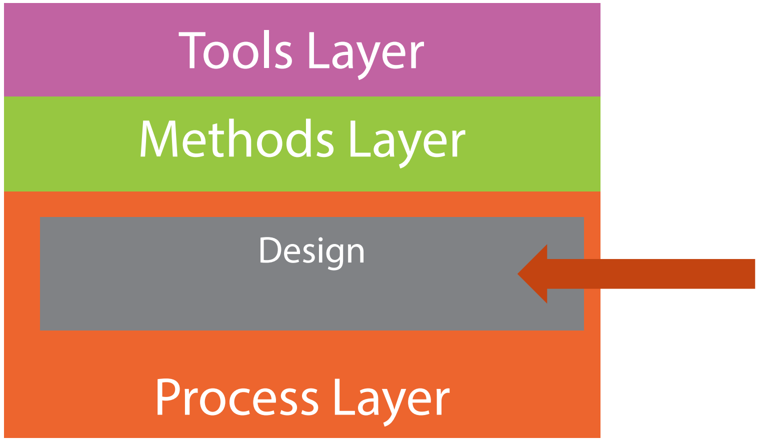
Design – Essentials

Mohamad Halabi
@mohamadhalabi

pluralsight
hardcore dev and IT training

Introduction



Analysis Activity vs. Design Activity

- **It’s a matter of focus:**

Analysis	Design
Understand the problem	Understand the solution
The “what” aspect	The “how” aspect
Translating functional requirements into software concerns	Functional requirements are translated into software components design
Non-functional requirements are listed quantitatively	Non-functional requirements are the core focus
Implementation-independent	Done at different abstraction levels (architecture, design, technology)

From Requirements Modeling to Design

- **Recall that analysis models guide the design phase:**
  - Use Case models illustrate what are the system functionalities
  - Class diagrams show what are major conceptual classes and relationships required to fulfill system functionalities
  - Interaction diagrams show what are the needed interactions between the conceptual classes
  - State Machine diagrams show what are the events that change the states of the conceptual classes
  - Non-functional requirements are clearly articulated

From Requirements Modeling to Design

- **In design, analysis models help us answer the following:**
  - How are objects grouped into system-level components?
  - How are these components structured?
  - How are the component interfaces designed?
  - How do these components interact?
  - How do these components satisfy the non-functional requirements?
  - How are the inner classes of these components designed?
  - How are the inner classes related?
  - How do the inner classes interact?
  - How events and states affect these inner classes?

Two Levels of Design

1. High-level design (architectural design or simply architecture)
2. Low-level design (detailed design or simply design)

- So what is architecture?

Architecture

- Software Engineering Institute (SEI): “*the structure or structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships among them*”
- IEEE: “*the fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution*”

Common Properties of Architecture

- **Focuses on architecture-relevant components**
  - A component encapsulates other elements (data, classes, procedures, logic, etc…) and exposes an interface which defines its behavior

- **Architecture-relevant components:**
  - Affect the overall structure and behavior
  - Influence quality attributes (ex: performance, security, scalability, etc…)
  - Influence environmental and technological constraints (i.e. non-functional requirements)

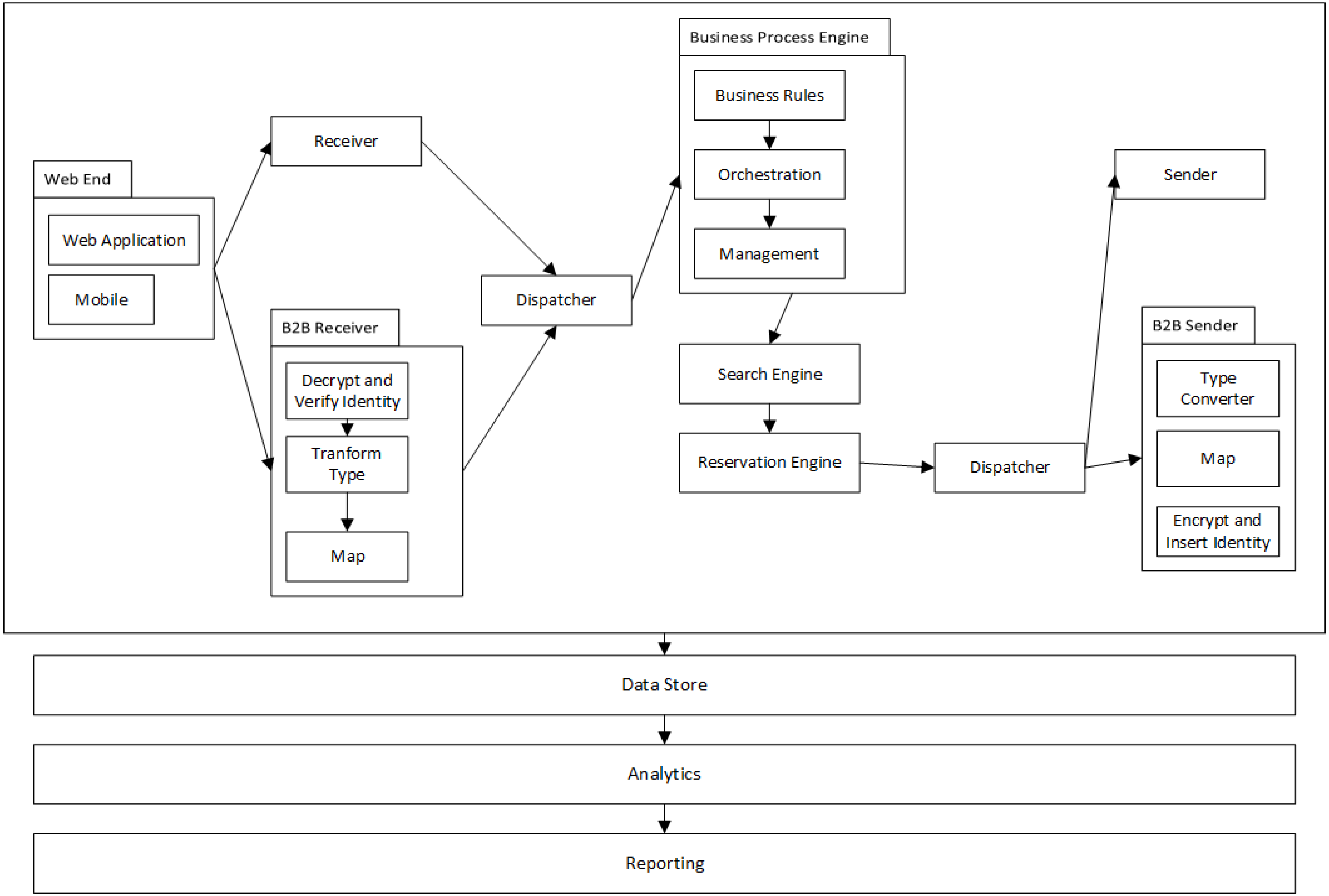
Common Properties of Architecture

- **Defines the system structure in terms of architecture-relevant components**
- **Defines the system behavior in terms of architecture-relevant components interactions**
- **Tackles most of the non-functional requirements**
  - Process and tools constraints can be tackled later
  - Quality attributes and environmental constraints are tackled via the architecture
- **Might conform to architectural styles**
  - Solutions to system-level organization problems
  - Provides predefined component types, responsibilities, and relationships
  - Ex: Pipe-and-Filter and Publish and Subscribe

What About Design?

- **Deals with lower abstraction layer**
  - Works with the constituents of the architecture-relevant components or non architecture-relevant components
  - Concerned with classes
  - Concerned with components that do not influence the overall structure and behavior
- **Defines the structure and behavior of the constituents and components**
- **Might conform to design patterns**
  - Solutions to detailed design problems
  - Ex: Singleton, Factory, Decorator, etc…

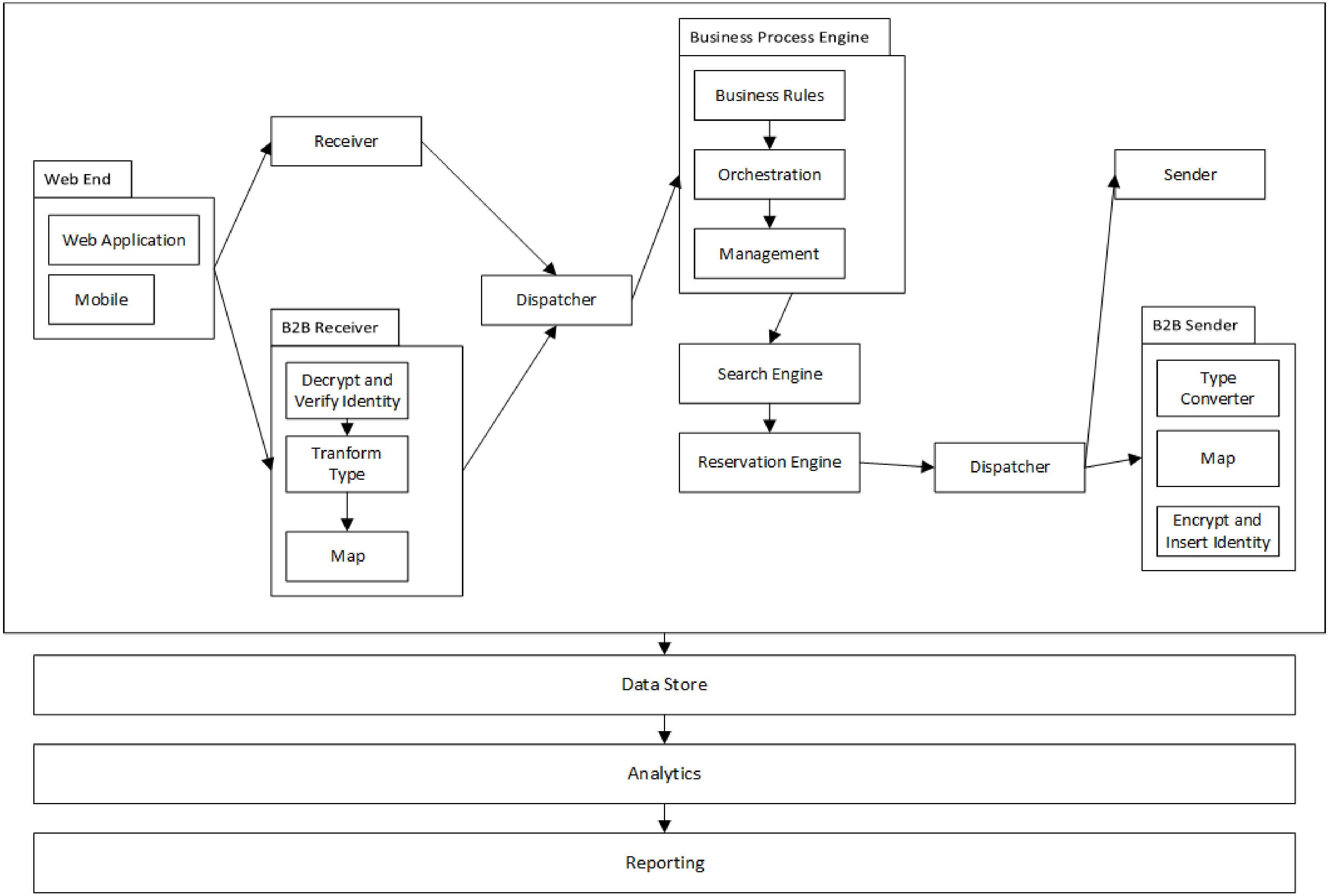
Logical Architecture Model: Structure



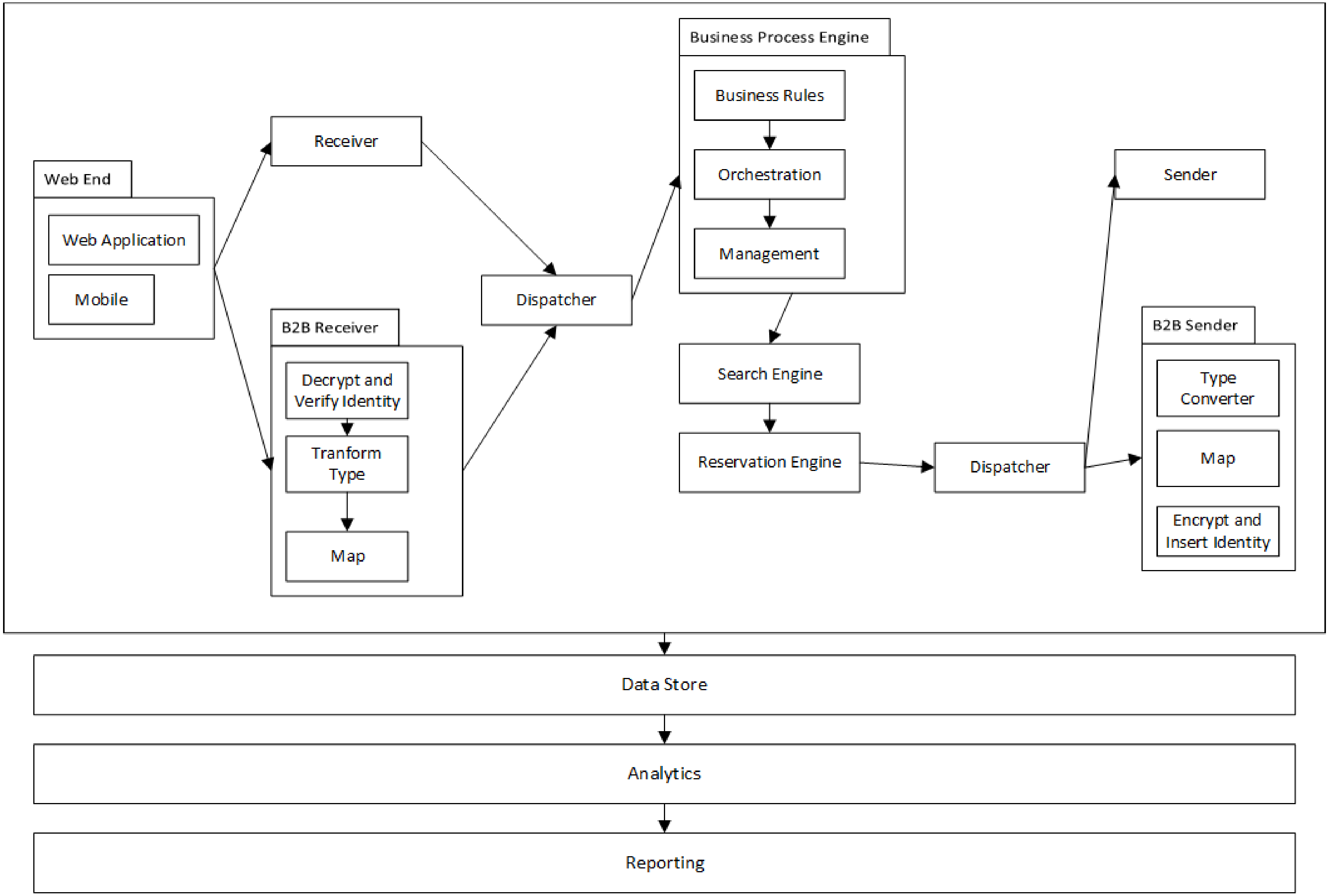
Architecture-Relevant Components

- **Architecture can be decomposed into multiple levels of architecture-relevant components**
  - The number of levels depends on architecture complexity
- **Architects reply on experience to identify components**
- **Recall that an architecture-relevant component:**
  - Affects the overall structure and behavior of the system
  - Achieves the required level of quality attributes and relevant constraints

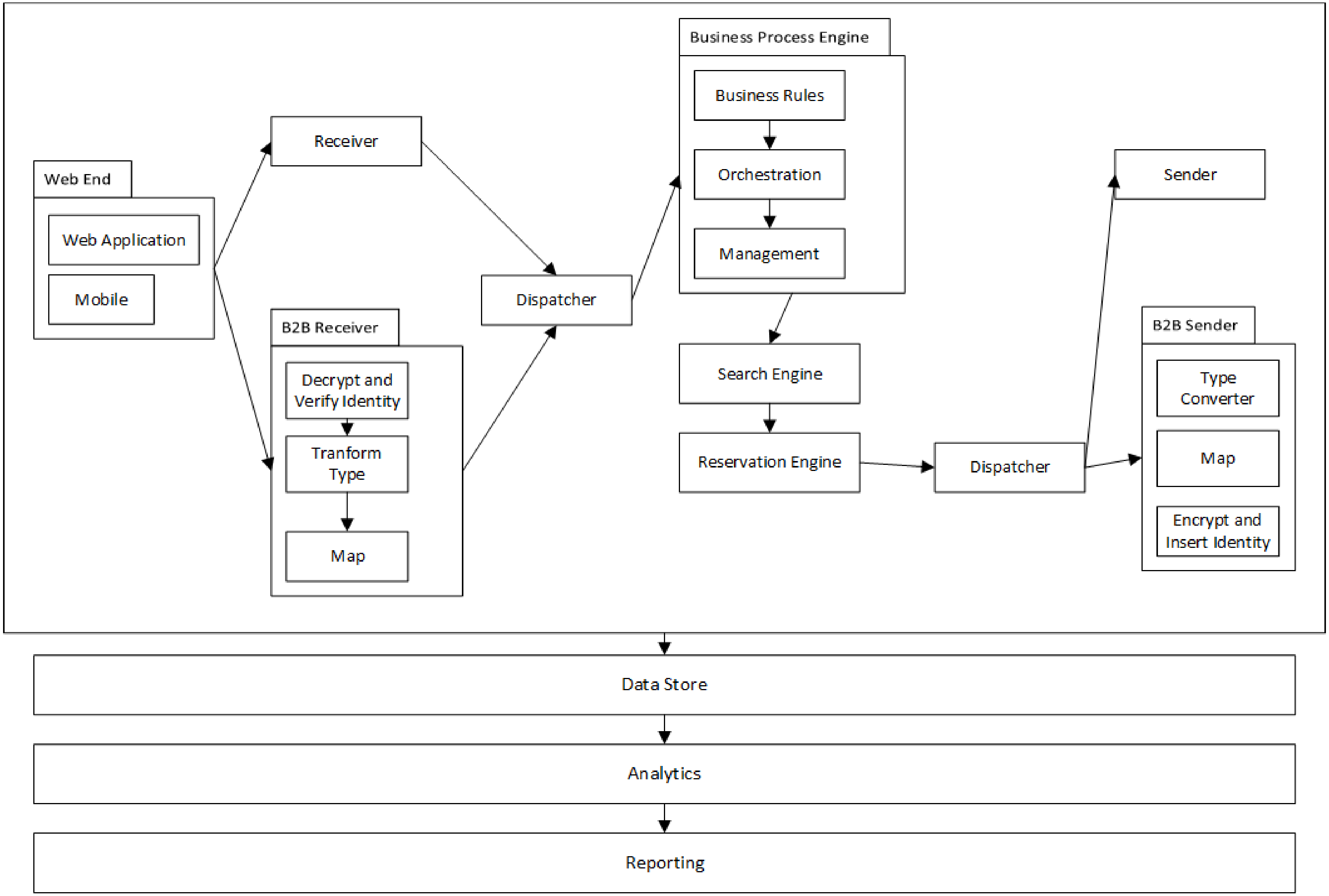
Logical Architecture Model: Behavior



Achieving Quality Attributes



Architectural Styles



Web End

Web Application

Mobile

Receiver

B2B Receiver

Decrypt and Verify Identity

Tranform Type

Map

Dispatcher

Business Process Engine

Business Rules

Orchestration

Management

Search Engine

Reservation Engine

Dispatcher

Sender

B2B Sender

Type Converter

Map

Encrypt and Insert Identity

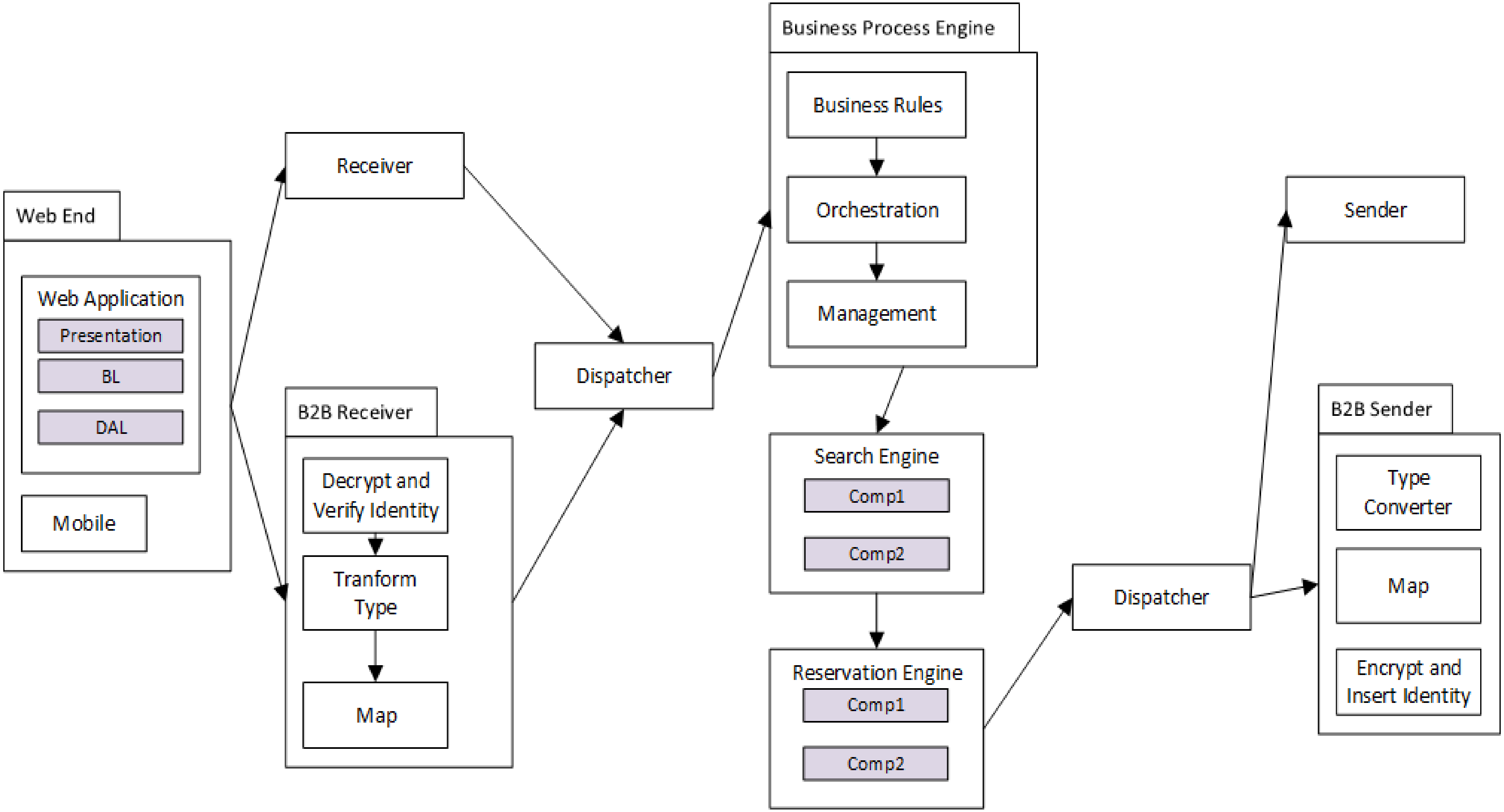
Data Store

Analytics

Reporting

Design

- **Focus is on the constituents of the lowest-level architecture-relevant components**



Abstraction Levels

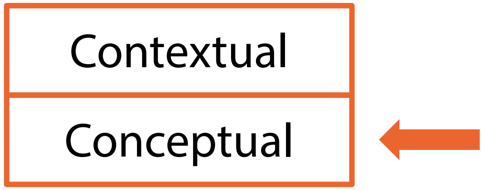
- **Abstraction levels:**
  - Encapsulate complexity
  - Look at the system from different viewpoints
  - Each viewpoint is meaningful for certain stakeholder groups

Contextual Level

Contextual

- **Why do we need the system?**
- **What are the business objectives?**
- **Typically answered in a project charter**
  - As a result of Enterprise Architecture or Portfolio Management

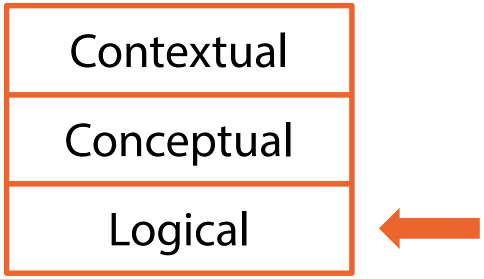
Conceptual Level


Contextual

Conceptual

- We study the requirements of the system
- What will the system do?
- Previous three modules covered the Conceptual level

Logical Level



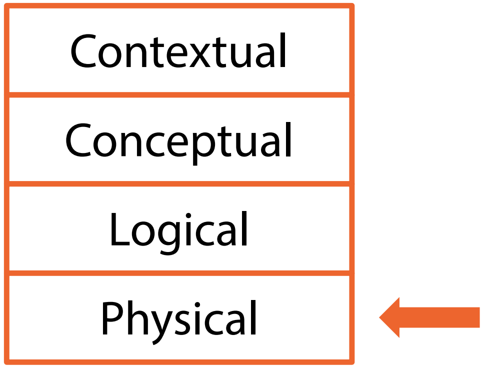
Contextual

Conceptual

Logical

- How will the requirements be met?
- Logical architecture and design are created
- Q: Can this level include technology information?
- A: Mostly no, but there can be exceptions
  - □ Ex: A Conceptual-level constraint to use an existing middleware product (ex: BizTalk, IBM Integration, etc…)
  - □ This imposes major design considerations
  - □ So it becomes a major part of the logical architecture and design
- Remember: Architecture (and design) is an art and never an exact science!

Physical Level



Contextual

Conceptual

Logical

Physical

- **With what will the solution be built?**
- **Concerns are physical components, products, specifications, technologies, etc…**
- **Mostly an architecture concern but details could be left to detailed design**
  - Architecture: Capacity planning and hardware sizing based on quality attributes results in hardware and network specs
  - Architecture: Load balancing and secure transmission
  - Design: Other details about model specs and cabling

Viewpoints and Views

- **Recall: analysis models show the problem at different abstraction levels (for example using the 4+1 View Model)**
- **Architects do the same to model the solution from different perspectives**
- **Views are representations of one or more architecture aspects**
  - Illustrate how the architecture addresses specific stakeholder groups concerns
- **Viewpoints define stakeholder groups concerns, and then define patterns, templates, and principles for creating views**
  - A viewpoint acts as a library that guides the creation of views
- **A view can be seen as an instance of a viewpoint**

The 4+1 ‘Viewpoint’ Model

- **Let’s rethink the 4+1 View Model**
- **Following the definition of viewpoints and views:**
  - The Logical, Process, Development, and Physical ‘views’ are actually viewpoints
  - The models created for each viewpoint are the views
  - Ex: A class diagram is a view of the Logical viewpoint
  - Ex: A deployment diagram is a view of the Physical viewpoint

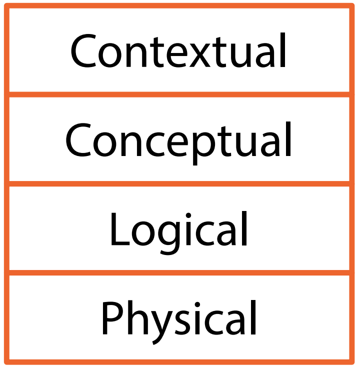
Benefits of Viewpoints and Views

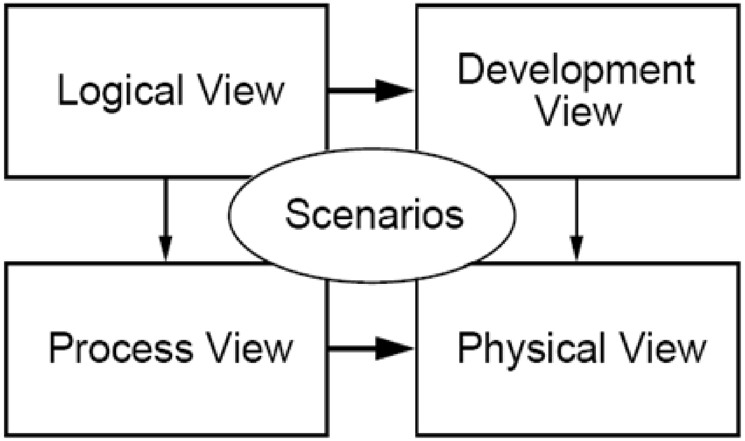
- **Creating viewpoints and views is an essential architectural skill:**
  - Separation of concerns: through different – but related views – architects can focus on different aspects of the overall solution
  - Stakeholder management: different views tackle the concerns of different stakeholder groups
    - Different stakeholder groups only see the information they care about
    - Ex: Infrastructure team only cares about physical models (suitable for their skills and expertise)
  - Guidance for development: views guide design and development
    - Developers can then focus on specific scope rather than the entire architecture

Static and Dynamic Views

- **Static system aspect covers design-time organization**
  - How components are structured and related
- **Dynamic system aspect covers runtime organization**
  - How components interact and change state in response to events
- **Viewpoints cover both aspects**

Abstraction Levels and Views

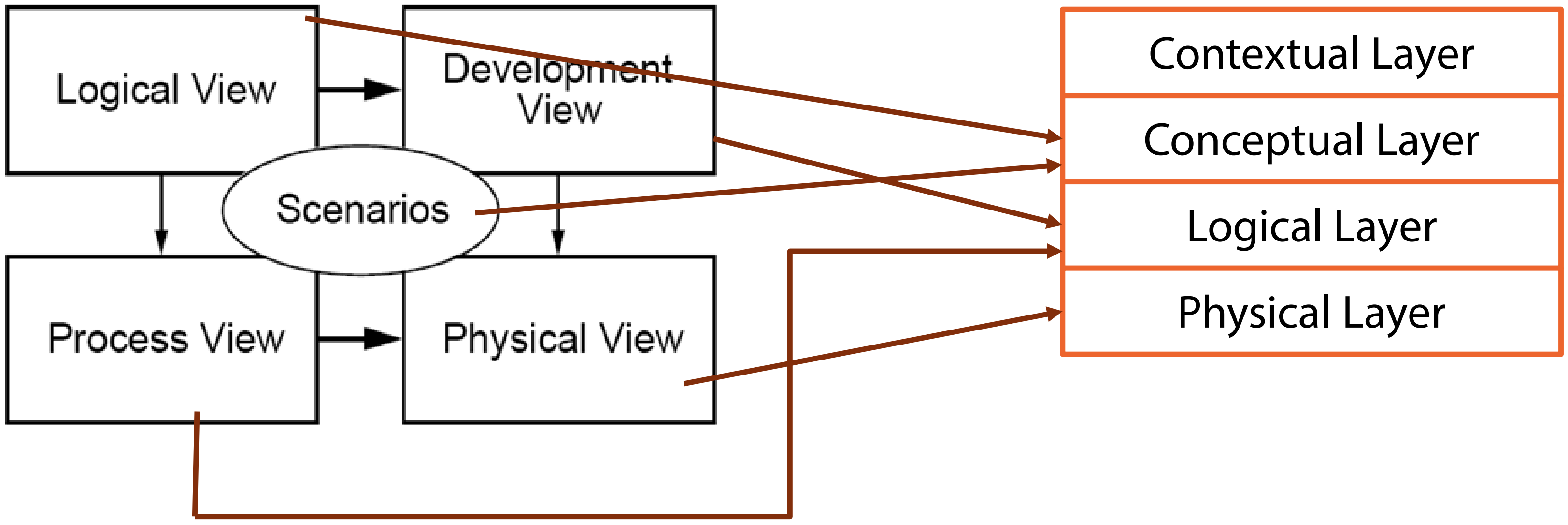




- **How are abstraction levels and view related?**

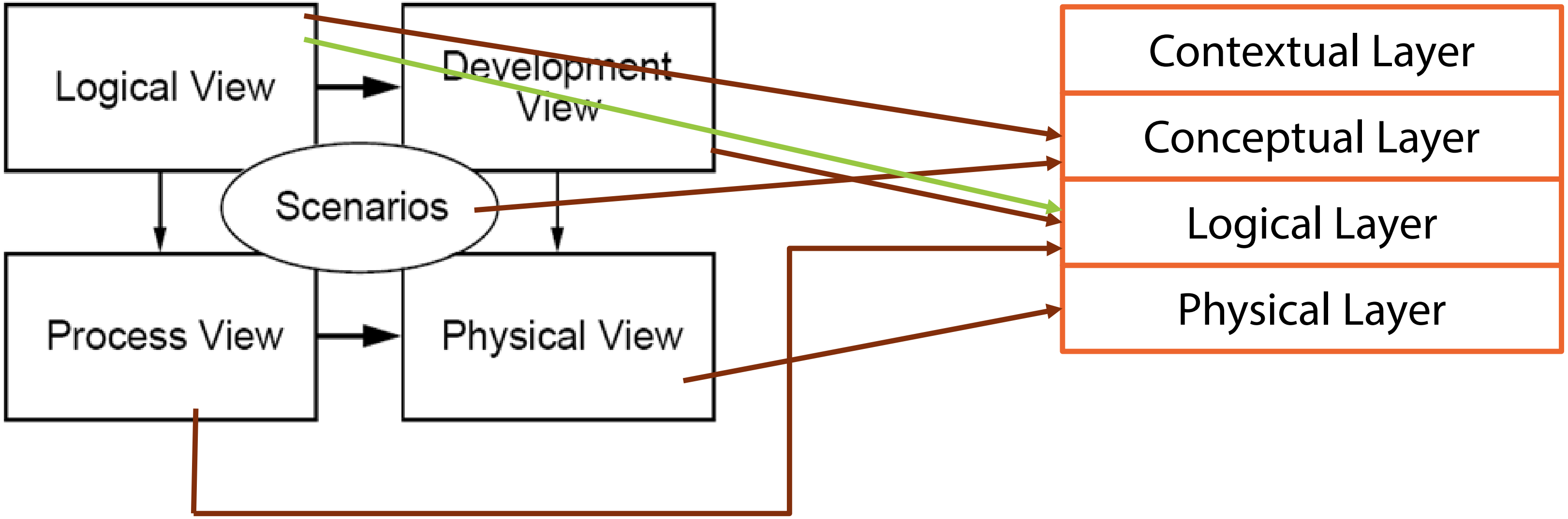
Abstraction Levels and Views

- **Recall how we used the 4+1 model to create analysis models:**



Abstraction Levels and Views

- Different mappings can be used as long as different perspectives are covered

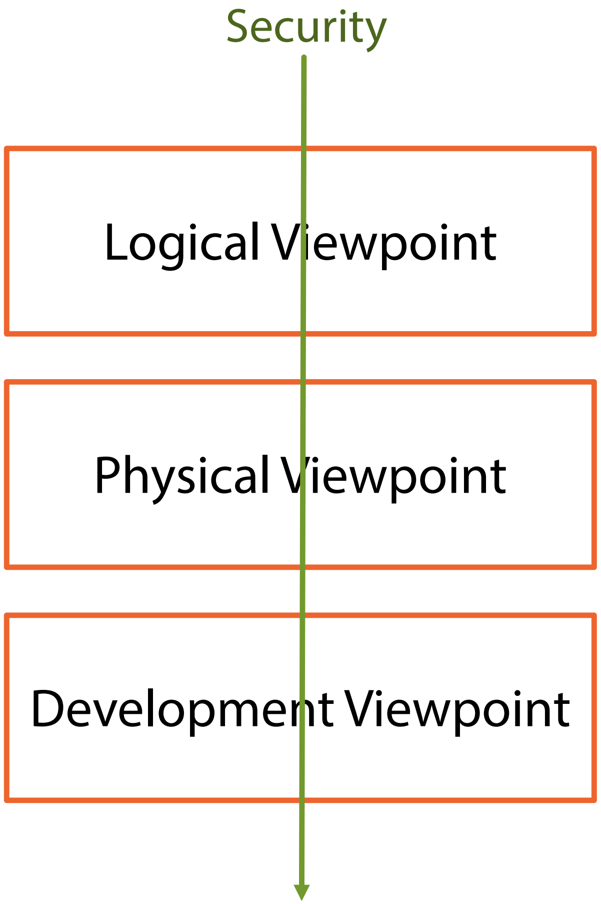


- Do not mix views and abstraction levels
  - □ Views model the system from a certain perspective at a certain abstraction layer
  - □ Abstraction layers encapsulate a level of detail

Quality Attributes

- **Quality attributes are non-functional requirements**
- **They are properties of the system, rather than functionalities**
  - Ex: Performance, Reliability, Security, Flexibility, etc…
- **Quality attributes are properties across the system**
  - They are cross-cutting
  - Therefore they affect the design of multiple viewpoints

Ex: Security Quality Attribute



Security

Logical Viewpoint

- Authentication and authorization
- Tamper proof communication (i.e. digital signature)
- Discrete communication (i.e. encryption)

Physical Viewpoint

- Secure password storage and transmission
- Protect web-facing servers from attacks (ex: via Firewalls)

Development Viewpoint

- Code access security
- Authorized access for source code

Non-functional Requirements

- **Constraints – in addition to quality attributes – are non-functional requirements**
  - Process, Infrastructure, technology, and environmental constraints
- **How do constraints affect viewpoints?**
  - A constraint can manifest itself as a quality attribute
    - Ex: A governmental regulation to use public/private key pair
  - Process constraints for example are unlikely to affect viewpoints
  - Technology constraints – such as adopting a specific middleware – will affect multiple viewpoints

Architectural Description

- **Requirement models are documented in Requirements Specification Document**

- **Architectural models are documented in an Architectural Description document (AD)**

- **AD essential sections:**
  - Viewpoints
  - Views (through models)
  - How quality attributes affect viewpoints (i.e. how are they fulfilled)
  - Other non-functional constraints
  - Decisions documentation (i.e. logic behind taking major architectural decisions)

- **Detailed design is not part of the AD**

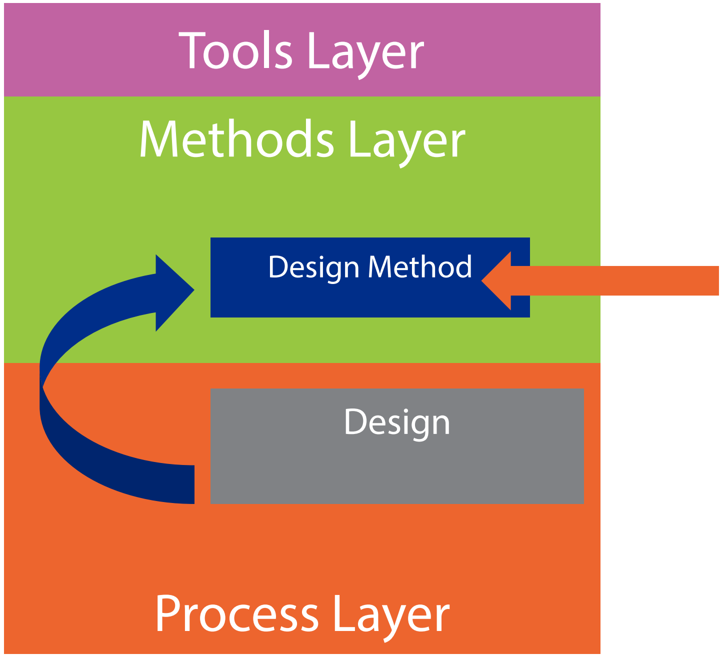
Summary

- **Design activity is divided into architectural and detailed design**
- **Architecture is concerned with**
  - Architecture-relevant components
  - Non-functional properties
  - Might conform to architecture styles
- **Design is concerned with**
  - Low-level components
  - Constituents of the architecture relevant components
  - Might conform to design patterns

Summary

- **Viewpoints and view help looking into the solution from different perspectives**
  - Viewpoints are libraries that guide views creation
  - A view describes one or more aspects of the architecture
- **Quality attributes are cross-cutting concerns**
  - They affect multiple viewpoints
  - Primary concern of architectural design

What’s Next?


Tools Layer
Methods Layer
Design Method
Design
Process Layer