**Zachman Framework:**

The Zachman Framework is an enterprise ontology and is a fundamental structure for Enterprise Architecture which provides a way of viewing an enterprise and its information systems from different perspectives, and showing how the components of the enterprise are related. Provides a formal and structured way of viewing an organization. It provides two-dimensional classification schema in which Column represents questions that are asked to enterprise and row represents distinct view of an organisation.

**Pros:**

* Easy to understand
* Provides a holistic perspective on the whole enterprise while at the same time allowing to focus on certain aspects of the object
* It is defined independently of tools or methodologies

**Cons:**

* The large number of cells, which is an obstacle for the practical applicability of the framework
* Does not give step-by-step process for creating a new architecture
* Does not provide any insight into relationship in between single models
* It does not specify how to collect, manage or interpret the information that is organized in the framework

**TOGAF Framework**

The Open Group Architecture Framework (TOGAF) is an enterprise architecture methodology that offers a high-level framework for an enterprise software development. It provides an approach for designing, planning, implementing, and governing an enterprise information technology architecture

**Pros:**

* It provides common terminology across the world
* Low cost for becoming a TOGAF certified
* Flexible for different business and can be integrated with other frameworks
* TOGAF knowledge body is available on Open Group website – making it an easily accessible and free resource
* Likely to be relevant in large and government organizations

**Cons:**

* TOGAF is not entirely consistent being written by many people over many years; each keen to make their own contribution
* TOGAF defined 75+ artifacts like deliverables, diagrams, matrices but doesn’t include a single template for any of these
* Provide little concrete guidance to solution architects.
* Some parts outside ADM are not so well considered
* Less likely to be relevant in smaller and digitized organizations

**Zachman v/s Toaff:**

The Zachman Framework is not a methodology but rather a template describing how different abstract ideas are viewed from different perspectives.

Zachman is focused on identifying the different viewpoints that might be relevant for different purposes while TOGAF is focused on the process of developing architectures.

**4 + 1 View Model:**

4+1 is a [view model](https://en.wikipedia.org/wiki/View_model) used for "describing the architecture of software-intensive systems, based on the use of multiple, concurrent views".[[1]](https://en.wikipedia.org/wiki/4%2B1_architectural_view_model#cite_note-Kru95-1) The views are used to describe the system from the viewpoint of different stakeholders, such as end-users, developers, system engineers, and project managers. The four views of the model are logical, development, process and physical view.

* **Logical view**: The logical view is concerned with the functionality that the system provides to end-users. [UML diagrams](https://en.wikipedia.org/wiki/Unified_Modeling_Language) are used to represent the logical view, and include [class diagrams](https://en.wikipedia.org/wiki/Class_diagram), and [state diagrams](https://en.wikipedia.org/wiki/State_diagram).
* **Process** **view**: The process view deals with the dynamic aspects of the system, explains the system processes and how they communicate, and focuses on the run time behavior of the system. The process view addresses concurrency, distribution, integrator, performance, and scalability, etc. UML diagrams to represent process view include the [sequence diagram](https://en.wikipedia.org/wiki/Sequence_diagram), [communication diagram](https://en.wikipedia.org/wiki/Communication_diagram), [activity diagram](https://en.wikipedia.org/wiki/Activity_diagram).
* **Development view**: The development view illustrates a system from a programmer's perspective and is concerned with software management. This view is also known as the implementation view. It uses the UML [Component diagram](https://en.wikipedia.org/wiki/Component_diagram) to describe system components. UML Diagrams used to represent the development view include the [Package diagram](https://en.wikipedia.org/wiki/Package_diagram).
* **Physical view**: The physical view depicts the system from a system engineer's point of view. It is concerned with the topology of software components on the physical layer as well as the physical connections between these components. This view is also known as the deployment view. UML diagrams used to represent the physical view include the [deployment diagram](https://en.wikipedia.org/wiki/Deployment_diagram).
* **Scenarios**: The description of an architecture is illustrated using a small set of [use cases](https://en.wikipedia.org/wiki/Use_case), or scenarios, which become a fifth view. The scenarios describe sequences of interactions between objects and between processes. They are used to identify architectural elements and to illustrate and validate the architecture design. They also serve as a starting point for tests of an architecture prototype. This view is also known as the use case view.

**Integration Architecture:**

Integration architecture is a software architecture that facilitates the integration of multiple IT components. This architecture changes with the advances in cross-platform utility and other development paradigms for new kinds of digital operations. Examples: SOA, Microservices, EDA, ETL

**Event Driven Architecture:** Event-driven architecture is a software architecture paradigm promoting the production, detection, consumption of, and reaction to events. An event can be defined as "a significant change in state". For example, when a consumer purchases a car, the car's state changes from "for sale" to "sold".

**Extract Transform Load:** In computing, extract, transform, load is the general procedure of copying data from one or more sources into a destination system which represents the data differently from the source or in a different context than the source. ETL is short for extract, transform, load, three database functions that are combined into one tool to pull data out of one database and place it into another database. Extract is the process of reading data from a database. In this stage, the data is collected, often from multiple and different types of sources.

**Catalyst Program:**  Mulesoft Catalyst Program is designed to deliver customer success, and one of the key components to doing so is our outcome-based delivery methodology of aligning to three core pillars: business outcomes, technology delivery, and organizational enablement.

As an overview, here is what we can expect across each pillar:

* Business outcomes: Define clear outcomes and KPIs with stakeholder alignment.
* Organizational enablement: Ensure organizational readiness with Anypoint Platform.
* Technology delivery: Enable platform availability and team readiness to build APIs and integrations.

**Diagramming Tools:**

**ArchiMate:**

Archimate offers a visual language to model and capture enterprise architecture. It provides a means to visualize relationships within and between different domains. These domains primarily include products/services, processes, organization, data applications and technical infrastructure.

With 3 types of layers, ArchiMate provides a way to look at service-oriented models. From top to bottom, they are business, application and technology. The further we go down, the more information we get in terms of implementation.

At the top level, the business layer highlights services provided for customers. In the middle, the application layer focuses on applications that support those services available to customers. At the bottom we can find the technology layer which shows the technical infrastructure offering services to those applications in the application layer.

**Unified Modelling Language (UML):**

A UML diagram is a diagram based on the UML (Unified modelling Language) with the purpose of visually representing a system along with its main actors, roles, actions, artifacts or classes, in order to better understand, alter, maintain, or document information about the system.

**Fundamental Modelling Concepts (FMC):**

Fundamental modelling concepts (FMC) provide a framework to describe software-intensive systems. It strongly emphasizes the communication about software-intensive systems by using a semi-formal graphical notation that can easily be understood.