# Module 1

# Anypoint platform

Anypoint platform is a hybrid, highly productive integration platform. It is used

* To connect various application eg JMS connector, SMTP connector. There area above 130+ connectors
* Various prototype

It consist of

* Web application suite
* Mule runtime

Anypoint platform is a subscription based service the anypoint sales.

Based on the subscription the features are available. It also has different pricing

# Architecture

It is used to implement the project. There are different kinds of architecture that are grouped as Enterprise Architecture and Integration Architecture/ Solution Architecture in terms of integration.

**Enterprise Architecture** EA are high level strategic architecture . EA are well experienced person who might not know the low level functionality of the applications. The role of a EA is to understand the need of a business in two to three years time like increase profit and design a long term strategy to achieve it . It consist of multiple projects and project are prioritized

**Integration Architecture** It is a project specific architecture. It can be APIs project, ETL project, file transfer, event driven architecture

1. Enterprise Architecture

It is a conceptual blue print that defines the structure and operation of organization. It helps to connect service, process, people.

Need to follow strategy to achieve a business goal

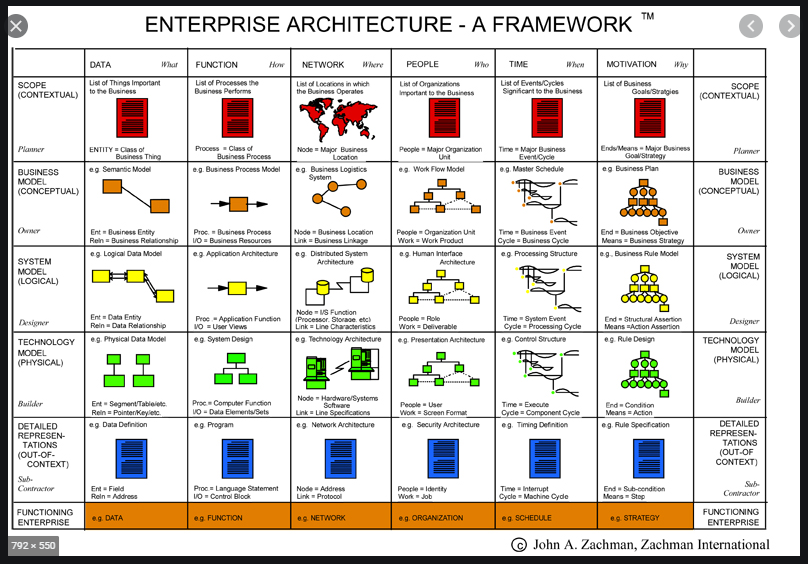
EA is used to collaborate the different existing system. Ensures that the collaboration between legacy system and new system is better . He is responsible to find the loop holes in the existing system and give higher level solution for the identified issue

Domains in ES

1. Business: defines the process and standard by which the business operate
2. Application: define the interactions among the process
3. Technology: defines the tools used to achieve the process goal like h/w, OS, programming and n/w tools
4. Information: define the organization of any data (file, image) that the business need to operate efficiently
5. Framework

**Types of framework:**

1. **Zachman Framework**





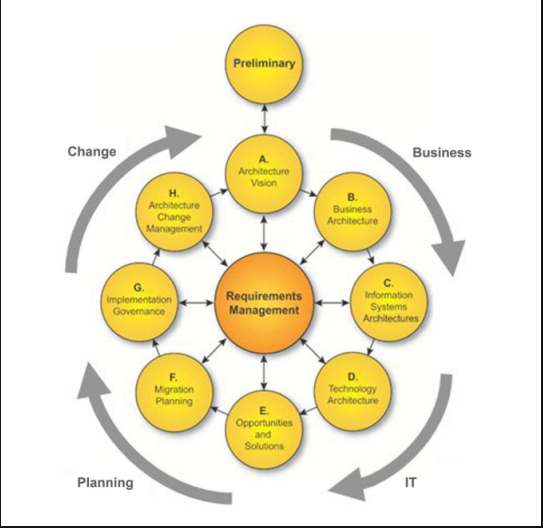
This model is a guidelines that EA follows so that he does not miss anything while creating the architecture.

It is a two dimensional schema that cover the six stakeholder perspective: planner, owner, designer, builder, subcontractor and user. Then from each user perspective it tries to answer questions different functions of an enterprise: Why, How, What, who, where and when

Questions that should come to achieve the goal

* What is the scope – amazon
* Do you need all feature
* How to categorize it
* What are the important things for the business. What kind of data are we dealing with: confidential, sensitive or public data
* To collect data we need model
* How the data flow in the system
* Where should the data initiate and how to make data secure. Where are the data stored
* Network for the data transmission
* How to choose right people

1. **TOGAF**

The Open Group Architectural Framework. High level framework

It aims to reduce error, on time delivery. On budget

**Main difference over Zachman: it** provides a set of process

Key element is **ADM**: Architecture Development Model. It helps a business to establish a process around a lifecycle of EA

ADM consist of the entire life cycle of the EA. It starts with a preliminary phase where information are collected to create the architecute

1. Architecture: Define the scope of the enterprise, identify stake holders, business vision and verify it .
2. Business: design business design based on agreed on the A
3. Information: develop information system architecture to support architecture vision
4. Technology: describe the technology needed to support the vision
5. It is where the implementation start Identify the opportunities and its solutions
6. It has two state: a base line (current state) and target architecture (future scope). How to achieve the target architecture and how to perform migration plan . It can me technology migration, data migration,
7. It provide how the design will be implemented. How the governance is implied
8. What changes can be done in the next cycle

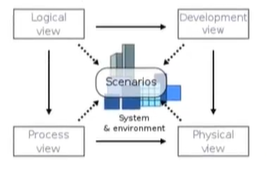
Requirement mgmt.: It is connected to all the phases. In each phase it checks if requirement are met or not

Data collected from one country cannot be stored in different country so for this **governance** is required

Note:

* After each phase a document and diagrams are provided: business requirement documents, function design, technical design, use case diagram.
* The architecture design as on needed for migration and planning

1. **4+1**



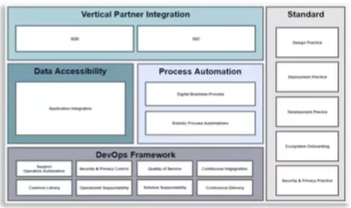
1. Logical view: high level diagram, end user perspective diagram
2. Development View: How to create a model, what technology is used . Look from developer perspective and focus on actual software developement
3. Process View : it is in system integration perspective. We can use sequence diagram
4. Physical view: Where the application is deployed and what framework is used. It considers the non-functional requirement of the system
5. Scenario and use cases : use a scenario to view all the 4 views

# API led connectivity

* Its is a method to implement the vision created by EA
* It is an architectural approach to connect data to applications through reusable and purpose full assets
* It is a multi-layered approach that scales IT capacity through its emphasis on modular components, decentralized authority over apps development and reusable assets.
* Instead of building one application with all the functionality, it uses a group of services each with own API that can be govern and monitored.
* Divided into three categories
  + System API: unlock data from the systems. Eg SAP , salesforce, e-commerce
  + Process API: Data are retrieved using these APIs and aggregated, transformed and orchestrated
  + Experience API: It is packaged, modified, as needed to best be consumed by end users eg web application
* Advantage:
  + Deliver project on time
  + Create reusable assets
  + Fast project development

# Integration Architecture

* It works at project level
* A project consist of business to business interaction, business to custome interaction, needs access to db, deal with dev ops framework
* Integration architect deals with integration of project components so that project is delivered in time, performance of the system to high , how data is transferred – technical
* over all the final application should be able to satisfy the customers



* Advantage
  + Use of APIs gives the reusability and speed delivery
  + Reduces the cost
  + Better management of the application

# Catalyst Program

It is a methodology and approach MuleSoft has developed to enable its customers to succeed. It is designed to enable customer success while implementing Mulesoft services

Why it is need?

* Organizations are not able to use the services to its full capacity so this acts as a guidelines for success of the organization
* It provides a structured step by step guide of best practices to accelerate customer success

IT includes three pillars:

* **Business outcomes:** defines the business goals and the **Key performance indicators** (**KPIs**) with the stakeholders alignment
* **Technology delivery**: enable platform availability and team readiness to build APIs
* **Organization enablement:** ensures the organizational readiness with anypoint platform

<https://trailhead.salesforce.com/fr/content/learn/modules/mulesoft-catalyst-playbooks/learn-about-the-mulesoft-catalyst-delivery-methodology?trail_id=deliver-it-success-with-mulesoft-catalyst>

* MuleSoft Catalyst pillars are further broken into six paths for users to follow: Business outcomes, Anypoint Platform, Projects, Center for Enablement (C4E), Internal support, and Training.
* Customers start with a vision, identify quick wins, and establish the overall program based on the desired business outcomes. For example, the Business Outcomes playbook contains information about how to identify and measure outcomes and align them to KPIs and stakeholders.
* The Anypoint Platform playbook and Projects playbook demonstrate the path to operate MuleSoft’s Anypoint Platform.
* The C4E playbook enables organizations to maximize their results through best practices, reuse, and self-service.
* The Internal Support playbook helps customers build support models for projects involving Anypoint Platform.
* And finally, the Training playbook shows customers how to use enablement resources to provide training and certification.

The key components of Catalyst program:

* Catalyst Knowledge hub
* Catalyst Mobilize
* Catalyst Accelerators

What is it imp? Why each process is needed?

# Diagramming Tools

## Archimate

* It is a modelling technique **used for EA**
* IT consist of three layers: business layer, application layer, technology layer
* **business layer:** offers products and services to external customers which are realized by business actors and roles,
* **application layer**: supports the business layers with application services which are realized by application components
* **technology layer**: offer infrastructural services need to run application layer
* In the new version it has 3 more layers: strategy layer, physical layer and Implementation and migration layer
* **Strategy layer** it takes the strategy of the organization (Capability, Resource, Course of Action)
* **Physical Layer** to model physical elements; for example, manufacturing, logistics, and other physical environments.
* An ***Implementation and Migration Viewpoint*** model relate programs and projects to the parts of the architecture that they implement. This view allows modeling of the scope of programs, projects, project activities
* IN addition to the 6 layers there is a **Motivational concepts**. It is used to model the motivations, or reasons, that underlie the design or change of some enterprise architecture. The motivation extension adds motivational concepts such as goal, principle, and requirement
* With in the layers also we can diagrammatically represent the components for the business

## UML diagram

* **U**nified **M**odeling **L**anguage.
* It’s a rich language to **model structure** example software solutions, application structures, system behavior and business processes.
* There total 14 diagrams that are broadly classified into two
  + **Structure diagrams**: show the static relationships between the components in the system. Example Class diagram: attributes, functions and relation between classes
  + **Behavior diagrams**: show how the components in the system react to each other, they capture how the system changes, and in some diagrams, how it changes over. Example Activity diagrams: model the behavior of users and systems as they follow a process

## FMC

* The Fundamental Modeling Concepts (FMC)
* used to describe **software-intensive systems**
* The organization that require such system are travel agency, flights
* It is based on a precise terminology and supported by a graphical notation which can be easily understood.
* the goal is to capture its essential structures necessary to understand its (existing or planned) behaviour (internal and to its environment) and to describe these structures in a comprehensive way.

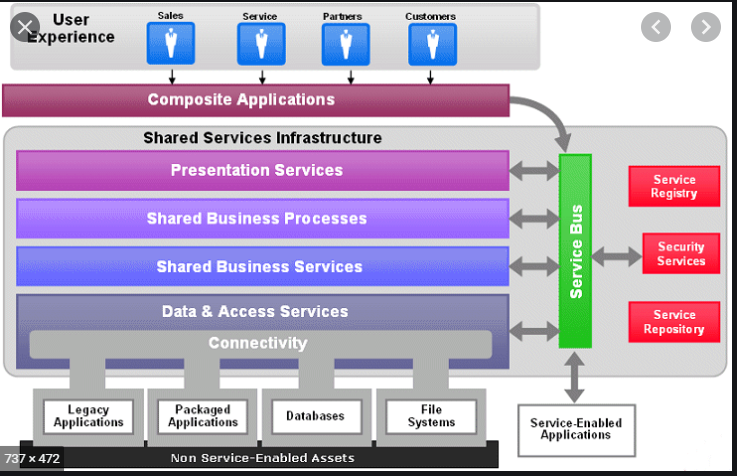
# Integration Architecture

* It is the architecture that facilitates the integration of multiple IT components.
* IT comes into play during project integration. The different functions that project can have are batch processing, file transfer, processing using queues, CRM applications, etc.
* Integration architect

## Integration Architecture Patterns

* SOA
* Microservice
* EDA
* ETL

## SOA



* Service oriented architecture
* Divide application into independent services
* Why use it: developing independent service, expose data and develop reusab;e services
* Each service is independent.
* The services are grouped according to the types of service they provide: presentations, shared business process , data and access
* The success of SOA depends in the ESB. Features of ESB are support web services, data transformation, conent based routing, enable to connect any technology in the enterprise
* All the services interact with each other via ESB
* Advantages:
  + Easy to maintain
  + Resused
  + Discoverable
  + Platform independent
  + Scalable
* Disadvantages
  + If one service occupies large bandwidth then the performance of the overall system is affected.
  + For high bandwidth message transfer it requires high speed server
  + High cost for installation

## Microservices

 It is an architectural style that structures an application as a collection of services that are

* Highly maintainable and testable
* Loosely coupled
* Independently deployable
* Organized around business capabilities
* Owned by a small team

Why SOA has ESB and microservice do not have?

## Microservice Vs SOA

**service-oriented architecture (SOA)** has an enterprise scope, while **the microservices** architecture has an application scope.

* **Communication:** In a microservices architecture, each service is developed independently, with its own communication protocol. With SOA, each service must share a common communication mechanism called an [enterprise service bus (ESB)](https://www.ibm.com/cloud/learn/esb). The ESB can become a single point of failure for the whole enterprise, and if a single service slows down, the entire system can be effected.
* **Interoperability:**In the interest of keeping things simple, microservices use lightweight messaging protocols like HTTP/REST. SOAs are more open to heterogeneous messaging protocols.
* **Service granularity:**Microservices architectures are made up of highly specialized services, each of which is designed to do one thing very well. The services that make up SOAs, on the other hand, can range from small, specialized services to enterprise-wide services.
* **Speed:** By leveraging the advantages of sharing a common architecture, SOAs simplify development and troubleshooting. However, this also tends to make SOAs operate more slowly than microservices architectures, which minimize sharing in favor of duplication.

Larger, more diverse environments lend themselves more to **service-oriented architecture (SOA**), which supports integration between heterogenous applications and messaging protocols via an enterprise-service bus (ESB). Smaller environments, including web and mobile applications, don't require such a robust communication layer and are easier to develop using a **microservices architecture**.

Difference between Mulesoft and API led connectivity

## Event Driven Architecture EDA

* Uses an event driven architecture
* Suitable with microservices
* When an action occurs an event is created and this event is then used to trigger the action
* It’s a Fire and forget methods
* It uses Pub-sub pattern: publish once and multiple users can get the access to data
* Example: change of address in insurance company by a customer
* An event-driven architecture uses events to trigger and communicate between decoupled services and is common in modern applications built with microservices.
* An event is a change in state, or an update, like an item being placed in a shopping cart on an e-commerce website.
* Events can either carry the state (the item purchased, its price, and a delivery address) or events can be identifiers (a notification that an order was shipped).
* Event-driven architectures have three key components: event producers, event routers, and event consumers. A producer publishes an event to the router, which filters and pushes the events to consumers. Producer services and consumer services are decoupled, which allows them to be scaled, updated, and deployed independently.

## Extract, Transform, Load

**ETL** is short for ***e***xtract, ***t***ransform, ***l***oad, three [database](https://www.webopedia.com/TERM/D/database.html) 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.
* **Transform** isthe process of converting the extracted data from its previous form into the form it needs to be in so that it can be placed into another database. Transformation occurs by using rules or lookup tables or by combining the data with other data.
* **Load** is the process of writing the data into the target database.

Why ETL?

* ETL makes it easier for business users to analyze and report on data relevant to their initiatives.
* it codifies and reuses processes that move data without requiring technical skills to write code or scripts

Examples

1. **Marketing data integration:** Marketing data such as web analytics, social media information or consumer data is moved into a place using ETL for further analyzation.
2. **Database Replication:** ETL tool can be used to derive data from many databases including, MySQL, PostgreSQL, and Oracle and transfer it into a cloud warehouse.

# Design Center

# API

* It is a software intermediary that allows two applications to talk to each other
* API designers can choose from a range of protocols and standards when creating a new API, depending on the type of API they are creating, and its purpose
* Types of API:
  + RestFul APIs
  + Non RestFul APIs

## RestFul APIs

* Based on REST architecture
* Support multiple data type like JSON, XML
* Follow Six principle of REST

1. Client-server:
   * separated client and server.
   * Interacts via message passing
   * loose coupling of the client and server enables each to be developed and enhanced independent of the other
   * scalability
2. Stateless:
   * the request contains all the information necessary for the server to respond
   * enables load balancing possible
3. Cacheable:

* Cache mandates that server responses be labeled as either cacheable or non-cacheable. Caching helps to mitigate some of the constraints of statelessness
* it increases the availability and reliability of the application

1. Uniform Interface
2. identification of resources;
3. manipulation of resources through representations;
4. self-descriptive messages; and,
5. hypermedia as the engine of application state (HATEOAS).
6. Layered System:

* The layered system constraint allows the architect to inject layers of service between the server and the client
* allow security policies to be enforced

1. Code on demand

allows client functionality to be extended by downloading and executing code in the form of Java applets or scripts

Examples of REST API : RAML, OAS

**Advantages of REST**

* REST can use different media type for information exchange like XML, JSON, Plain-text
* Easy to implement and takes less bandwidth
* Use any HTTP methods to pass message
* No standard service definition. If any then it human readable

**Disadvantages of REST**

* It is less secure than SOAP. It uses SSL and HTTPS
* chattiness.

### RAML - RESTful API Modeling Language

* is a [YAML](https://en.wikipedia.org/wiki/YAML)-based language for describing [RESTful](https://en.wikipedia.org/wiki/RESTful) [APIs](https://en.wikipedia.org/wiki/API).
* It provides all the information necessary to describe RESTful APIs.
* It can be used in a multitude of ways: to implement interactive API consoles, generate documentation, describing an API you are planning to build, and more.

Components :

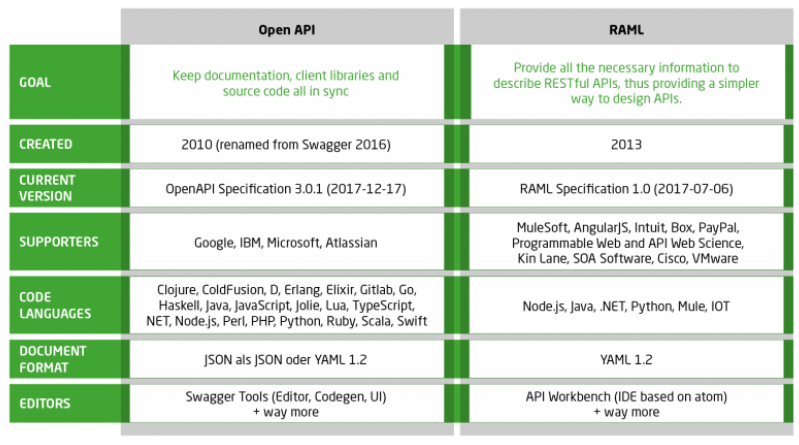
* + Root
  + Security
  + Data types
  + Methods
  + Query Parameter
  + Request
  + Response

### OAS - Open API

* It was designed to solve the need for keeping the API design and documentation in sync.
* It is a type of framework that was designed to describe, produce, visualize, and consume RESTful web services
* written in JSON or YAML
* An API defined by the OpenAPI (formerly known as Swagger) specification can be divided into 3 main sections –

1. Meta information: contains OAS version, API version, Title, Security
2. Path items (endpoints): Parameters, Request bodies, and Responses
3. Reusable components: Schemas (data models), Parameters, Responses, and Other components

### OPEN API Vs RAML



## Non RestFul APIs

APIs that do not follow six REST principle are non restful APIs

### RPC- Remote Procedure Call

* An API is built by defining public methods;
* then, the methods are called with arguments.
* RPC is just a bunch of functions, but in the context of an HTTP API, that entails putting the method in the URL and the arguments in the query string or body.
* It use only GET and POST

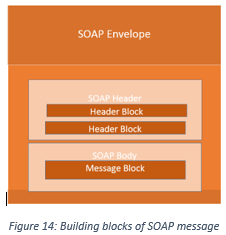
**The advantages of RPC are**: [7]

* It’s relatively easy to understand and implement. All you need to know is a function name, parameters it accepts, and rest is taken care of by RPC itself.
* It’s very efficient and high performance because of its low payload footprint on the network and a well-defined communication protocol.
* It is easier to design an API endpoint that only requires you to create a function.

**Problems include:**

* tight coupling - tend to have very high coupling to the under lying system
* low discoverability – how do I know how to start, what to call
* function explosion – create similar function which overlap and results is hard to understand
* hard to understand and maintain
* need better abstraction between the underlying system and the API

### SOAP- Simple Object Access Protocol

* It is an open-standard, XML-based messaging protocol for accessing web services over HTTP
* uses an XML data format to declare its request and response messages
* 
* SOAP API logic is written in Web Service Description Language (WSDL).
* It defines the operations available and what input/output fields to expect.
* It is also used to generate human-readable documentation for SOAP APIs

**Advantages of SOAP** [14]

* **Language- and platform-agnostic.**The built-in functionality to create web-based services allows SOAP to handle communications and make responses language- and platform-independent.
* **Bound to a variety of transport protocols.**SOAP is flexible in terms of transfer protocols to accommodate for multiple scenarios.
* **Built-in error handling.**SOAP API specification allows for returning the Retry XML message with error code and its explanation.
* **A number of security extensions.**Integrated with the WS-Security protocols, SOAP meets an enterprise-grade transaction quality. It provides privacy and integrity inside the transactions while allowing for encryption on the message level.

**Disadvantages of SOAP.**

* **XML only.**SOAP messages contain a lot of metadata and only support verbose XML structures for requests and responses.
* **Heavyweight.**Due to the large size of XML-files, SOAP services require a large bandwidth.
* **Narrowly specialized knowledge.** Building SOAP API servers requires a deep understanding of all protocols involved and their highly restricted rules.
* **Tedious message updating.**Requiring additional effort to add or remove the message properties, rigid SOAP schema slows down adoption.

### GRAPH QL

* a new API standard that provides a more efficient, powerful and flexible alternative to REST
* writing schemas using  [Schema Definition Language](https://www.prisma.io/blog/graphql-sdl-schema-definition-language-6755bcb9ce51) (SDL)
* API you simply send a single query to the GraphQL server
* The server then responds with a JSON object

**The advantages are** [7]**:**

* Fetching data with a single API call - low network overhead
* No over- and under-fetching problems
* typed schema - developers can see what the schema can query and how the data is set up there
* Autogenerating API documentation.

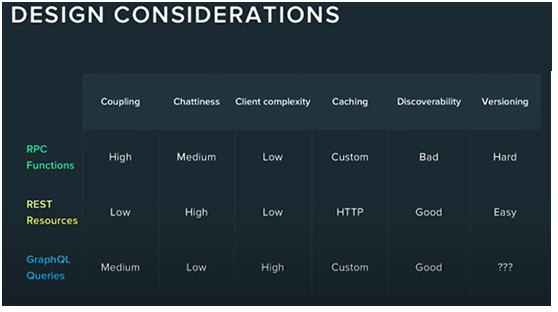
**Disadvantage:**

* complexity – While enabling clients to request exactly what they need, GraphQL query can encounter performance issues if a client asks for too many nested fields at once
* caching - GraphQL doesn’t rely on the HTTP caching methods, which enable storing the content of a request. Due to having only one endpoint with many different queries, it’s much harder to use HTTP caching with a GraphQL API
* versioning – Evolving API entails a problem of having to keep the old version around until developers make the transition to the new one
* still early in development and application.

### Async API

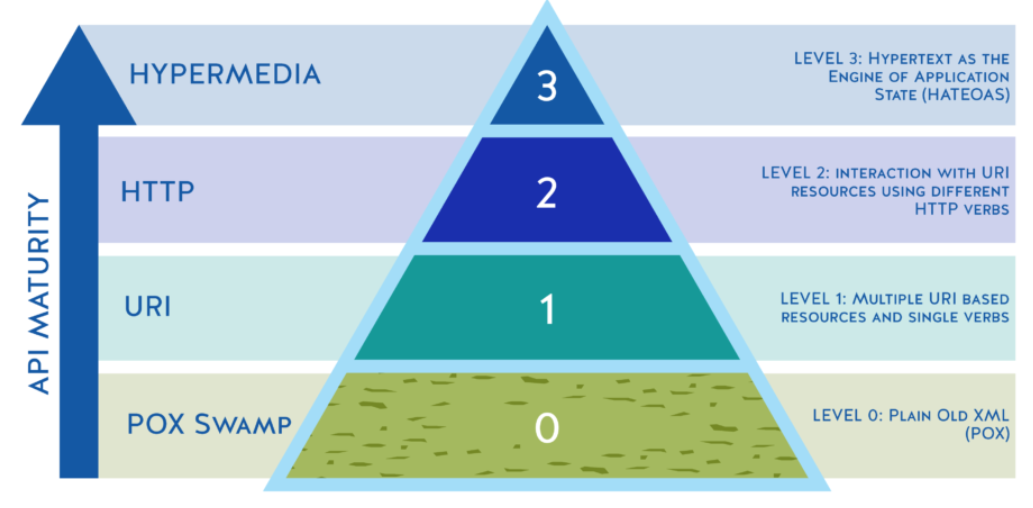
* It is event driven architecture: publisher, broker and subscriber
* APIs uses JSON and YAML language
* AsyncAPI is entirely built around the concept of message-centric API interaction, which is typically found in IoT and similar platforms
* The protocols supported by AsyncAPI are:Advanced Message Queuing Protocol (AMQP),Message Queuing Telemetry Transport (MQTT), WebSocket API, Kafka, JMS, STOMP

## Comparisons



* If the requirement is mostly action based then may be RPC is suitable. RPC style endpoints are great when you want only one job done well. This makes it useful for one or two app clients because it is a niche service. RPC endpoints can implement business logic inside the service, given that it only does one thing. This adds simplicity and clarity to the service
* If API is mostly used for CRUD operation then may be REST is more suitable. It has the highest abstraction and best modeling of the API. But it tends to be heavier on the wire and chattier – a downside if you’re working on mobile.
* SOAP is good for applications that require *formal contracts* between the API and consumer since it can enforce the use of formal contracts by using WSDL. It has built in *WS-Reliable messaging* which increase security in asynchronous execution and processing
* GraphQL server keep performance as the foremost priority. But client has to know about the data schema to query.

# Richard’S Maturity Model and Hypermedia



* + - 1. Swap of POX
* use of HTTP or RPC
* use only one URI and one POST HTTP method for all operations
  + - 1. URI
* View endpoint as a collection of resources.
* uses **divide and conquer method**
* Break huge services into multiple endpoints
* But uses only HTTP
  + - 1. HTTP Verb
* you use the **various HTTP methods** for manipulating the resources: GET, POST, PUT , DELETE
* use the response codes of your application protocol: ON success 200 status code
  + - 1. Hypermedia controls
* server r**esponds** not only with **requested data** or state of the resource but also **the links to actions** he can take on the resource

# Visual Editor and Design Center

* Anypoint platform design center allows to desing APIs, implement flow integration and build connectors
* API Designer is an web based interface for designing, documenting, testing and sharing API specification and fragments
* Two method to create API
  + Visual Editor
  + Text Editor

Visual editor

Allows

* To create API specification
* Support XML and JSON
* Enables to create resources, datatypes, security schema
* Publish to exchange
* Add documentation

Cannot do

* Create fragments
* Cannot switch back to visual editor

Text Editor

# API Fragments

* An **API specification** provides a broad understanding of how an **API** behaves
* API fragments are reusable component of RAML to make the design and build of a reusable API even quicker and easier
* **Data Types**, a concise and versatile way to describe and validate data inside your API definition.
* Traits is like function and is used to define common attributes for HTTP method (GET, PUT, POST, PATCH, DELETE, etc) such as whether or not they are filterable, searchable, or pageable. To use a trait we use a keyword **is**
* Libraries provide the ability to bring in pre-defined sets of data-types, resourceTypes, traits, security schemas, and reusable assets - all in a namespaced environment

# Data Modeling

Data modeling is a process of describing the structure, association and constraints related to the available data and then coding these rules into a resusable standards

Two data model

* + - 1. Canonical Data Model
      2. Bounded Context

Canonical Data Model

* + It is type of data model that aims to present data entities and relationships in the simplest possible form in order to integrate process across various system and database
  + Purpose: create a common definition for entire enterprise. This allows for smooth integration between the system
  + Idea is to have a common data model that all the system should understand and when the system wants to interact the data transformation is done common language
  + Adv:
    1. Require few translation compared to point to point mapping
    2. Easy to scale and maintain

Bounded Context

* + Uses the concept of DDD
  + In this a large domain is divided in to sub domains that are independent and a have oen context
  + Three concepts:
    1. Ubiquitous language: Each sub domain has its own language with which the team interact
    2. Bounded context: each bounded context will have a model specific to the context. They have their own language
       1. Unrelated context- only belong to a context
       2. Related context- that is shared between the contexts
    3. Context map: It is a pattern to identify relation based on the context

Mirror Backend System

* + Create a copy of files and folders from source and create a backup
  + Unique about it: id data is deleted from the source, it gets deleted from the back up too
  + Takes time to setup but its easy to maintain backup in long run
  + Can be backup in disk like RAID or use a server

# Connectors

* Connector is a software that provides connection between a mule flow and an external resource
* There are 4 types of connectors mulesoft provide
  + Select
  + Premium
  + Mulesoft certified
  + Community

# Connector Types

Use connectors to connect a Mule app to specific software applications, databases, and protocols.

## Select

* These are developed by mulesoft
* Are open source
* Anyone can use it but the support is provided to the Anypoint platform subscribers
* Example: File connectors, object store, database, http, email

HTTP: send and receive message using http protocol

Database: establish connection between mule and db

Email: to send and receive email messages. POP3, SMTP, IMAP

## Premium

* Developed by mulesoft
* To use it need an active cloud hub premium plan or enterprise subscription
* Example AS2, FTP, Anypoint custom metrics
  + - 1. AS2: Allows to send business transactions with added security over HTTP or HTTPS protocol. Security is provided using digital signatures
      2. FTPS: connect to file system. Add secure SSL. Default port is 21
      3. Anypoint custom metrics: enables to send the operational and business metric value from an application to an anypoint monitoring. These metrics can be visualized in real time in custom dashboard

## Mulesoft certified

* + Are developed by mulesoft partners and developer community
  + To get the support for these connectors one need to contact the mulesoft partners
  + These are reviewed by mulesoft
  + There are three types:
    1. Google analytics connectors: used to integrate google analytics data with other business applications. To use this it can be downloaded from mule exchange. We need to set parameters like average download time, page load time, page view, etc. Theses helps to analysis customer needs and decision making
    2. Atlassian Jira REST connectors: allows businesses to synchronize data and automate processes between Jira and third party collaboration, mobile, and social applications such as GitHub, Clarizen, Salesforce or Desk.com
    3. Microsoft office 365 excel connectors: enables you to seamlessly integrate with Microsoft Office 365 instance. Allows to read, modify workbook

## Community

* MuleSoft or members of the MuleSoft community write and maintain the Community connectors
* are generally open-source and free to use
* It includes SLACK connectors and HubSpot Connector
  + - 1. Slack Connectors
  + Using this connector, businesses can create instant connectivity to popular collaboration, mobile, and social applications to streamline connectivity and integrate business processes.
  + Slack Connector is an easy and fast way to integrate with your organization’s team chats, create notifications, automate responses, and much more

1. HubSpot Connectors
   * enables instant API connectivity to numerous HubSpot APIs
   * allows users to interface with HubSpot APIs to perform key functions without having to directly connect to the HubSpot platform.
   * It includes a sequal engine that simplifies data connectivity
   * This enables user to perform complex data manipulation

## Custom connectors

* Custom connectors are the connectors created by the mulesoft developers community.
* We can develop our own connector using the new Mule SDK platform for Mule Runtime 4.

Why connectors?

* Sometime we **do not want to expose the services** than we use the connecter
* It hides the implementation detail from the user. User only need to know the required input and expected output
* An application can connect to another application multiple times. SO when we create a connector a single instance of the connector is created to the application. Every request goes through that channel. **Few number of communication instances** is created
* It is **easy to maintain**
* We can customize the error response in the connectors
* It allows to define the threads that connectors can use