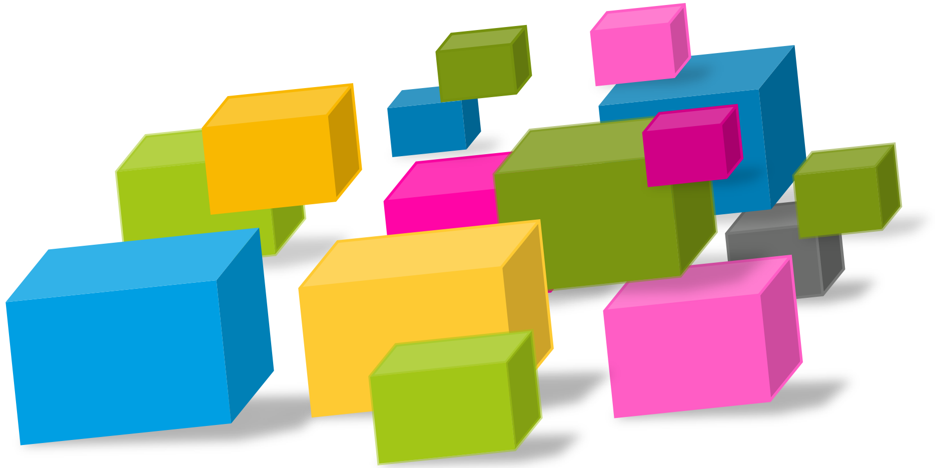
Architecture Views & ViewPoints

– A Review

Kim Horn

Version 0.1 1 August 2015

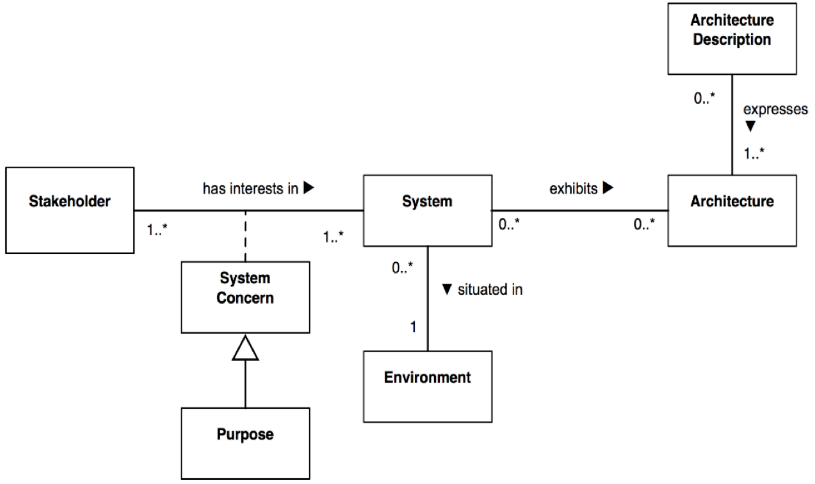


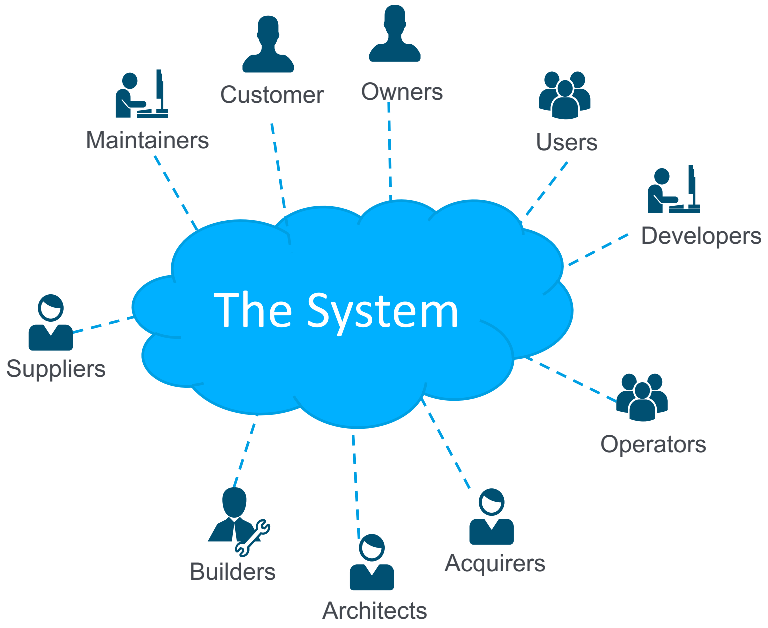
Agenda

- Stakeholders and Views
- 8 Models Reviewed:
    - RUP 4+1 Views – Philip Kruchten, 1995 (IEEE Software);
    - Siemens Four – Soni, Nord & Hofmeister, 1995;
    - Business Component Factory – Herzum & Sims 1999;
    - SEI View Model – Clements et al, 2002;
    - Garland & Anthony 2003;
    - ISO RM-ODP - *ISO/IEC 10746;*
    - Microsoft
    - Rozanski & Woods – 7 Viewpoints, 2012;
    - Archimate

Stakeholders

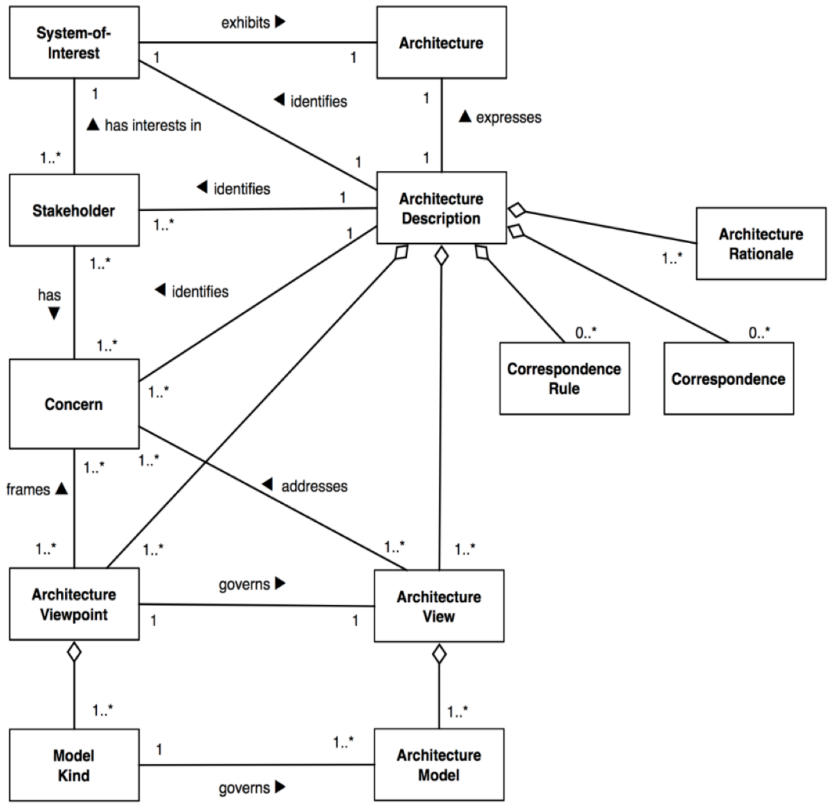
- A stakeholder is any person or group who have some vested interest or concerns about the system.
- Each Stakeholders has a different need or concern and views the system in a different way.





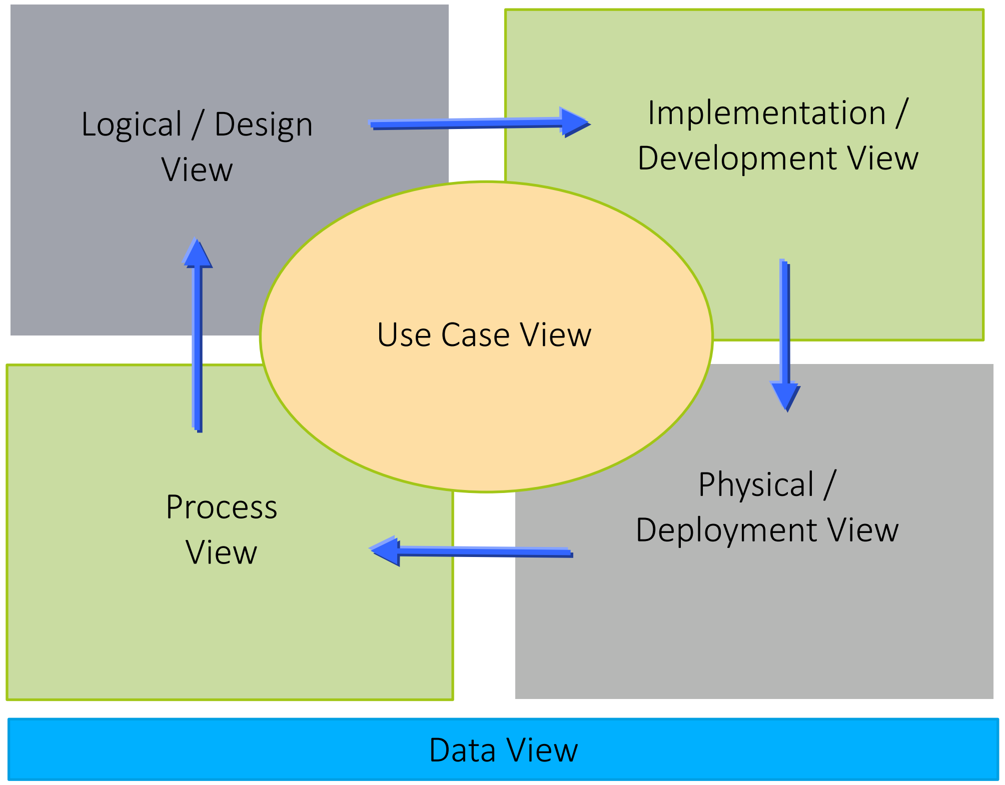
Architecture Description Standard

System	Collection of components (business and technical) organized to accomplish a specific function or set of functions
Architecture	The system's fundamental organization, embodied in its components, their relationships to each other and to the environment, and the principles guiding its design and evolution
Architecture Description	Collection of artifacts that document an architecture
Stakeholder	Person (or group of people) who has a key role in, or concern about, the system
Concern	Key interests that are crucially important to the stakeholders in the system and determine the acceptability of the system
View	Representation of a whole system from the perspective of a related set of concerns
Viewpoint	Defines the perspective from which a view is taken: • How to construct and use a view • Information that should appear in the view • Modeling techniques for expressing and analysing the information • Rationale for these choice



Ref: ISO/IEC/IEEE 42010 *Systems and software engineering — Architecture description*

RUP 4+1 Views – Philip Kruchten, 1995 (IEEE Software)



4+1 Views

The logical view supports behavioural requirements—the services the system should provide to its end users.

Designers decompose the system into a set of key abstractions, taken mainly from the problem domain. These abstractions are objects or object classes that exploit the principles of abstraction, encapsulation, and inheritance. In addition to aiding functional analysis, decomposition identifies mechanisms and design elements that are common across the system.

The development view focuses on the organization of the actual software modules in the software-development environment. The units of this view are small chunks of software—program libraries or subsystems—that can be developed by one or more developers. The development view supports the allocation of requirements and work to teams, and supports cost evaluation, planning, monitoring of project progress, and reasoning about software reuse, portability, and security. It is the basis for establishing a line of product.

4+1 Views

The *physical view* takes into account the system’s requirements such as system availability, reliability (fault tolerance), performance (throughput), and scalability.

The *process view* takes into account some requirements such as performance and system availability. It addresses concurrency and distribution, system integrity, and fault tolerance. The process view also specifies which thread of control executes each operation of each class identified in the logical view. The process view can be seen as a set of independently executing logical networks of communicating programs (“processes”) that are distributed across a set of hardware resources, which in turn are connected by a bus or local area network or wide area network.

4+1 (+1) Issues … only some of them

- Provides no model for organising or decomposing the system:
  - What are the views of; viewpoints are unclear ?
  - Where are the platforms, infrastructure, etc ?
  - Where are the tiers ?
- Variety of naming schemes and many interpretations ?
- Deals with layers add hoc within Views – not views of layers;
- Use Cases are not architecture.
  - These may drive logical choices but not infrastructure;
  - NFRs drive architecture.
- Process View is usually empty;
- Implementation view is not architectural – should be 'Component' view;
- Data is not a view or a global concern.
  - Data is spread across layers and tiers, e.g. Value Objects, DTO's
- So 2 views only…. but not sure of what.

SEI View Model has 3 Viewtypes:

Each Viewtype has a set of Styles that are recurring forms.

- **Module** – the systems principal units of implementation, a design time view.
  - Styles: decomposition, uses, generalisation, layered, aspects, data model.
- **Component & Connector (C&C)** – the systems units of execution, a runtime view.
  - Styles: Architectural Styles, including: client server, pipes & filters, peer to peer, SOA, publish subscribe, shared data, and multi-tier.
- **Allocation** – the relationships between a systems software and its development and execution environments.
  - Styles: deployment, implementation, work assignment.

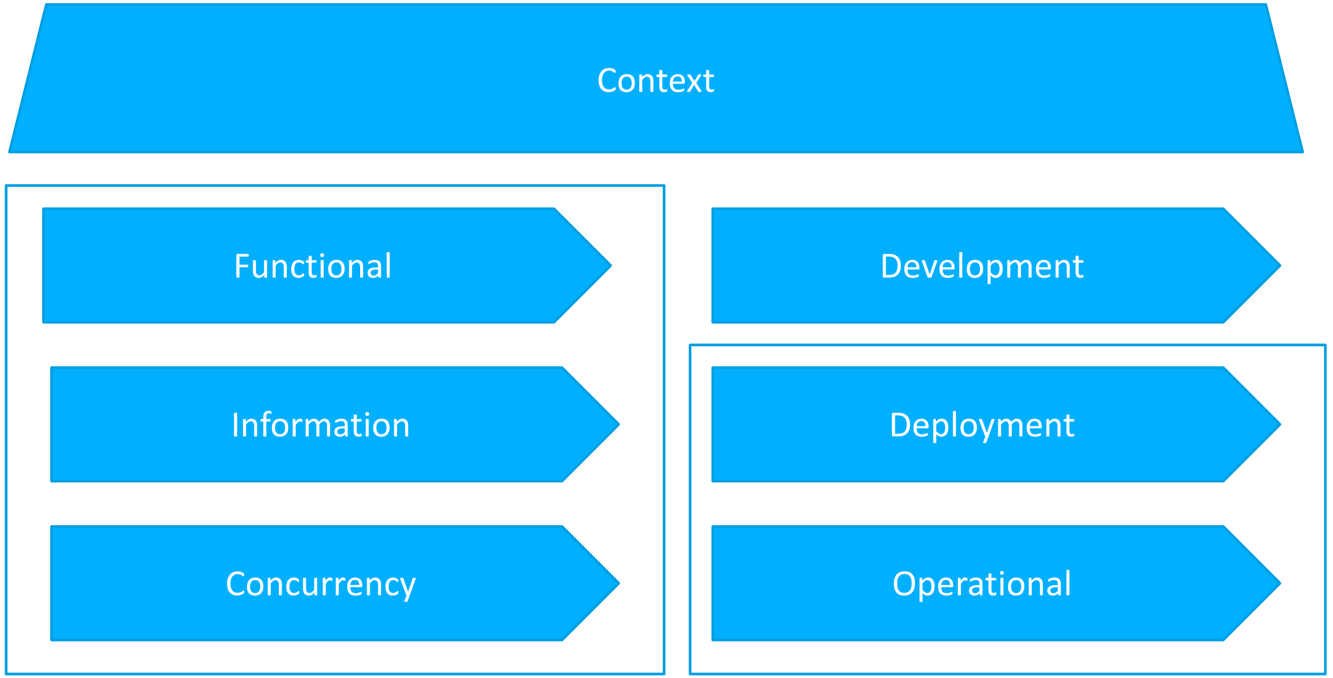
Siemens Four Views

- **Conceptual**: Major conceptual design elements of the system and their relationships.
- **Module Interconnection**: Two orthogonal views:
  - Functional decomposition of systems and sub-systems;
  - Layers.
- **Code:** How the source code, binaries and libraries are organised into a development environment.
- **Execution**: Dynamic structures of the system.
- Happened at about the same time as 4+1; strangely similar.

Comparison of these 3 Models across Groups of Features

Group	Group Features	"4+1" model	SEI model	Siemens model
Functional	Decomposition, Uses, Layers, Abstraction.	Logical view	Module viewtype	Module view
Behavioural	Process, Concurrency, Shared Data, Client–Server	Process view	C&C viewtype	Execution view
External	Implementation, Work assignment	Development view	Allocation viewtype	Code view

Rozanski & Woods Seven ViewPoints



Rozanski & Woods ViewPoints

- **Context**: the relationships, dependencies, and interactions between the system and its environment (the people, systems, and external entities with which it interacts).
- **Functional**: the system's functional elements, their responsibilities, interfaces, and primary interactions. It drives the shape of other system structures such as the information structure, concurrency structure, deployment structure, and so on. It also has a significant impact on the system's quality properties such as its ability to change, its ability to be secured, and its runtime performance.
- **Information**: the way that the architecture stores, manipulates, manages, and distributes information. The objective of this analysis is to answer the big questions around content, structure, ownership, latency, references, and data migration.

Rozanski & Woods ViewPoints continued…

- **Concurrency**: identifies the parts of the system that can execute concurrently and how this is coordinated and controlled. Includes models that show the process and thread structures and the interprocess communication mechanisms used to coordinate their operation.
- **Development**: Communicate the aspects of the architecture of interest to those stakeholders involved in building, testing, maintaining, and enhancing the system.
- **Deployment**: the environment into which the system will be deployed, and the runtime dependencies. This view captures the hardware environment that your system, the technical environment requirements for each element, and the mapping of the software elements to the runtime environment that will execute them.
- **Operational**: Describes how the system will be operated, administered, and supported when it is running in its production environment. The aim of this viewpoint is to identify system-wide strategies for addressing the operational concerns of the system's stakeholders and to identify solutions that address these.

ViewPoints Intersect with Perspectives, Example

	Security	Performance	Availability	Evolution
Functional				Flexible interfaces, extension point
Informational	Access control, AAA			
Concurrency		Shared Resources, blocking, queuing		
Operational				

RM-ODP Provides 5 Viewpoints

- **Enterprise**: focuses on the purpose, scope and policies for the system. It describes the business requirements and how to meet them.
- **Information**: focuses on the semantics of the information and the information processing performed. It describes the information managed by the system and the structure and content type of the supporting data.
- **Computational**: enables distribution through functional decomposition on the system into objects which interact at interfaces. It describes the functionality provided by the system and its functional decomposition.
- **Engineering**: focuses on the mechanisms and functions required to support distributed interactions between objects in the system. It describes the distribution of processing performed by the system to manage the information and provide the functionality.
- **Technology**: focuses on the choice of technology. It describes the technologies chosen to provide the processing, functionality and presentation of information.

Business Component Factory has 4 Architectures:

- **Technical** – the Component execution environment, tools, UI frameworks, and technical services required to develop and run the system.
- **Application** – the architectural decisions, patterns, guidelines, and standards
- **Project Management** – the concepts, guidelines, principles and management tools required to build a scalable system with a large team.
- **Functional Architecture** – the specification and implementation.

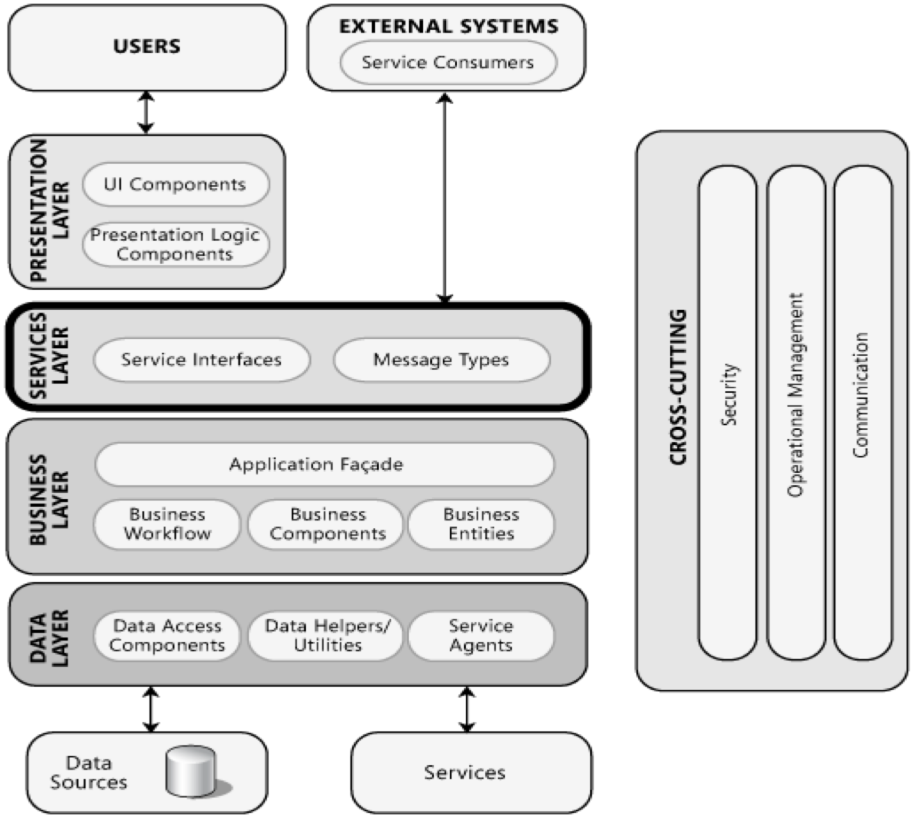
MicroSoft Application Architecture – Old Version

Add:

- Layers
  - Service Layer
    - B2B by-passing presentation
- Components
- Patterns
- Cross-cutting concerns

Issues:

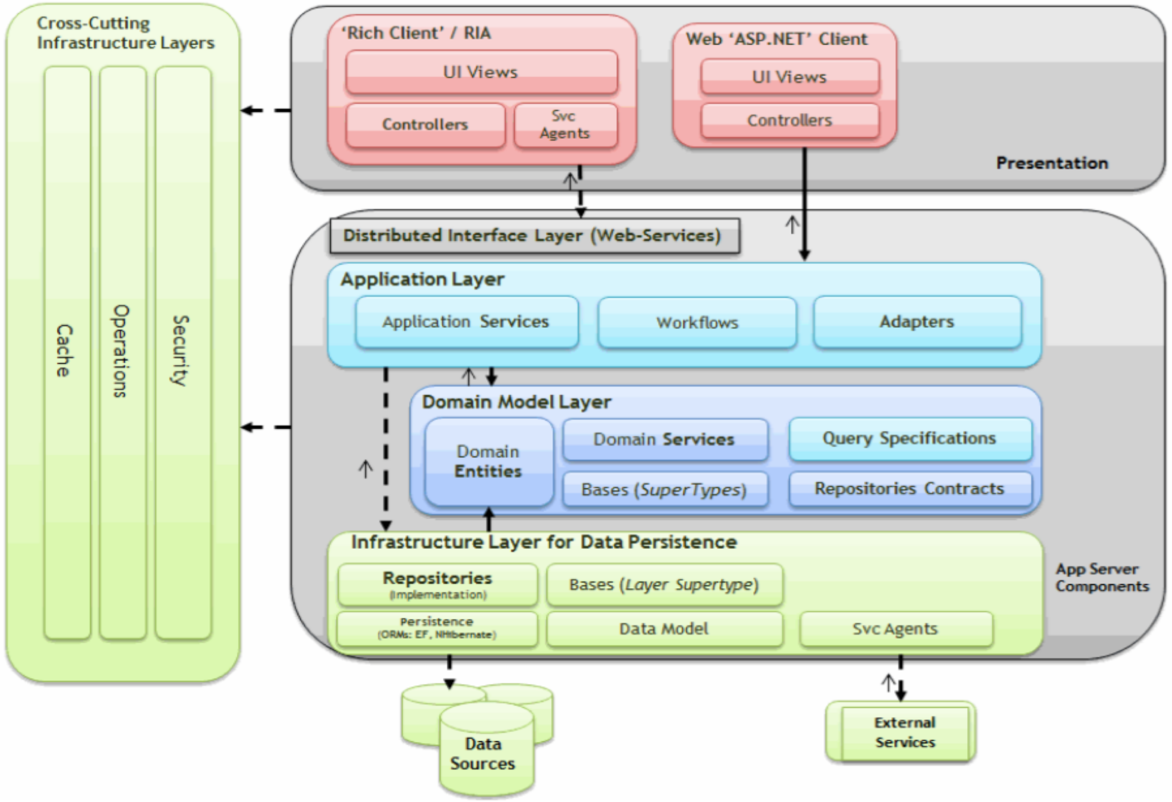
- What no Views ?
- The layers are tiers.
- Logical only ?
- What gets deployed ?
- Infrastructure stack ?
- Systemic Qualities ?
- and so on.....



MicroSoft Application Architecture – New Version

Microsoft's layered model with Domain Driven Design (DDD) Building Blocks.

Services appear in many layers



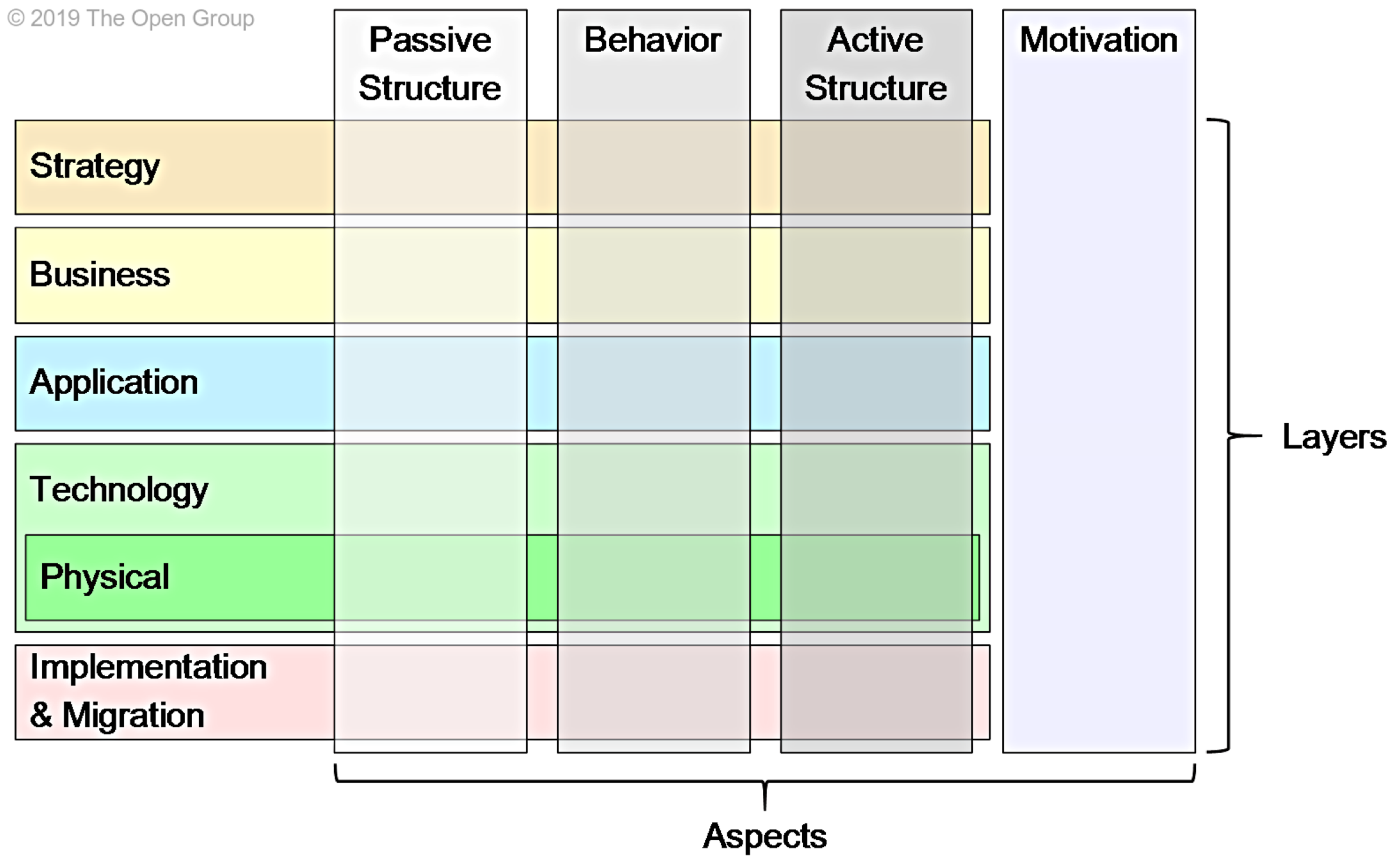
Microsoft Views of the Application Architecture

- **Conceptual view:** a *business model* defines the business requirements and the users' view of the application. Modeling techniques, such as use case analysis, activity diagrams, process design, and business entity modeling, help to build a description of the key business processes and the data they use, free of implementation technology.
- **Logical view:** *application models* are used to describe how to meet business objectives and requirements. Concerned with the overall structure of the application. They decide on the mapping of data management and process steps, they design the interactions between parts of the model in terms of logical messages and sequences, and they determine what data and state should be held by the model.
- **Physical view:** maps application elements of real technologies. In this way application models are realized as *implementation models*. Part of this task is undertaken during conventional development but much of the implementation activities are classed as framework completion, extending the underlying software infrastructure. This infrastructure shields the developer from the intricacies of technology.

Microsoft Views Of the Technology Architecture

- **Conceptual view:** used to map out areas of technology into a structure and framework; to define, name, and position these areas for a common understanding between the IT supplier and the organizations using the technology. Ensures that all the technology areas required to implement an organization's operational or nonfunctional requirements are defined and available to the organization.
- **Logical view:** where the major functional elements that provide support for enterprise-scale operational requirements and their interrelationships are provided. Enterprise technology elements such as databases, mail systems, transaction support, and reliable messaging are provided in the logical view. The technologies that are provided at this level are normally packaged together as servers by enterprise software vendors.
- **Physical view:** elements are mapped to real technologies for both hardware and software. Technology architectures are realized as complete systems of networks, servers, operating systems, and so on. Actual physical locations, server product names, and connectivity are shown at this level

Archimate



© 2019 The Open Group
Passive Structure
Behavior
Active Structure
Motivation
Strategy
Business
Application
Technology
Physical
Implementation & Migration
Layers
Aspects

Archimate Viewpoints

ArchiMate suggests a set of example viewpoints that can be used as starting points for modelling efforts. Each of these ArchiMate viewpoint comprises elements from different ArchiMate layers, addressing specific stakeholder concerns.

- **Basic Viewpoints**: Concepts from the three layers of Business, Application, and Technology may be used.
- **Motivation Viewpoints**: For modelling motivational aspects of an architecture.
- **Strategy Viewpoints**: For describing the strategic aspect of the enterprise by describing the high-level strategic direction and make-up of the enterprise.
- **Implementation and Migration Viewpoints**: For modelling the management of architecture change, the transition from baseline to target architecture and relationships between programs and projects.

Basic Viewpoints

1. **Composition**: Viewpoints that define internal compositions and aggregations of elements.
2. **Support**: Viewpoints where you are looking at elements that are supported by other elements. Typically from one layer and upwards to an above layer.
3. **Cooperation**: Towards peer elements which cooperate with each other. Typically across aspects.
4. **Realisation**: Viewpoints where you are looking at elements that realize other elements. Typically from one layer and downwards to a below layer.