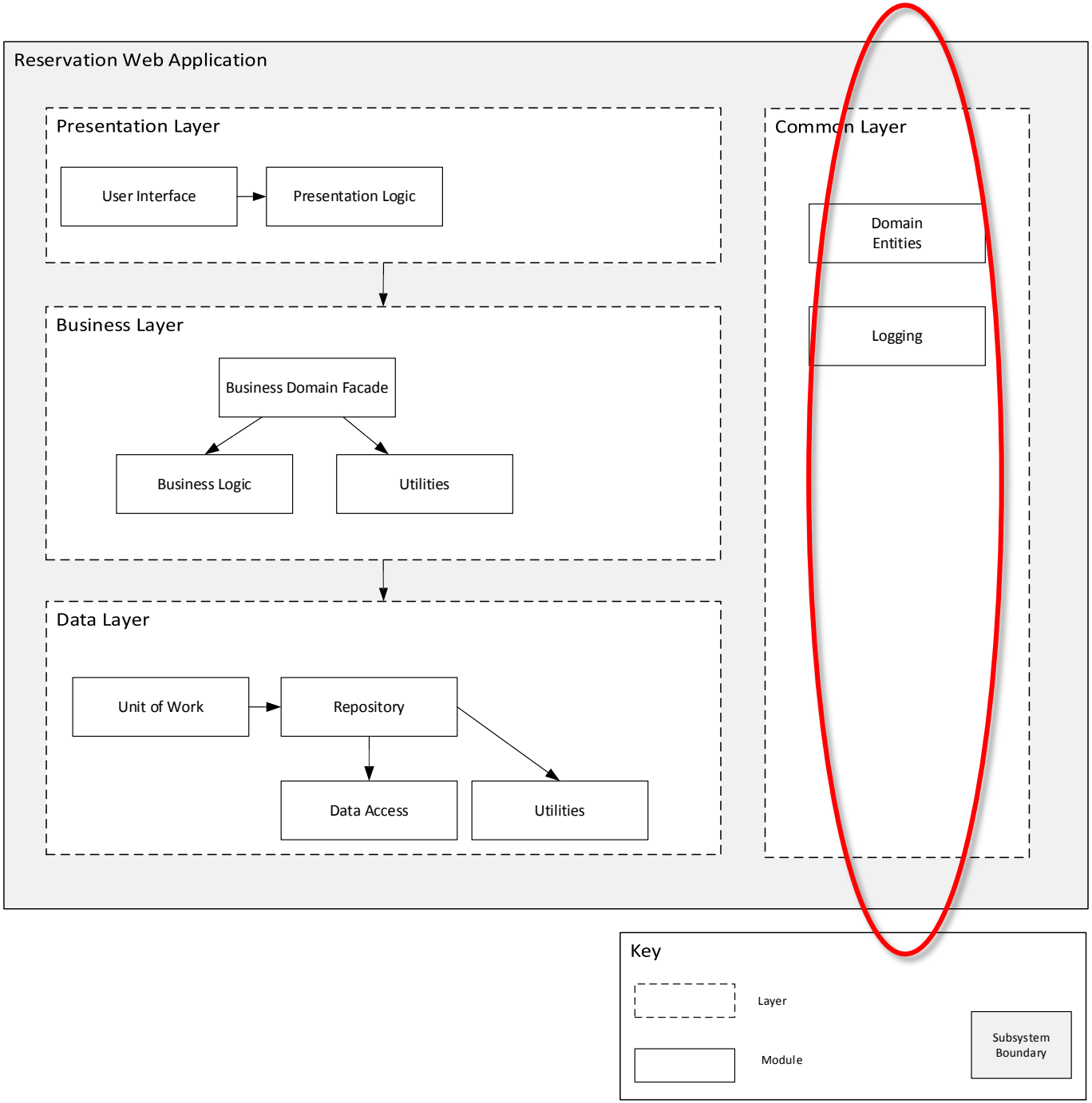

Reservation Web Application
Presentation Layer
User Interface
Presentation Logic
Business Layer
Business Domain Facade
Business Logic
Utilities
Data Layer
Unit of Work
Repository
Data Access
Utilities
Key
Layer
Module
Subsystem Boundary


Reservation Web Application
Presentation Layer
User Interface
Presentation Logic
Business Layer
Business Domain Facade
Business Logic
Utilities
Data Layer
Unit of Work
Repository
Data Access
Utilities
Key
Layer
Module
Subsystem Boundary

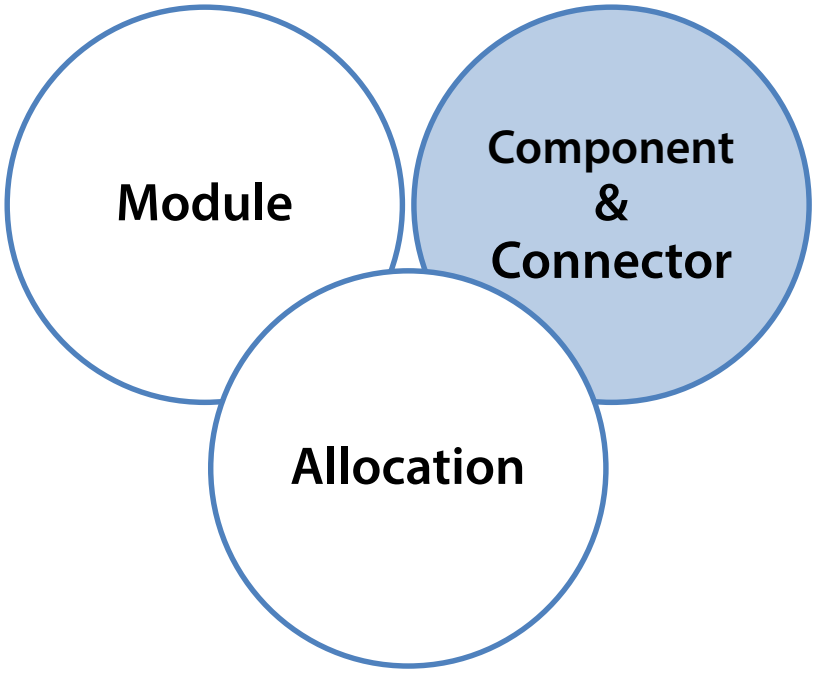

Reservation Web Application
Presentation Layer
User Interface
Presentation Logic
Business Layer
Business Domain Facade
Business Logic
Utilities
Data Layer
Unit of Work
Repository
Data Access
Utilities
Key
Layer
Module
Subsystem Boundary


Reservation Web Application
Presentation Layer
User Interface
Presentation Logic
Business Layer
Business Domain Facade
Business Logic
Utilities
Data Layer
Unit of Work
Repository
Data Access
Utilities
Key
Layer
Module
Subsystem Boundary

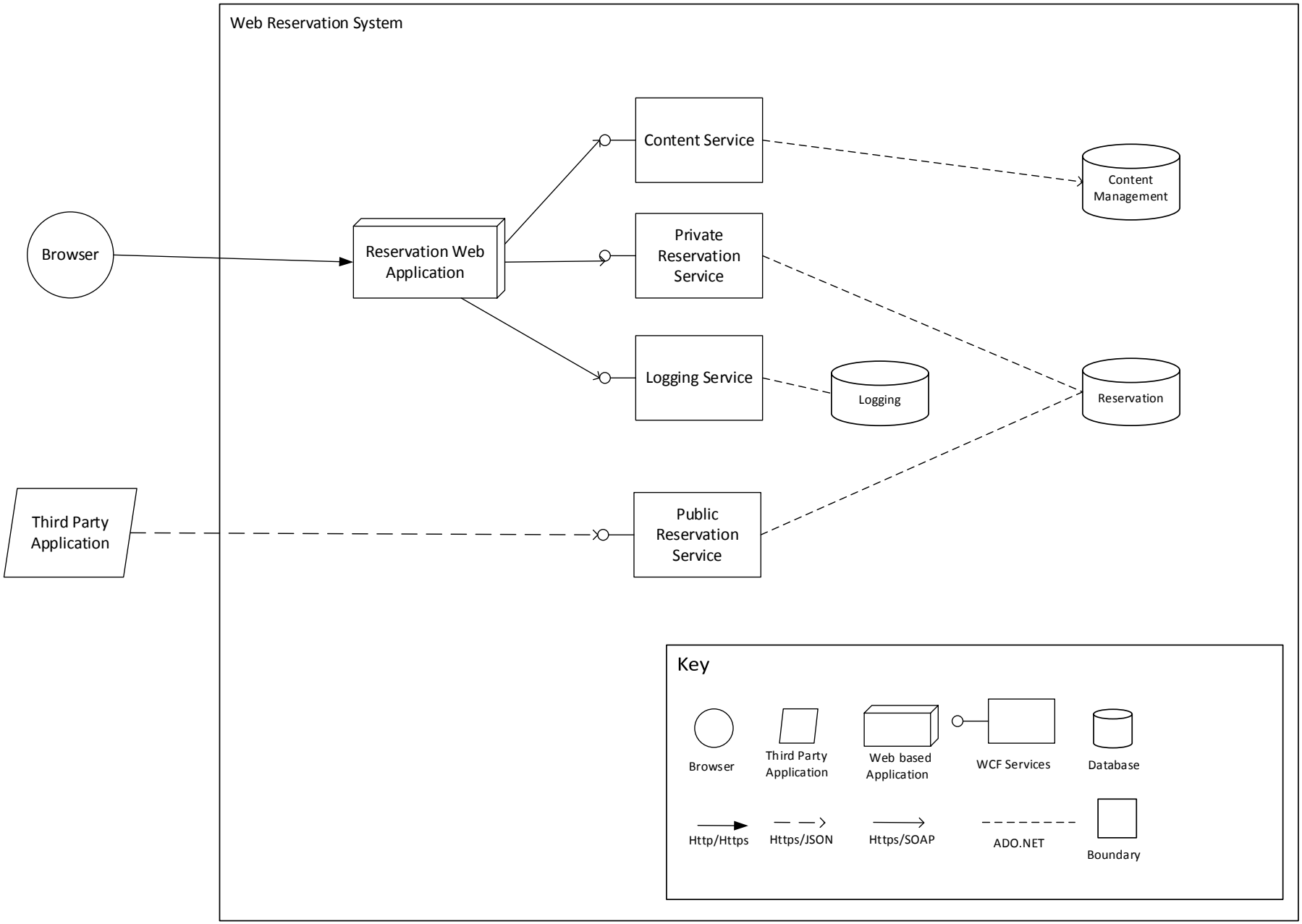

Reservation Web Application
Presentation Layer
User Interface
Presentation Logic
Business Layer
Business Domain Facade
Business Logic
Utilities
Data Layer
Unit of Work
Repository
Data Access
Utilities
Common Layer
Domain Entities
Logging
Key
Layer
Module
Subsystem Boundary

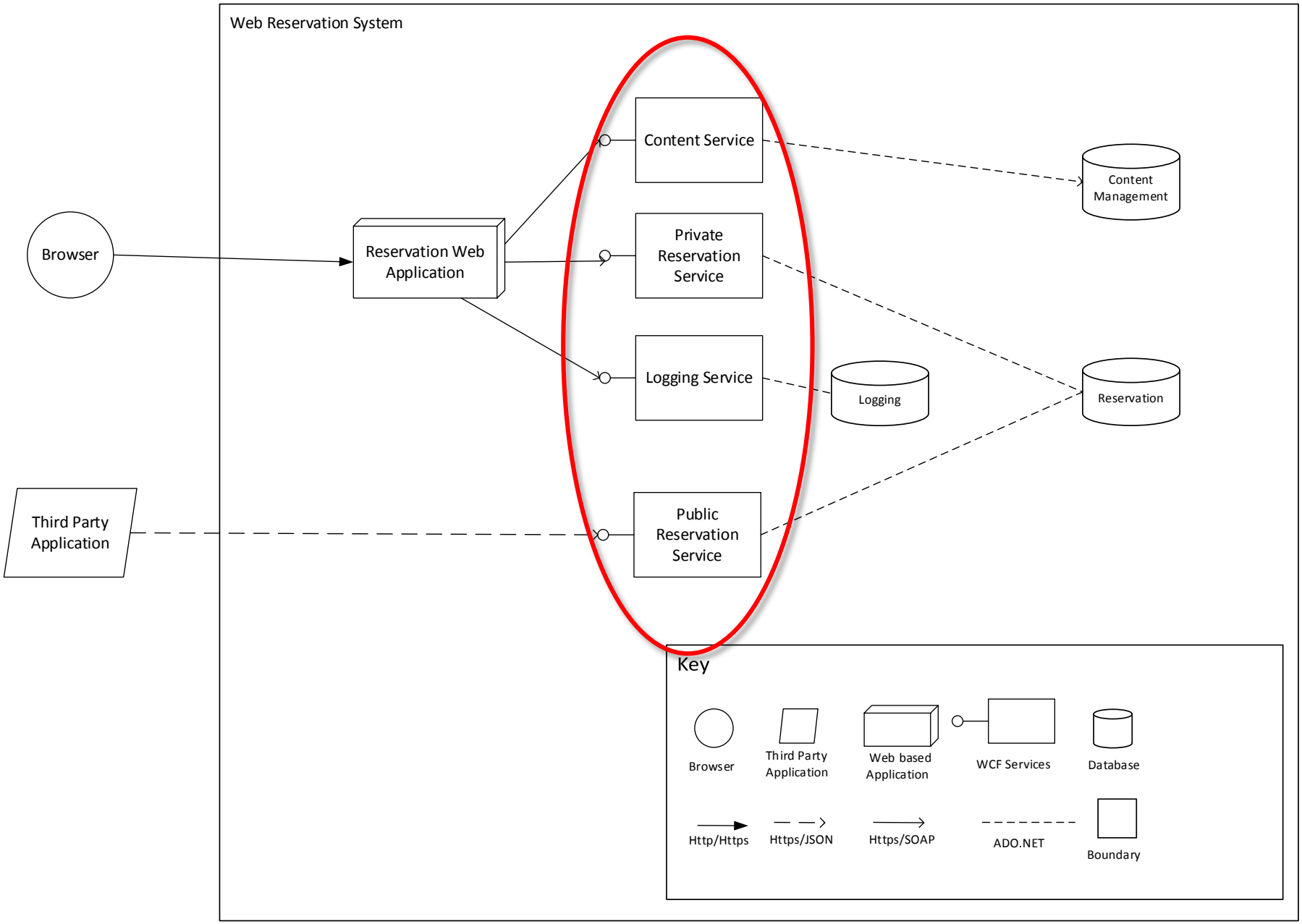

Reservation Web Application
Presentation Layer
User Interface
Presentation Logic
Business Layer
Business Domain Facade
Business Logic
Utilities
Data Layer
Unit of Work
Repository
Data Access
Utilities
Common Layer
Domain Entities
Logging
Key
Layer
Module
Subsystem Boundary

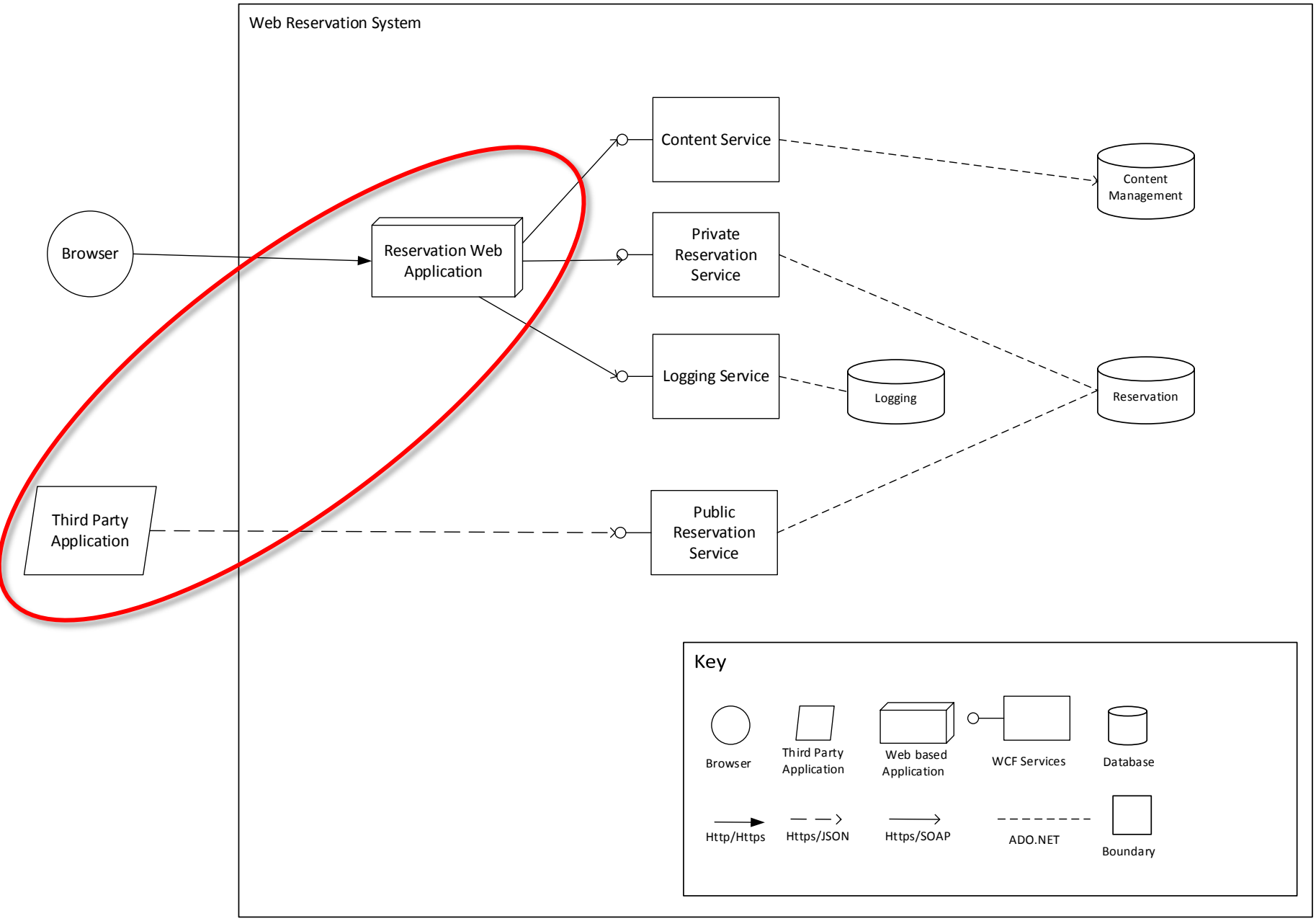
Views & Beyond in practice

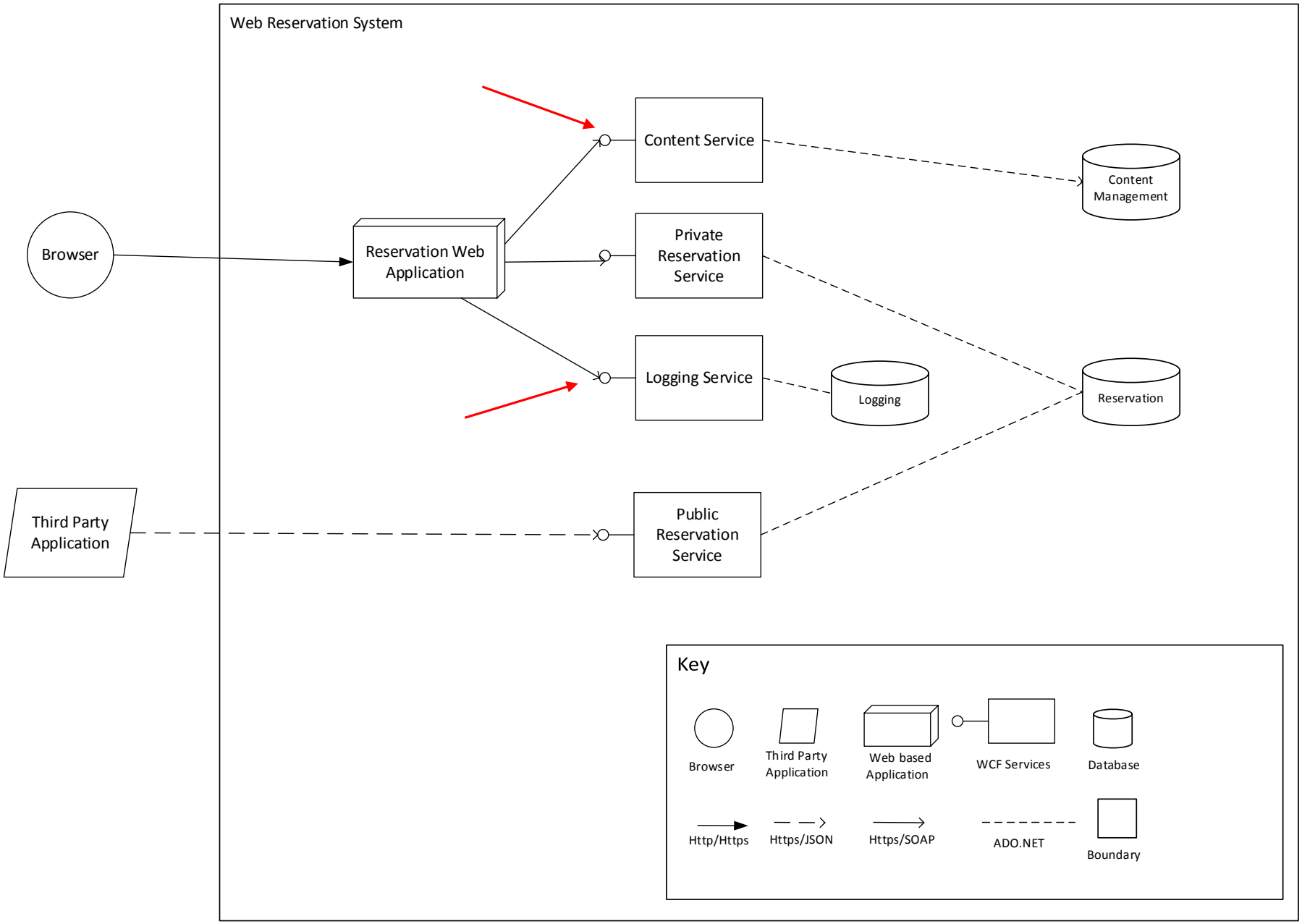


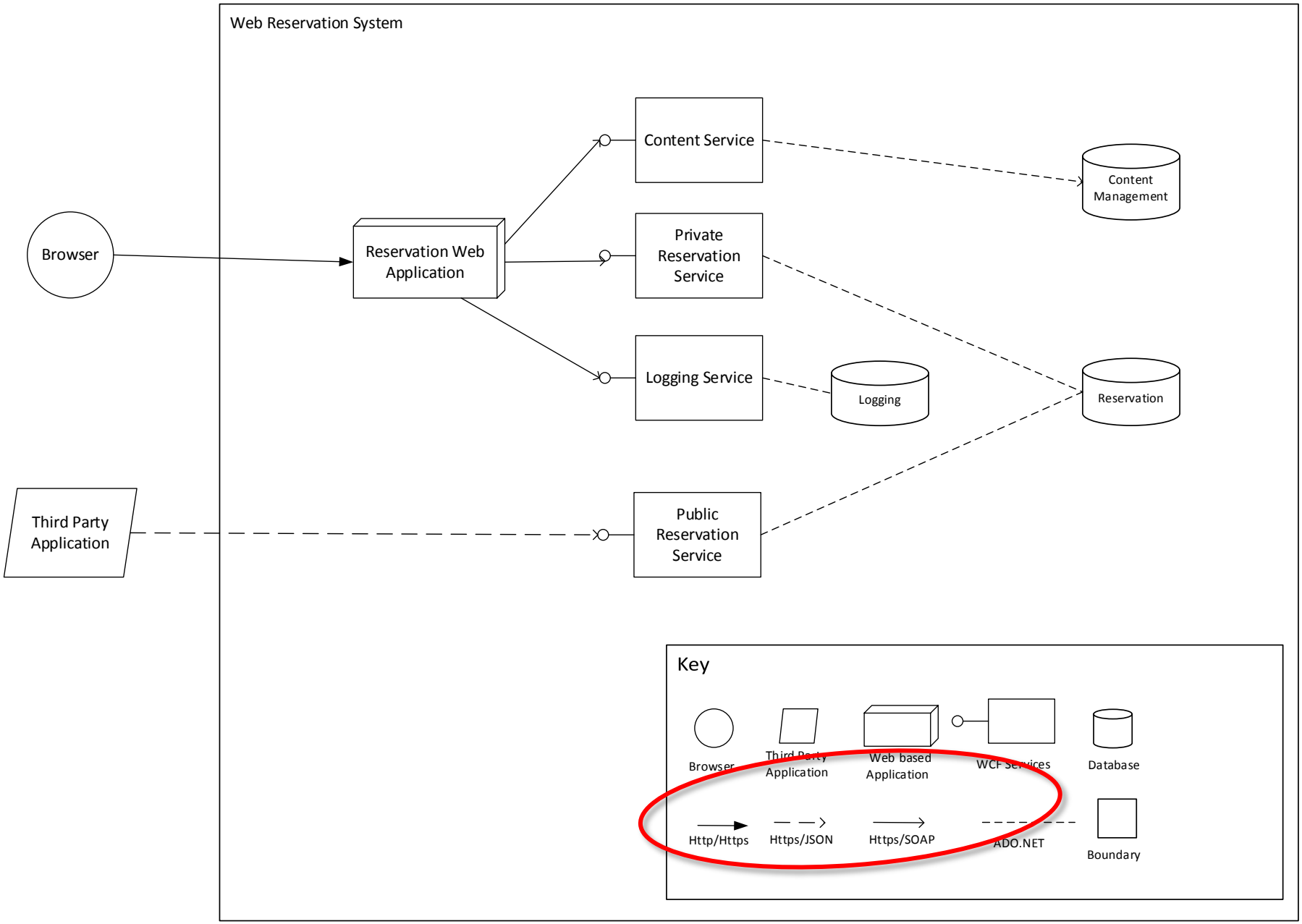
- Behavioral view of the architecture at run-time
- Components are execution units
- How the elements work together at run-time
- Performance
- Reliability
- Availability


Web Reservation System
Content Service
Content Management
Browser
Reservation Web Application
Private Reservation Service
Logging Service
Logging
Reservation
Third Party Application
Public Reservation Service
Key
Browser
Third Party Application
Web based Application
WCF Services
Database
Http/Https
Https/JSON
Https/SOAP
ADO.NET
Boundary

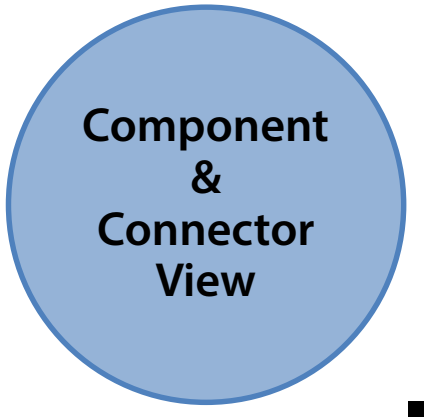

Web Reservation System
Browser
Reservation Web Application
Content Service
Private Reservation Service
Logging Service
Public Reservation Service
Third Party Application
Content Management
Logging
Reservation
Key
Browser
Third Party Application
Web based Application
WCF Services
Database
Http/Https
Https/JSON
Https/SOAP
ADO.NET
Boundary


Web Reservation System
Content Service
Content Management
Browser
Reservation Web Application
Private Reservation Service
Logging Service
Logging
Reservation
Third Party Application
Public Reservation Service
Key
Browser
Third Party Application
Web based Application
WCF Services
Database
Http/Https
Https/JSON
Https/SOAP
ADO.NET
Boundary


Web Reservation System
Content Service
Content Management
Browser
Reservation Web Application
Private Reservation Service
Logging Service
Logging
Reservation
Third Party Application
Public Reservation Service
Key
Browser
Third Party Application
Web based Application
WCF Services
Database
Http/Https
Https/JSON
Https/SOAP
ADO.NET
Boundary

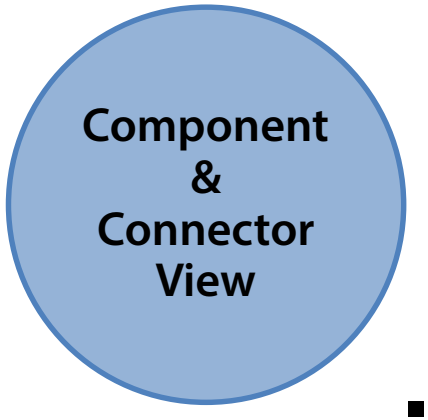

Web Reservation System
Content Service
Content Management
Browser
Reservation Web Application
Private Reservation Service
Logging Service
Logging
Reservation
Third Party Application
Public Reservation Service
Key
Browser
Third Party Application
Web based Application
WCF Services
Database
Http/Https
Https/JSON
Https/SOAP
ADO.NET
Boundary

Views & Beyond in practice



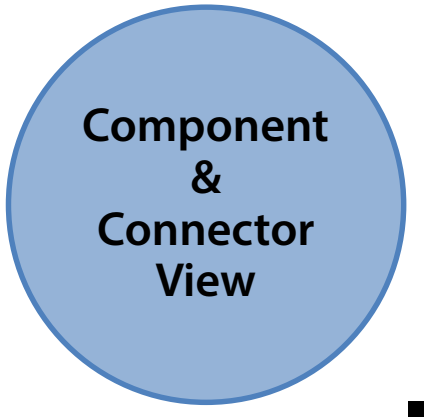
- **SOA**
  - Identify high level service related components
  - Interactions
  - Dependencies

Views & Beyond in practice

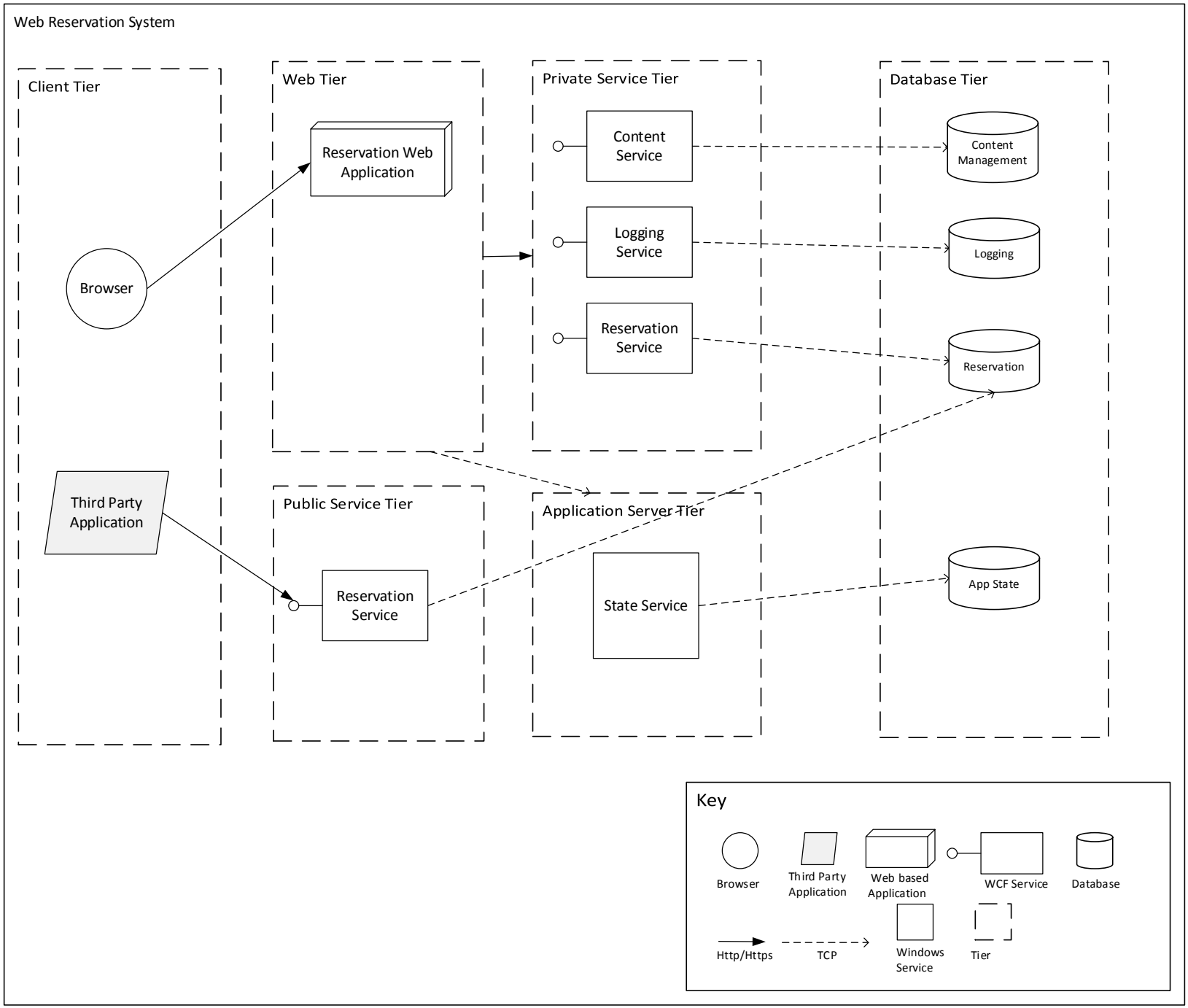


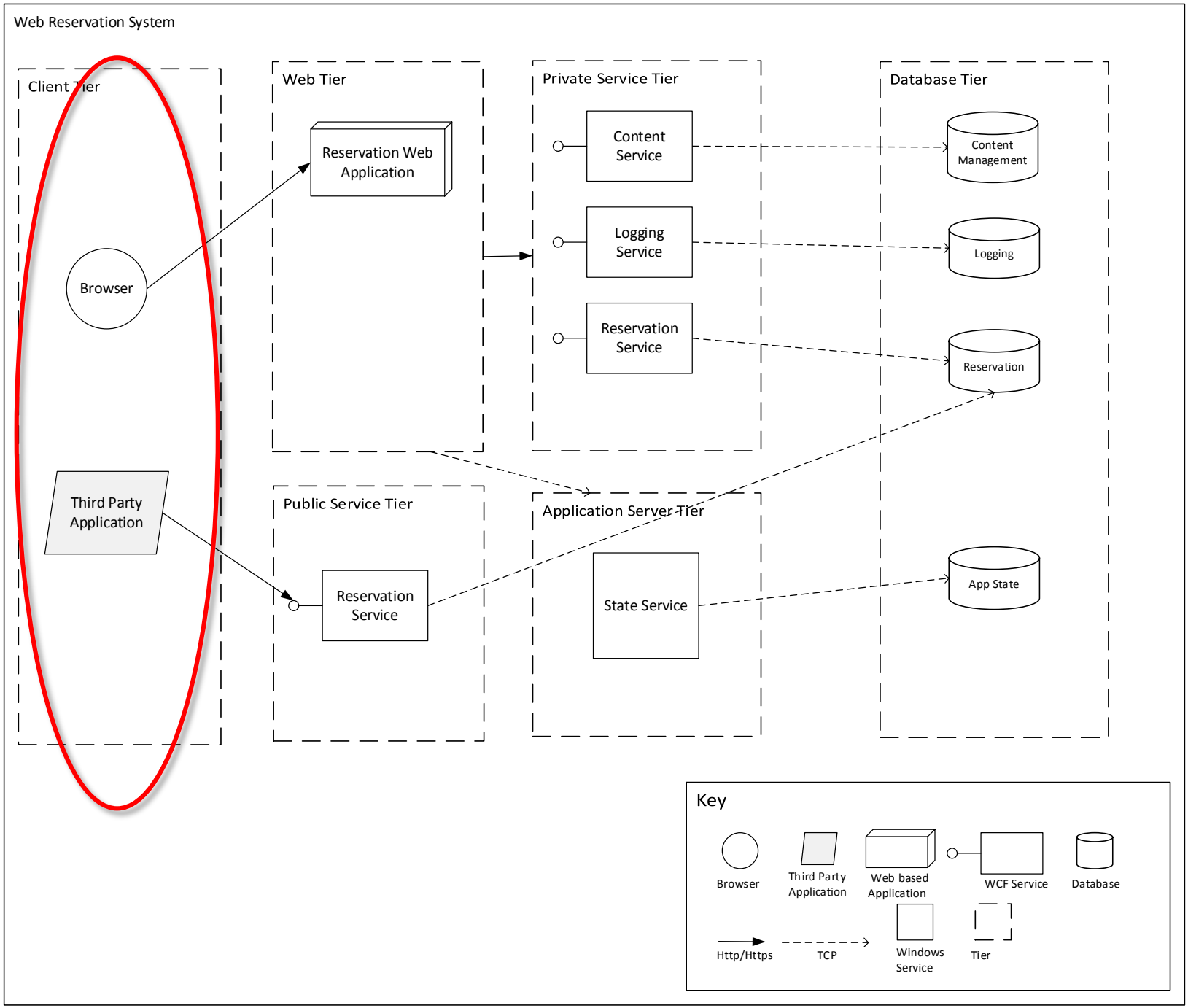
- **Tiered**
  - Depicts logical groupings of execution
  - Components grouped together logically by
    - Type
    - Execution requirement
    - Purpose
  - Identify communication requirements between components and tiers
  - Physical organization
  - Deployment model

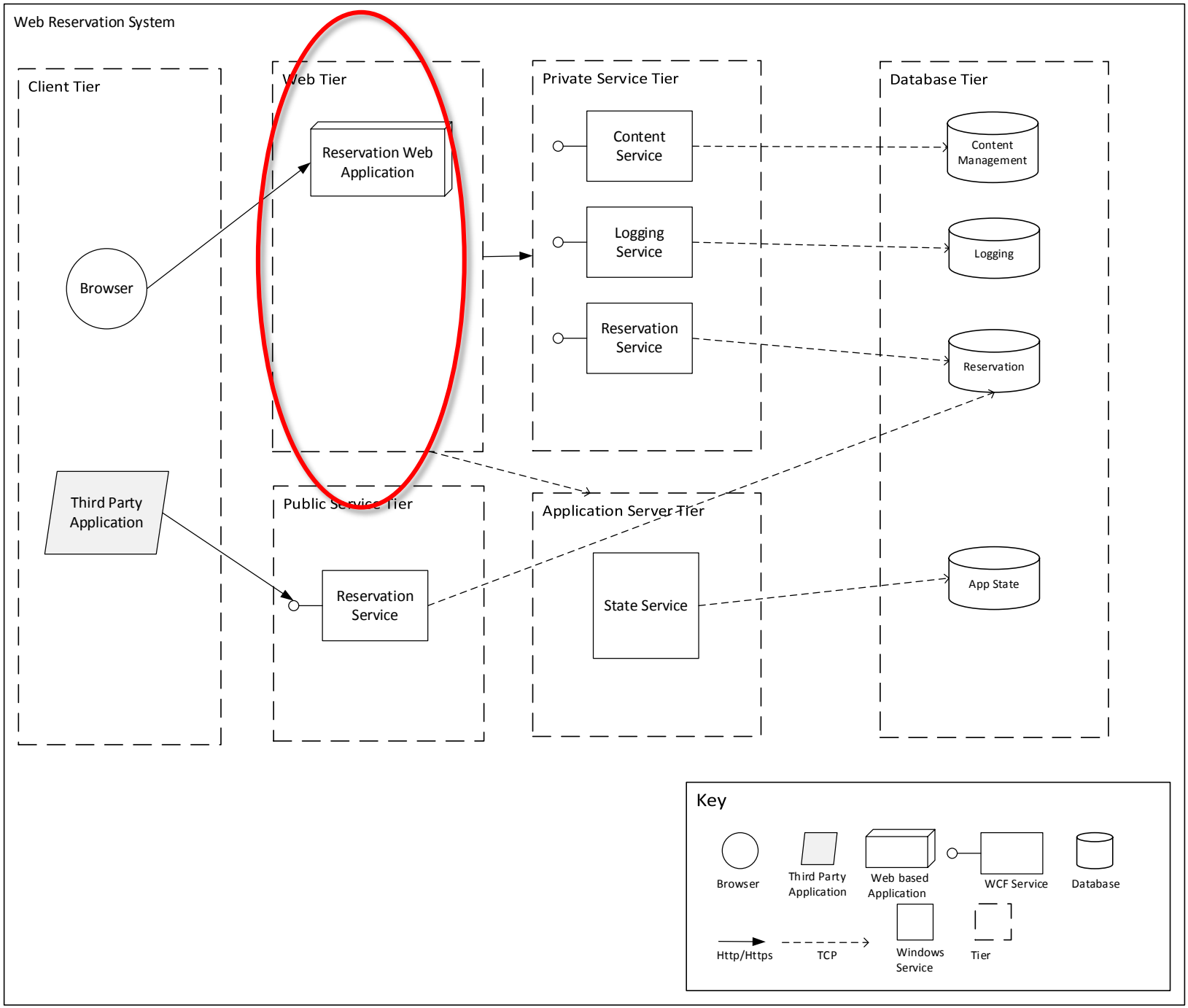
Views & Beyond in practice

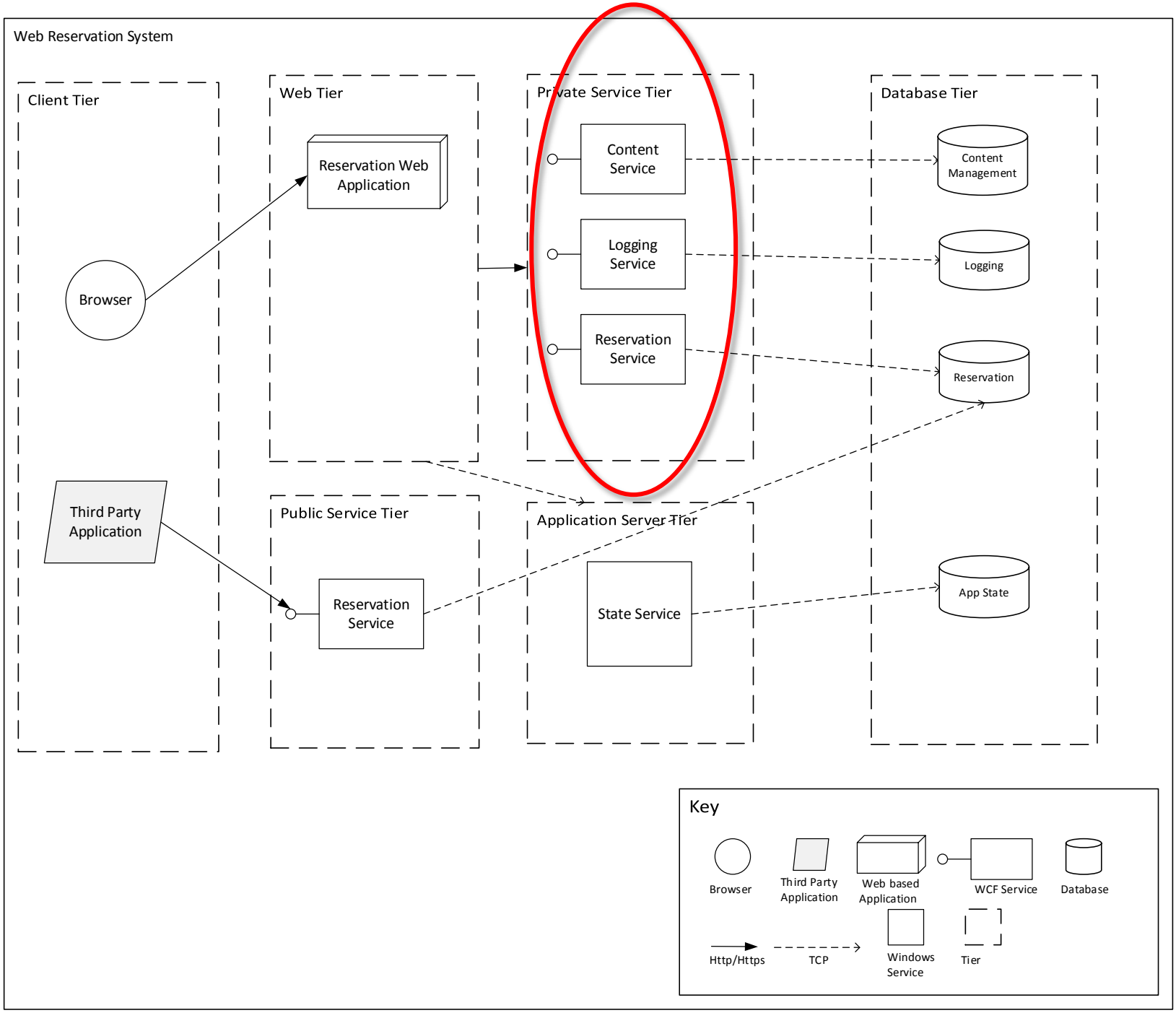


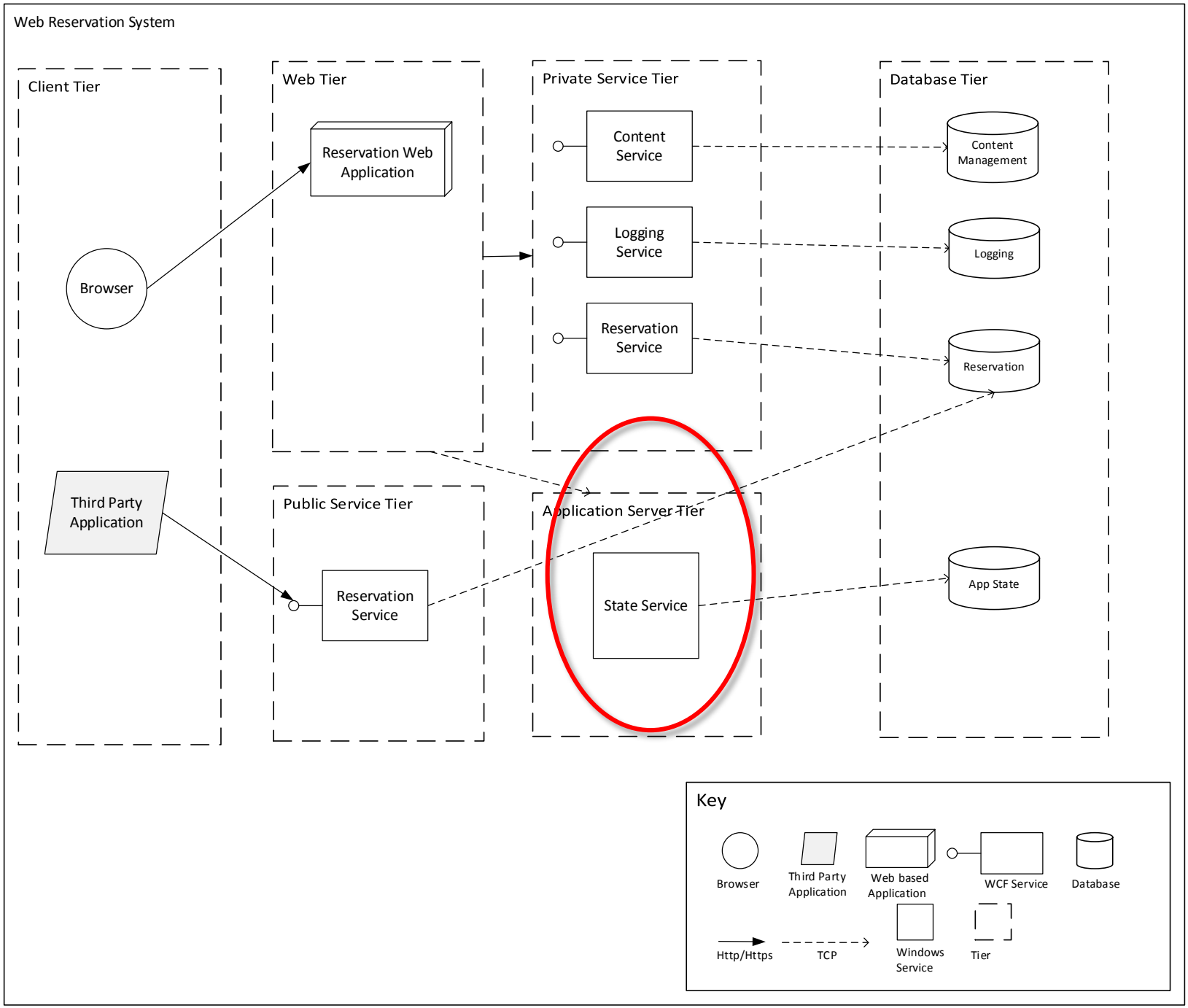
- **Tiered**
  - Understand deployment requirements
  - Deployment constraints are important
  - Identify infrastructure challenges
  - Not mapped to physical machines

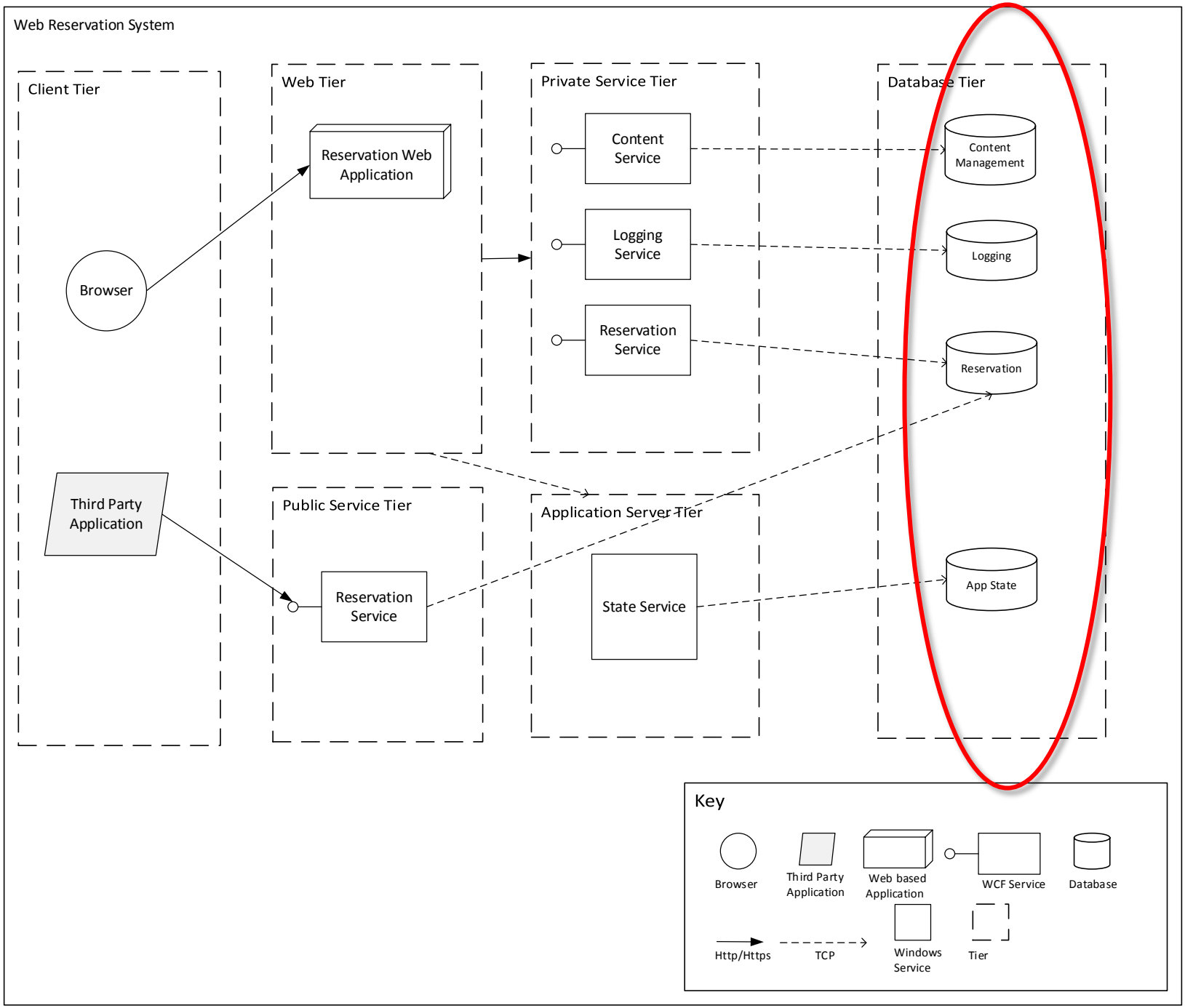

Web Reservation System
Client Tier
Web Tier
Private Service Tier
Database Tier
Reservation Web Application
Content Service
Content Management
Logging Service
Logging
Browser
Reservation Service
Reservation
Third Party Application
Public Service Tier
Application Server Tier
Reservation Service
State Service
App State
Key
Browser
Third Party Application
Web based Application
WCF Service
Database
Http/Https
TCP
Windows Service
Tier


Web Reservation System
Client Tier
Browser
Third Party Application
Web Tier
Reservation Web Application
Public Service Tier
Reservation Service
Private Service Tier
Content Service
Logging Service
Reservation Service
Application Server Tier
State Service
Database Tier
Content Management
Logging
Reservation
App State
Key
Browser
Third Party Application
Web based Application
WCF Service
Database
Http/Https
TCP
Windows Service
Tier

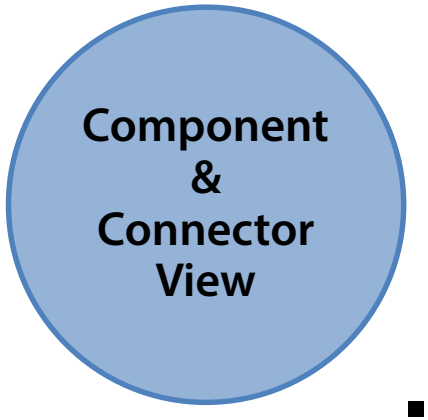

Web Reservation System
Client Tier
Web Tier
Private Service Tier
Database Tier
Reservation Web Application
Content Service
Content Management
Logging Service
Logging
Reservation Service
Reservation
Browser
Third Party Application
Public Service Tier
Application Server Tier
Reservation Service
State Service
App State
Key
Browser
Third Party Application
Web based Application
WCF Service
Database
Http/Https
TCP
Windows Service
Tier


Web Reservation System
Client Tier
Browser
Third Party Application
Web Tier
Reservation Web Application
Public Service Tier
Reservation Service
Private Service Tier
Content Service
Logging Service
Reservation Service
Application Server Tier
State Service
Database Tier
Content Management
Logging
Reservation
App State
Key
Browser
Third Party Application
Web based Application
WCF Service
Database
Http/Https
TCP
Windows Service
Tier


Web Reservation System
Client Tier
Web Tier
Private Service Tier
Database Tier
Reservation Web Application
Content Service
Content Management
Logging Service
Logging
Browser
Reservation Service
Reservation
Third Party Application
Public Service Tier
Application Server Tier
Reservation Service
State Service
App State
Key
Browser
Third Party Application
Web based Application
WCF Service
Database
Http/Https
TCP
Windows Service
Tier

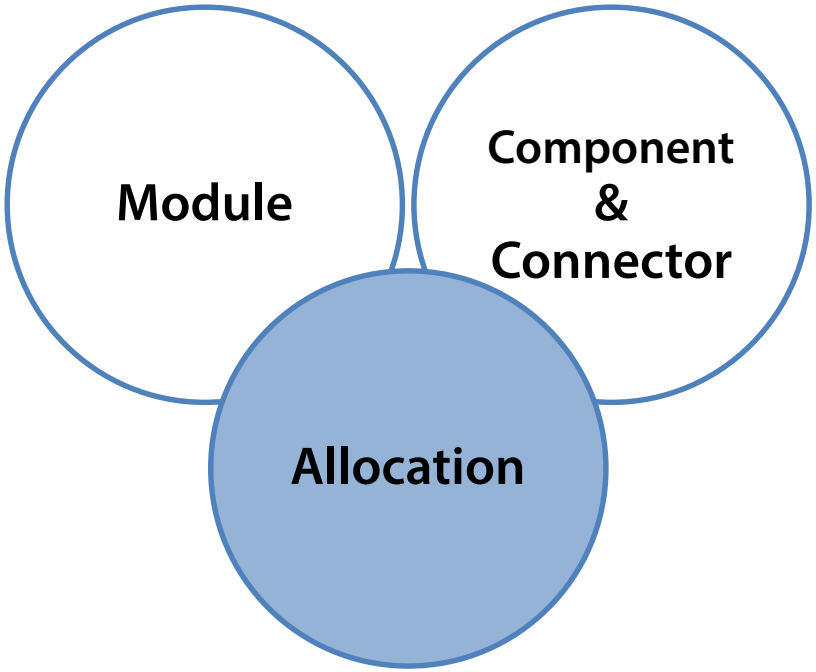

Web Reservation System
Client Tier
Browser
Third Party Application
Web Tier
Reservation Web Application
Public Service Tier
Reservation Service
Private Service Tier
Content Service
Logging Service
Reservation Service
Application Server Tier
State Service
Database Tier
Content Management
Logging
Reservation
App State
Key
Browser
Third Party Application
Web based Application
WCF Service
Database
Http/Https
TCP
Windows Service
Tier

Views & Beyond in practice



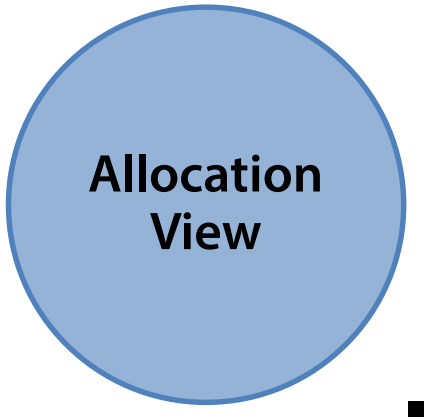
- Guided by your architectural style
- Visualize the behavior of these choices at run-time

Views & Beyond in practice



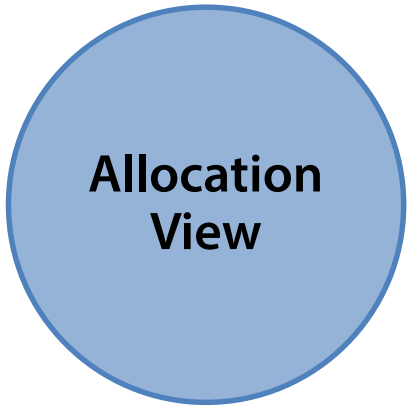
- **Identify the mapping between software elements & non-software elements**
  - System
  - Work team

Views & Beyond in practice



- **Work Assignment**
  - Simple lists
  - Subsystems or modules assignments

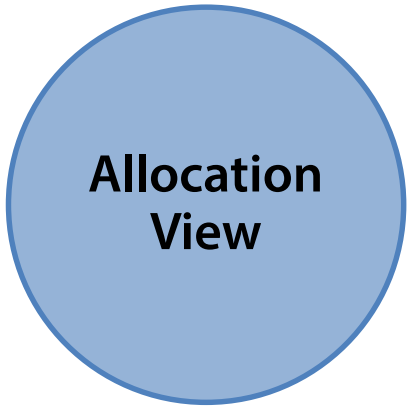
Views & Beyond in practice



Susbsytem	Module	Resource
Reservation Workflow Manager	Workflow Manager	Developer 1
	Availibility	Developer 2
	Reservations	Developer 1
	Payments	Developer 2
Reservation Services	Availibility	Developer 3
	Reservations	Developer 4
	Payments	Developer 3

- **Work Assignment**
  - Simple lists
  - Subsystems or modules assignments

Views & Beyond in practice



Susbsytem	Module	Resource
Reservation Workflow Manager	Workflow Manager	Developer 1
	Availibility	Developer 2
	Reservations	Developer 1
	Payments	Developer 2
Reservation Services	Availibility	Developer 3
	Reservations	Developer 4
	Payments	Developer 3

- **Work Assignment**
  - Simple lists
  - Subsystems or modules assignments

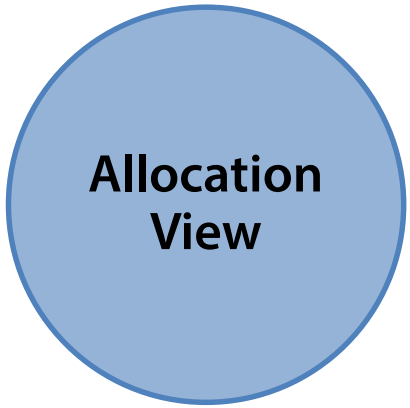
Views & Beyond in practice

Allocation View

Susbsytem	Module	Resource
Reservation Workflow Manager	Workflow Manager	Developer 1
	Availibility	Developer 2
	Reservations	Developer 1
	Payments	Developer 2
Reservation Services	Availibility	Developer 3
	Reservations	Developer 4
	Payments	Developer 3

- **Work Assignment**
  - Simple lists
  - Subsystems or modules assignments

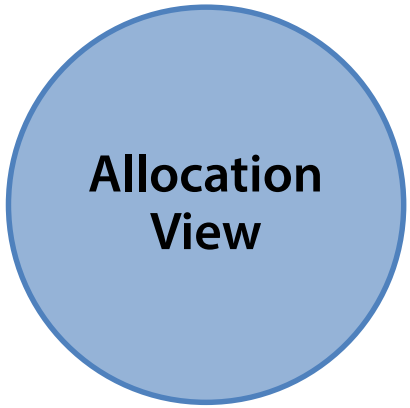
Views & Beyond in practice



Susbsytem	Module	Resource
Reservation Workflow Manager	Workflow Manager	Developer 1
	Availibility	Developer 2
	Reservations	Developer 1
	Payments	Developer 2
Reservation Services	Availibility	Developer 3
	Reservations	Developer 4
	Payments	Developer 3

- **Work Assignment**
  - Simple lists
  - Subsystems or modules assignments

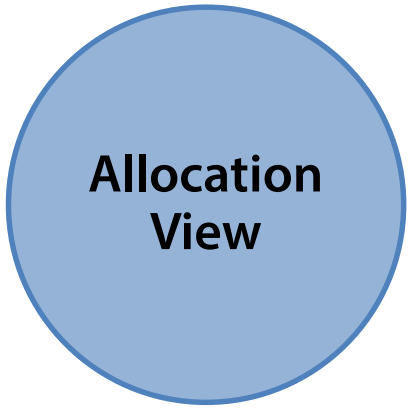
Views & Beyond in practice



Susbsytem	Module	Resource
Reservation Workflow Manager	Workflow Manager	Developer 1
	Availibility	Developer 2
	Reservations	Developer 1
	Payments	Developer 2
Reservation Services	Availibility	Developer 3
	Reservations	Developer 4
	Payments	Developer 3

- **Work Assignment**
  - Simple lists
  - Subsystems or modules assignments

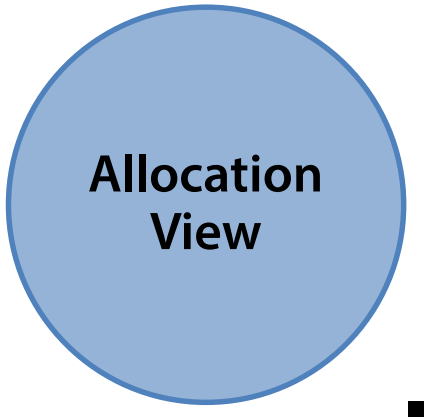
Views & Beyond in practice



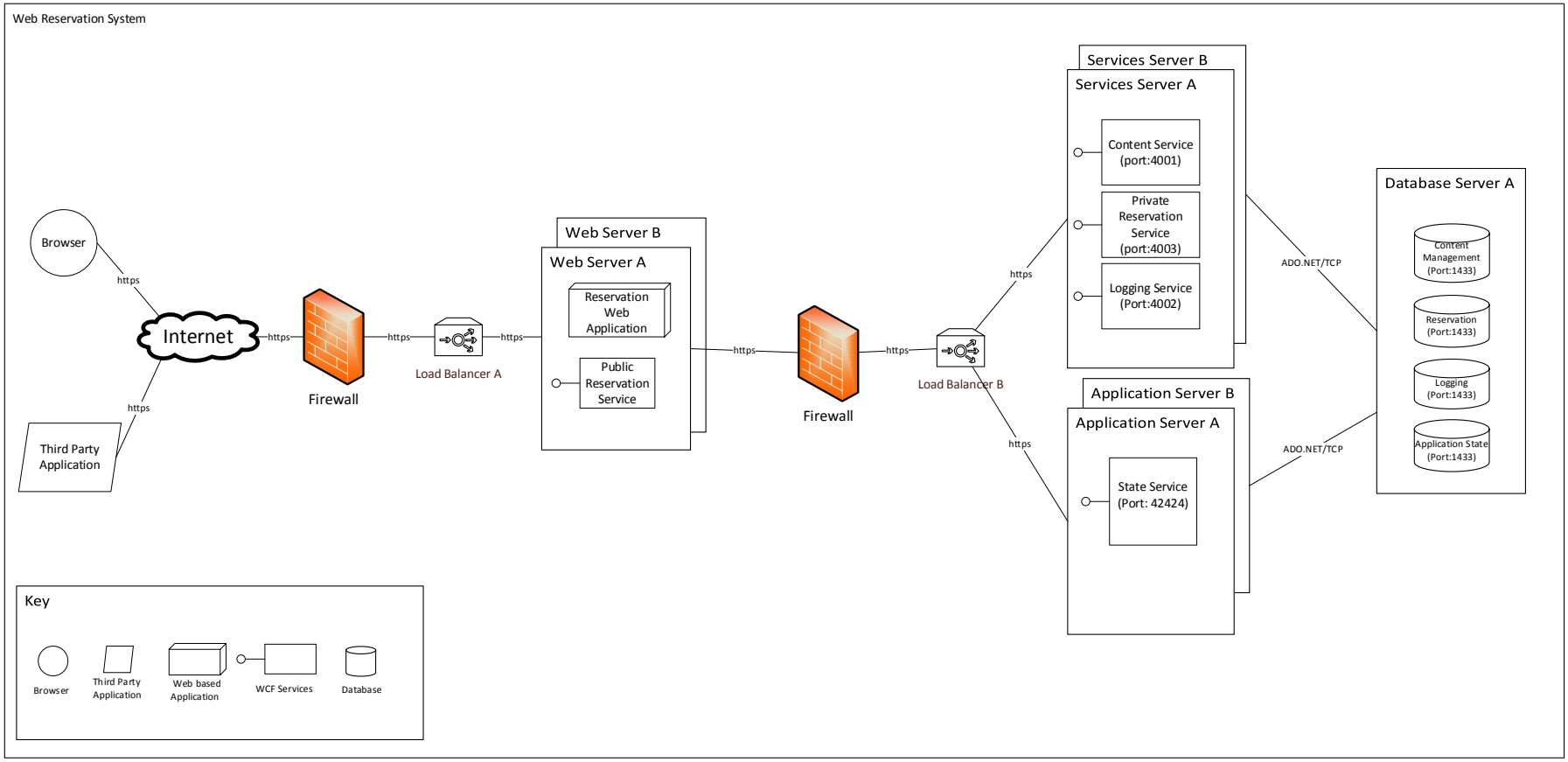
Susbsytem	Module	Resource
Reservation Workflow Manager	Workflow Manager	Developer 1
	Availibility	Developer 2
	Reservations	Developer 1
	Payments	Developer 2
Reservation Services	Availibility	Developer 3
	Reservations	Developer 4
	Payments	Developer 3

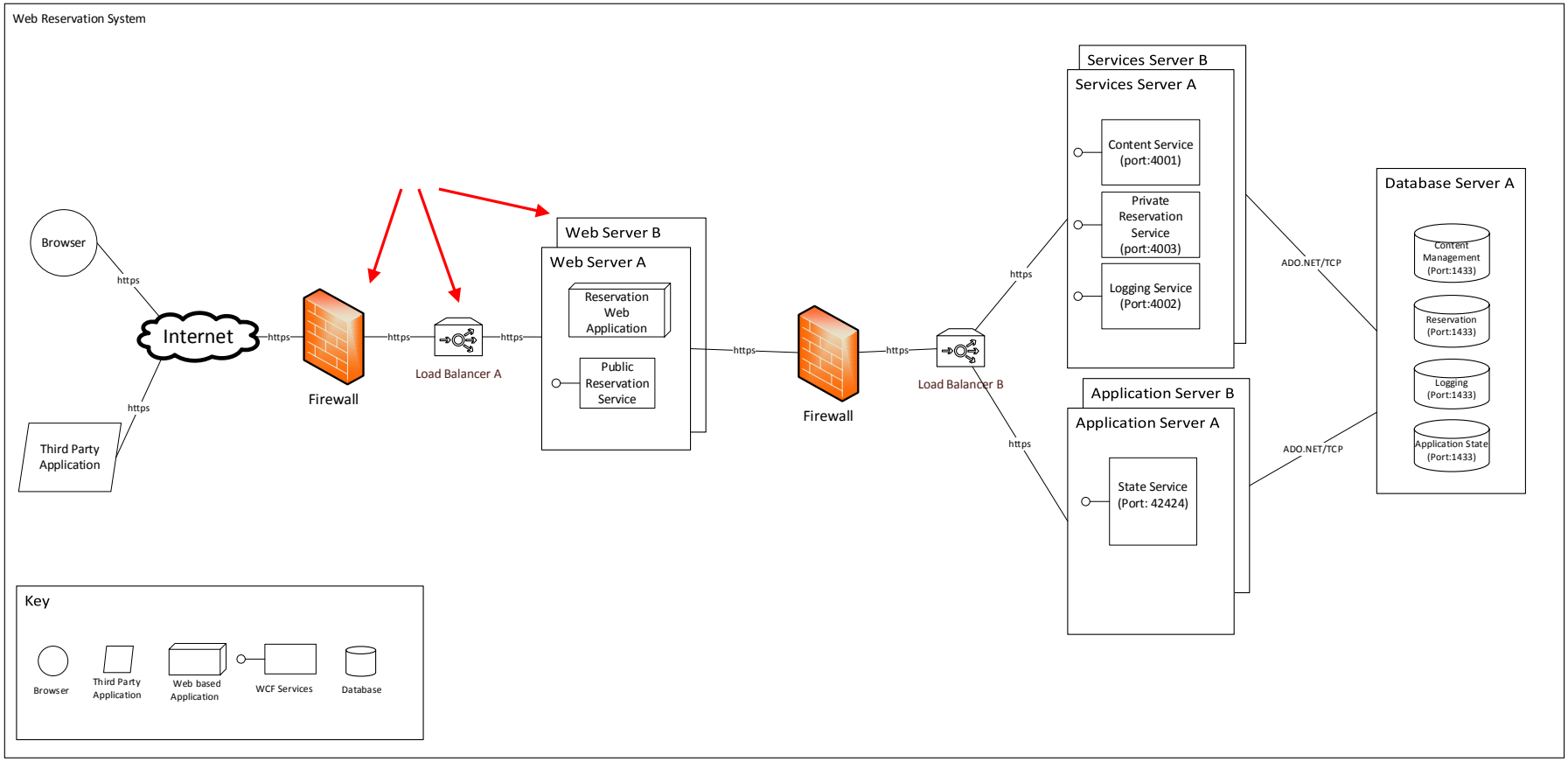
- **Work Assignment**
  - Simple lists
  - Subsystems or modules assignments
  - Resource assignments may be performed as part of your sprint planning sessions

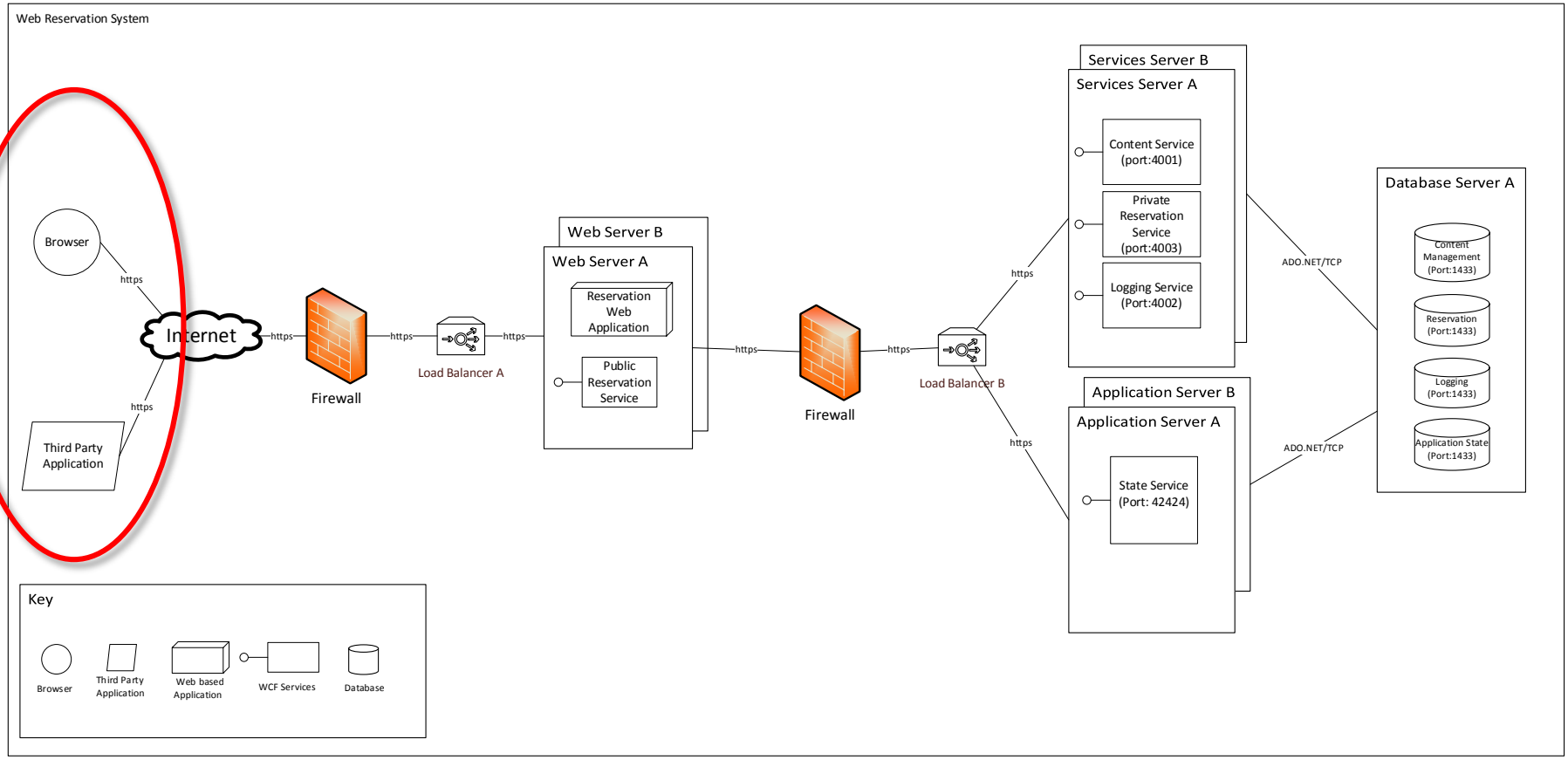
Views & Beyond in practice

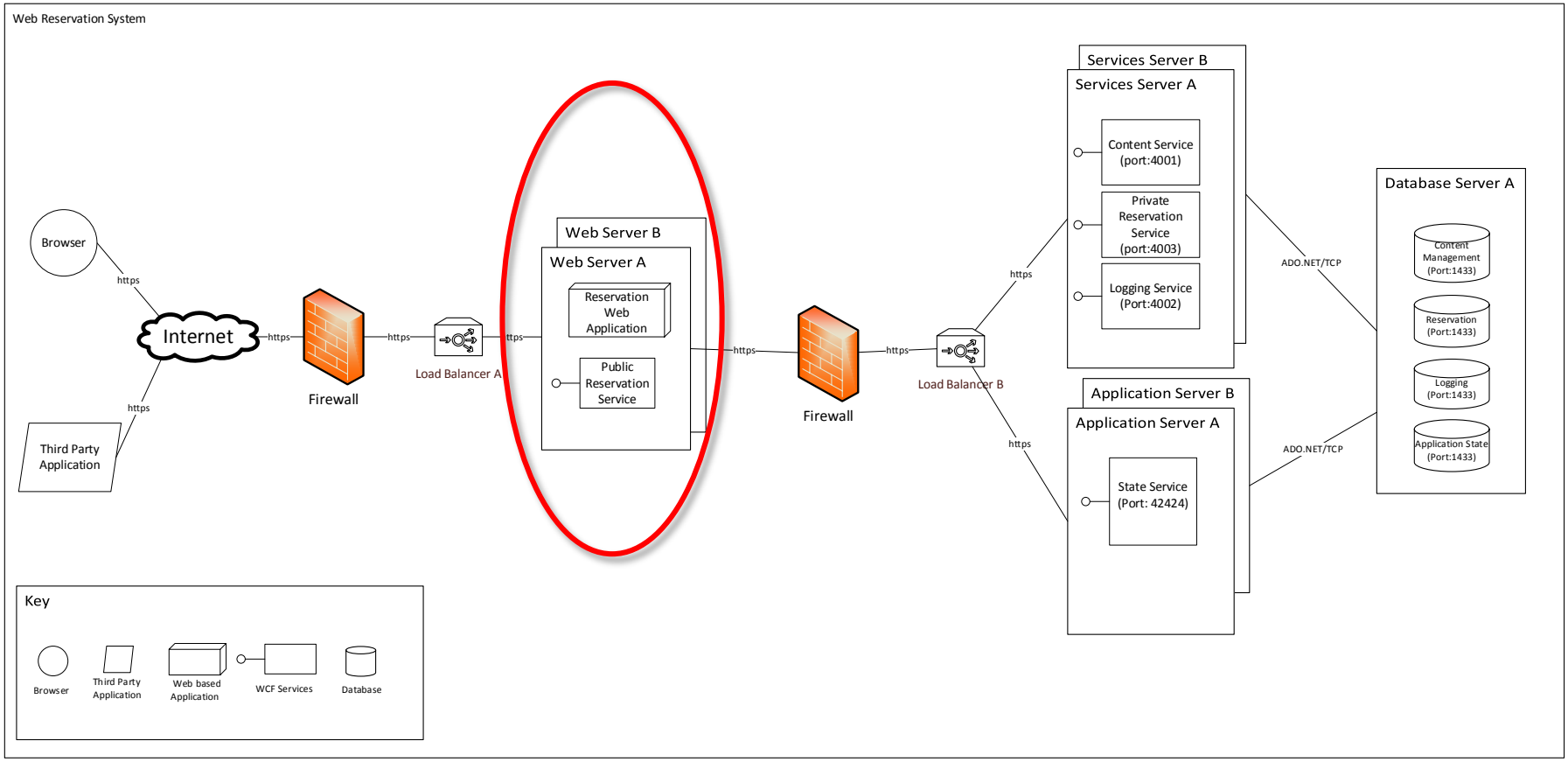


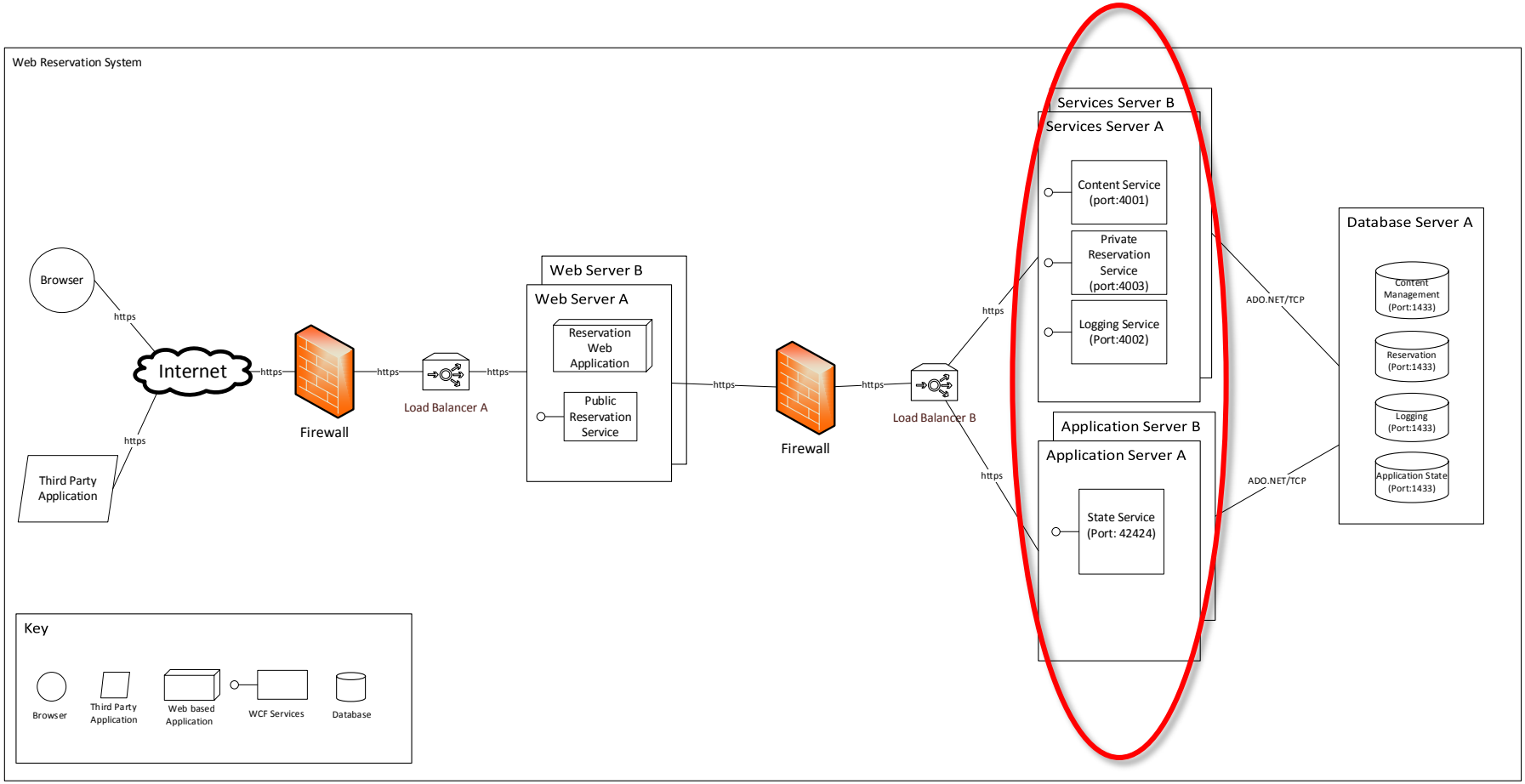
- **Deployment**
  - Map components to physical machines
  - Performance
  - Availability
  - Reliability
  - Security
  - Use generic like Server A and B

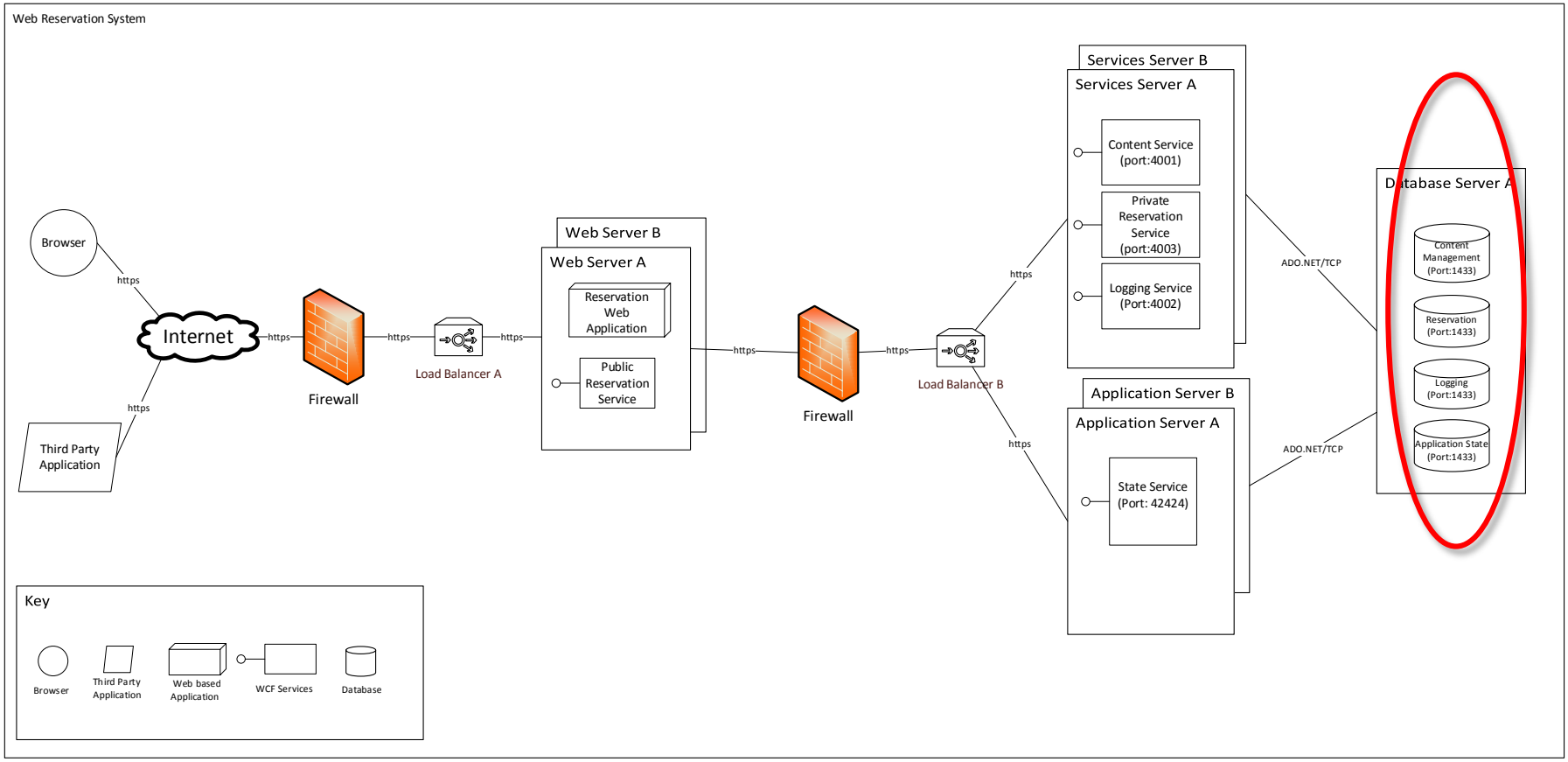

Web Reservation System
Browser
Third Party Application
Internet
https
Firewall
Load Balancer A
Web Server B
Web Server A
Reservation Web Application
Public Reservation Service
Firewall
Load Balancer B
Services Server B
Services Server A
Content Service (port:4001)
Private Reservation Service (port:4003)
Logging Service (Port:4002)
Application Server B
Application Server A
State Service (Port: 42424)
ADO.NET/TCP
Database Server A
Content Management (Port:1433)
Reservation (Port:1433)
Logging (Port:1433)
Application State (Port:1433)
Key
Browser
Third Party Application
Web based Application
WCF Services
Database

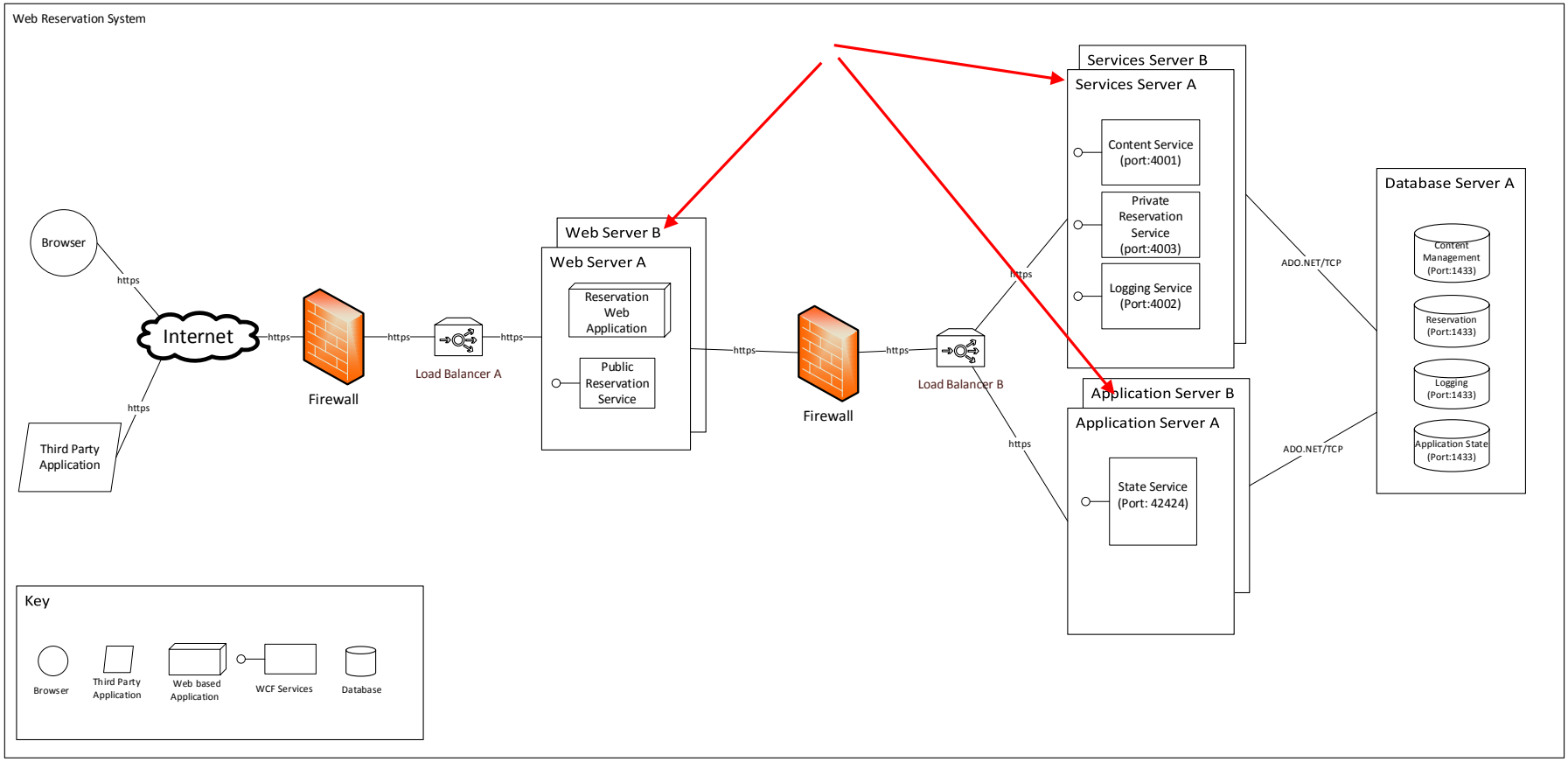

Web Reservation System
Browser
https
Internet
https
Third Party Application
https
Firewall
https
Load Balancer A
https
Web Server B
Web Server A
Reservation Web Application
Public Reservation Service
https
Firewall
https
Load Balancer B
https
https
Services Server B
Services Server A
Content Service (port:4001)
Private Reservation Service (port:4003)
Logging Service (Port:4002)
Application Server B
Application Server A
State Service (Port: 42424)
ADO.NET/TCP
ADO.NET/TCP
Database Server A
Content Management (Port:1433)
Reservation (Port:1433)
Logging (Port:1433)
Application State (Port:1433)
Key
Browser
Third Party Application
Web based Application
WCF Services
Database

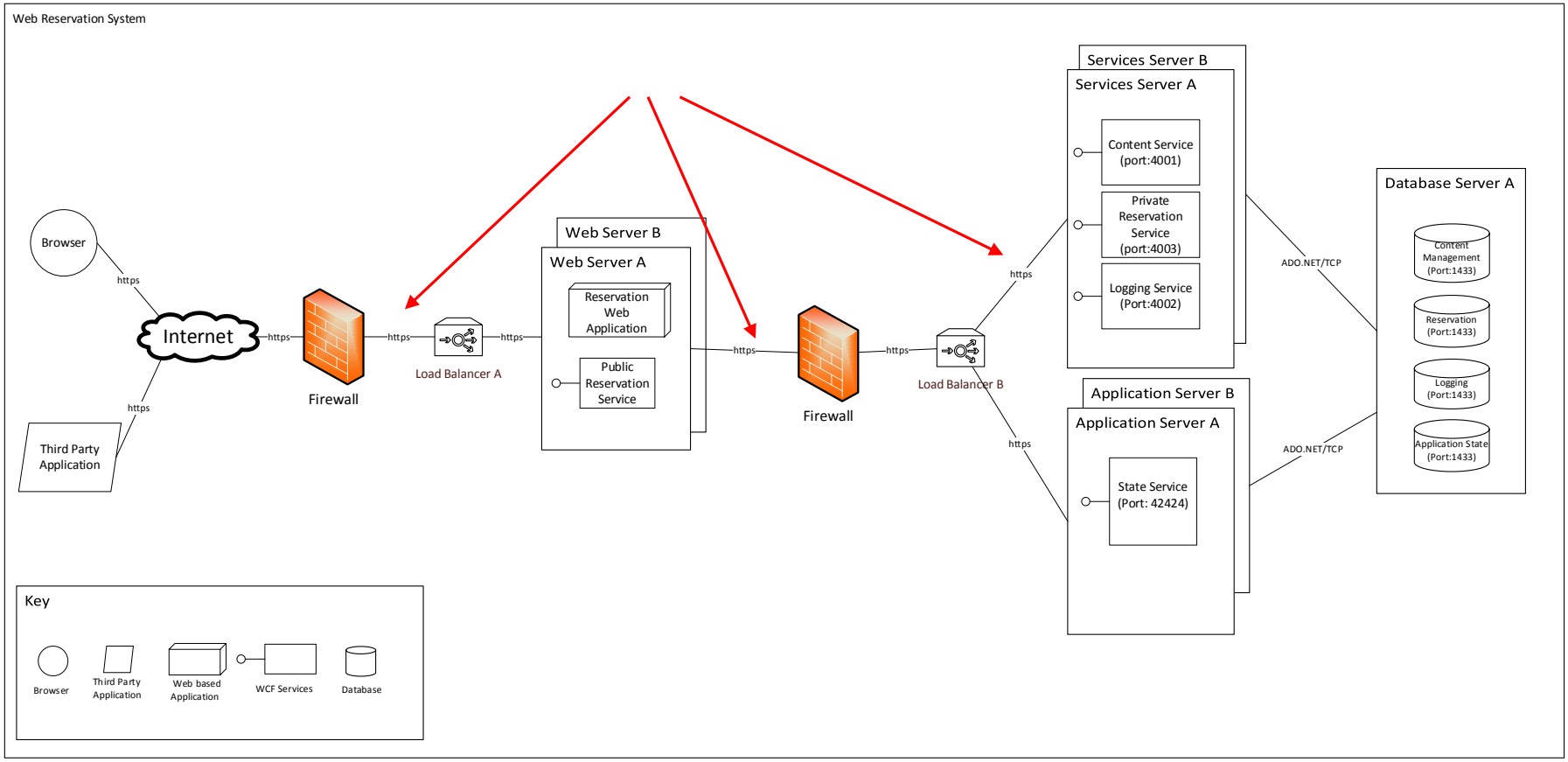

Web Reservation System
Browser
https
Internet
https
Third Party Application
https
Firewall
https
Load Balancer A
https
Web Server B
Web Server A
Reservation Web Application
Public Reservation Service
https
Firewall
https
Load Balancer B
https
https
Services Server B
Services Server A
Content Service (port:4001)
Private Reservation Service (port:4003)
Logging Service (Port:4002)
Application Server B
Application Server A
State Service (Port: 42424)
ADO.NET/TCP
ADO.NET/TCP
Database Server A
Content Management (Port:1433)
Reservation (Port:1433)
Logging (Port:1433)
Application State (Port:1433)
Key
Browser
Third Party Application
Web based Application
WCF Services
Database

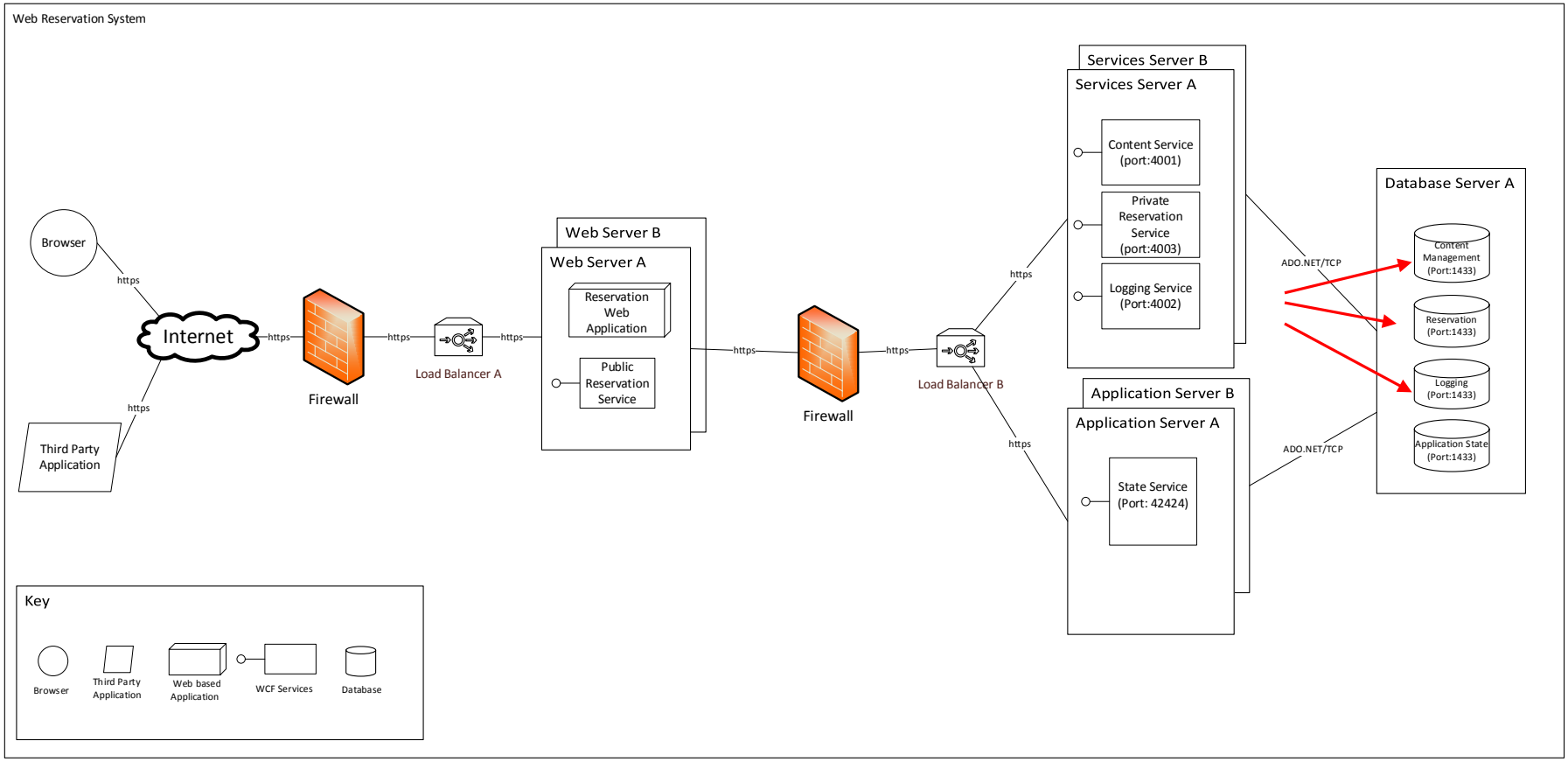

Web Reservation System
Browser
Third Party Application
Internet
https
Firewall
Load Balancer A
Web Server B
Web Server A
Reservation Web Application
Public Reservation Service
Firewall
Load Balancer B
Services Server B
Services Server A
Content Service (port:4001)
Private Reservation Service (port:4003)
Logging Service (Port:4002)
Application Server B
Application Server A
State Service (Port: 42424)
ADO.NET/TCP
Database Server A
Content Management (Port:1433)
Reservation (Port:1433)
Logging (Port:1433)
Application State (Port:1433)
Key
Browser
Third Party Application
Web based Application
WCF Services
Database


Web Reservation System
Browser
Third Party Application
https
Internet
Firewall
Load Balancer A
Web Server B
Web Server A
Reservation Web Application
Public Reservation Service
Firewall
Load Balancer B
Services Server B
Services Server A
Content Service (port:4001)
Private Reservation Service (port:4003)
Logging Service (Port:4002)
Application Server B
Application Server A
State Service (Port: 42424)
ADO.NET/TCP
Database Server A
Content Management (Port:1433)
Reservation (Port:1433)
Logging (Port:1433)
Application State (Port:1433)
Key
Browser
Third Party Application
Web based Application
WCF Services
Database

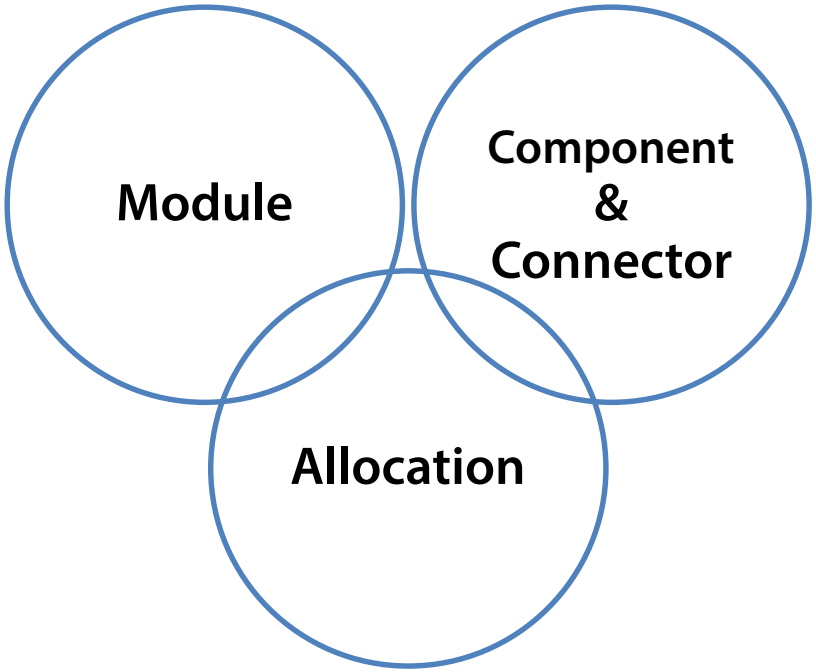

Web Reservation System
Browser
https
Internet
https
Third Party Application
https
Firewall
https
Load Balancer A
https
Web Server B
Web Server A
Reservation Web Application
Public Reservation Service
https
Firewall
https
Load Balancer B
https
https
Services Server B
Services Server A
Content Service (port:4001)
Private Reservation Service (port:4003)
Logging Service (Port:4002)
ADO.NET/TCP
Application Server B
Application Server A
State Service (Port: 42424)
ADO.NET/TCP
Database Server A
Content Management (Port:1433)
Reservation (Port:1433)
Logging (Port:1433)
Application State (Port:1433)
Key
Browser
Third Party Application
Web based Application
WCF Services
Database


Web Reservation System
Browser
https
Internet
https
Third Party Application
https
Firewall
https
Load Balancer A
https
Web Server B
Web Server A
Reservation Web Application
Public Reservation Service
https
Firewall
https
Load Balancer B
https
https
Services Server B
Services Server A
Content Service (port:4001)
Private Reservation Service (port:4003)
Logging Service (Port:4002)
Application Server B
Application Server A
State Service (Port: 42424)
ADO.NET/TCP
ADO.NET/TCP
Database Server A
Content Management (Port:1433)
Reservation (Port:1433)
Logging (Port:1433)
Application State (Port:1433)
Key
Browser
Third Party Application
Web based Application
WCF Services
Database


Web Reservation System
Browser
https
Internet
https
Third Party Application
https
Firewall
https
Load Balancer A
https
Web Server B
Web Server A
Reservation Web Application
Public Reservation Service
https
Firewall
https
Load Balancer B
https
https
Services Server B
Services Server A
Content Service (port:4001)
Private Reservation Service (port:4003)
Logging Service (Port:4002)
Application Server B
Application Server A
State Service (Port: 42424)
ADO.NET/TCP
ADO.NET/TCP
Database Server A
Content Management (Port:1433)
Reservation (Port:1433)
Logging (Port:1433)
Application State (Port:1433)
Key
Browser
Third Party Application
Web based Application
WCF Services
Database

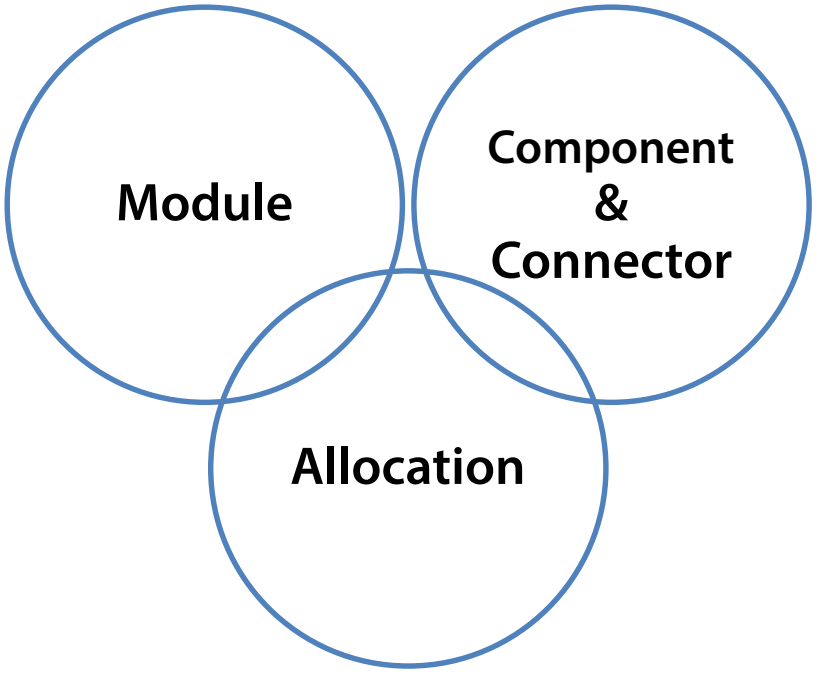

Web Reservation System
Browser
Third Party Application
Internet
https
Firewall
Load Balancer A
Web Server B
Web Server A
Reservation Web Application
Public Reservation Service
Firewall
Load Balancer B
Services Server B
Services Server A
Content Service (port:4001)
Private Reservation Service (port:4003)
Logging Service (Port:4002)
Application Server B
Application Server A
State Service (Port: 42424)
ADO.NET/TCP
Database Server A
Content Management (Port:1433)
Reservation (Port:1433)
Logging (Port:1433)
Application State (Port:1433)
Key
Browser
Third Party Application
Web based Application
WCF Services
Database

Views & Beyond in practice



- Diagram is the primary presentation
- Element catalog
- Variability guide
- Rationale

Views & Beyond in practice



- It isn't important to document everything all at once
- Ok to just say TBD
- Iterative and evolutionary
- Text offers many samples of UML & informal notations
- Elements, behaviors and relationships are the same

Summary

- **Design**
  - Goals of software architecture & detailed design
  - How they are different
  - Who should perform them
  - Role of architecture in an agile world
  - Two fundamental approaches
    - Top-down
    - Bottom-up
- **Architectural Design Process**
  - Design considerations
  - Design process step by step
  - Prototypes
  - Architectural patterns
  - When to use them

Summary

- **Communicating the Solution**
  - 3 main objectives
  - Documentation standards
- **Views**
  - 4+1 architectural view model
  - Views and beyond
- **Views & Beyond applied**
  - Module
  - Component-and-Connector
  - Allocation

Summary

- Practical as possible
- Many real world examples
- Starting point
- Continue your education
- No matter how experienced you are there are always still more things to learn
- Software Architecture in Practice (3rd Edition)
- Documenting Software Architectures: Views and Beyond (2nd Edition)

Developer to Architect

The Software Architect Role in the Enterprise

Chris Simmons
www.avidsoftware.com
chris.simmons@avidsoftware.com

pluralsight
hardcore developer training