



Introduction to reliable data architectures



L5 Data Engineer Higher Apprenticeship
Module 1 / 12 (“Data Fundamentals”)
Topic 4 / 4

rev. 2 (2024)

Group discussion

Your data system strategy...



John A. Zachman. Early pioneer of enterprise architecture



“Seven thousand years of history would suggest the only known strategy for addressing complexity and change is architecture.”

John Zachman

When working with data systems, what is your strategy for addressing:

Substantial increases in complexity and rapid increases in the rate of change?

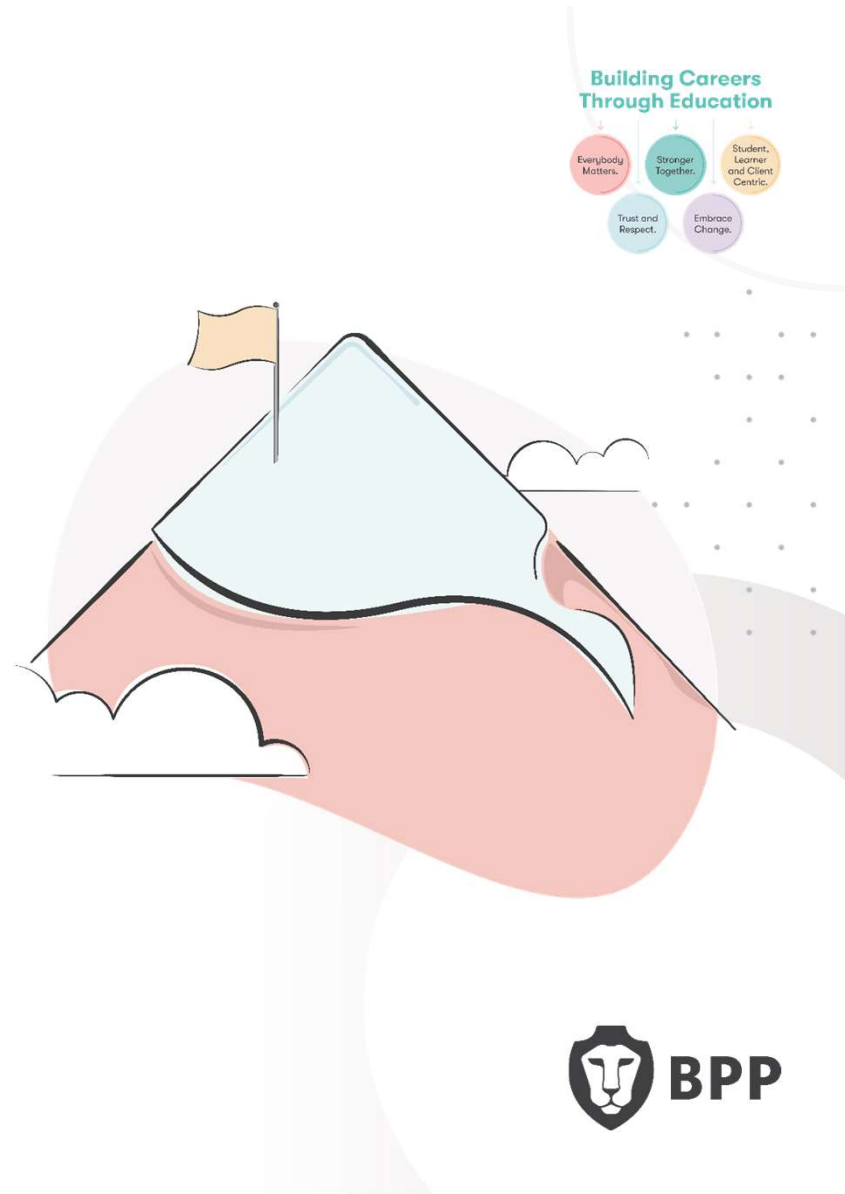
Building Careers Through Education



Learning outcomes

This webinar is designed to support the following learning outcomes:

- Model standardised big data ecosystems and architectures for enterprise data using visual approaches
- Describe the benefits of the three-layered architecture pattern for organisational data management
- Describe how the microservices architecture style enables organisations to build flexible, scalable, and resilient data products
- Define architectural governance and its role in ensuring reliability and compliance



Design vs architecture diagrams

Its easy to confuse the following two types of diagrams

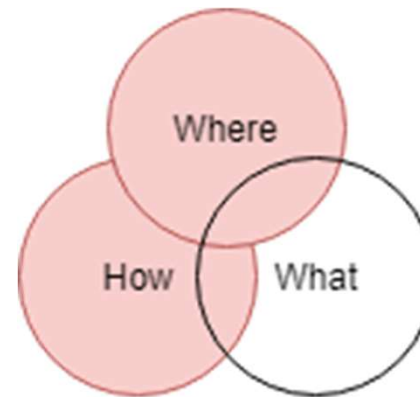
It is easy to confuse the following two types of diagrams:

An architecture diagram

- Explains the abstract idea for your system
- How stakeholders will interact with it
- The constraints of the system
- Is often drawn as layers

A design diagram

- Normally does not have stakeholders, constraints or layers
- Focuses on one part of the system and shows its building blocks



Design



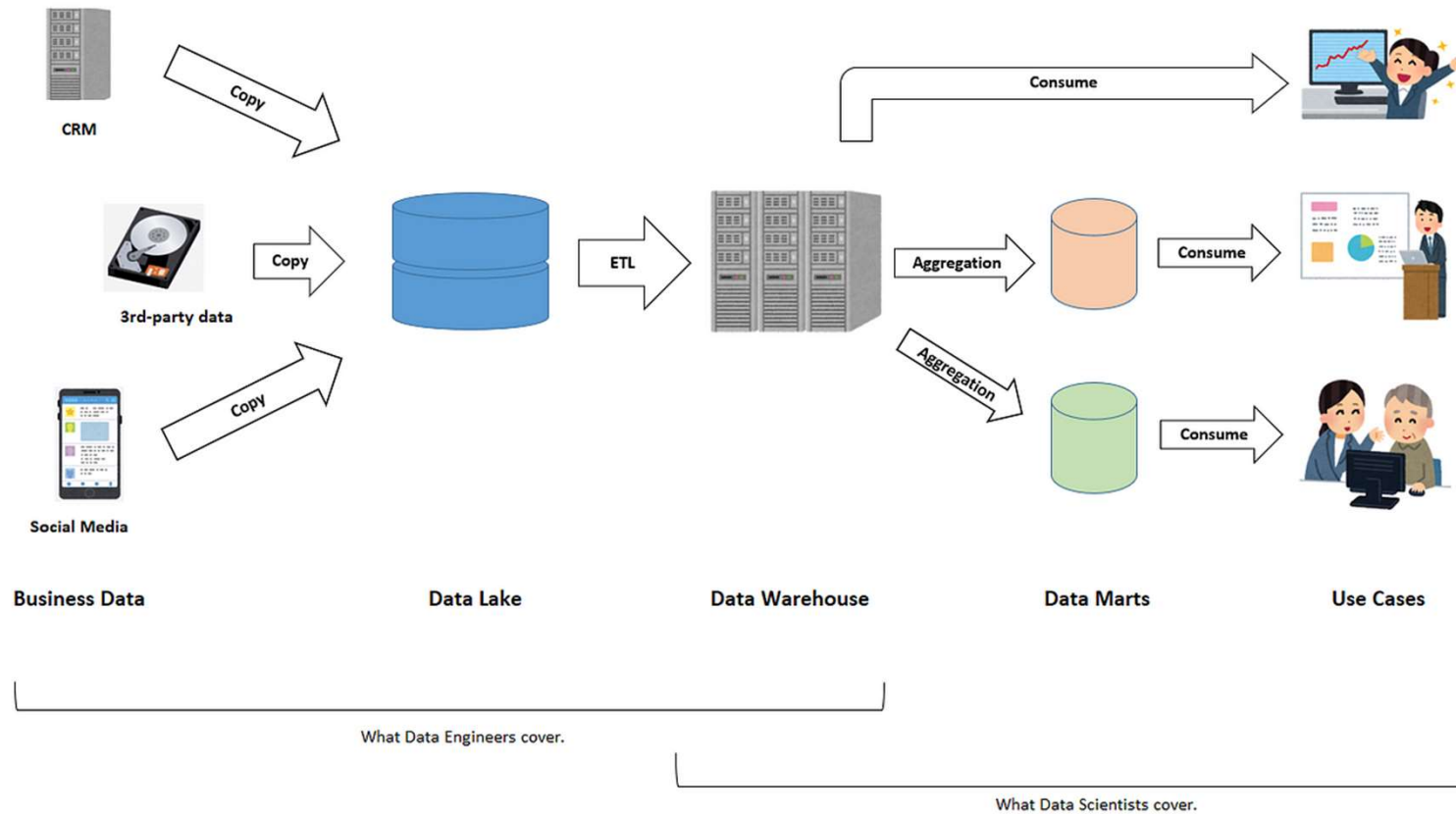
Architecture

The scrum process form start to finish

Image source: LinkedIn

A focus on architecture diagrams

What are they designed to tell use?

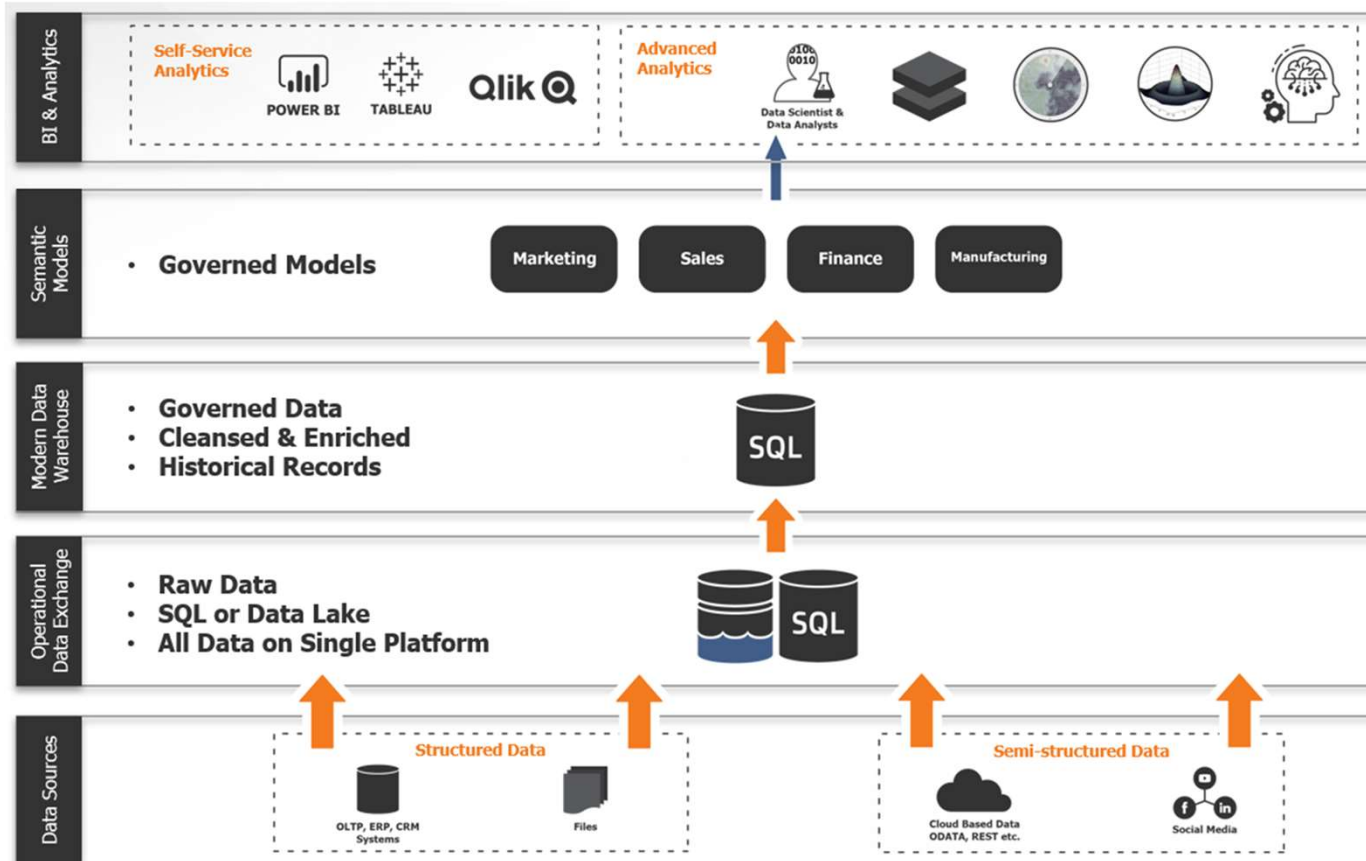


Architecture diagram



Representing architectures with layers

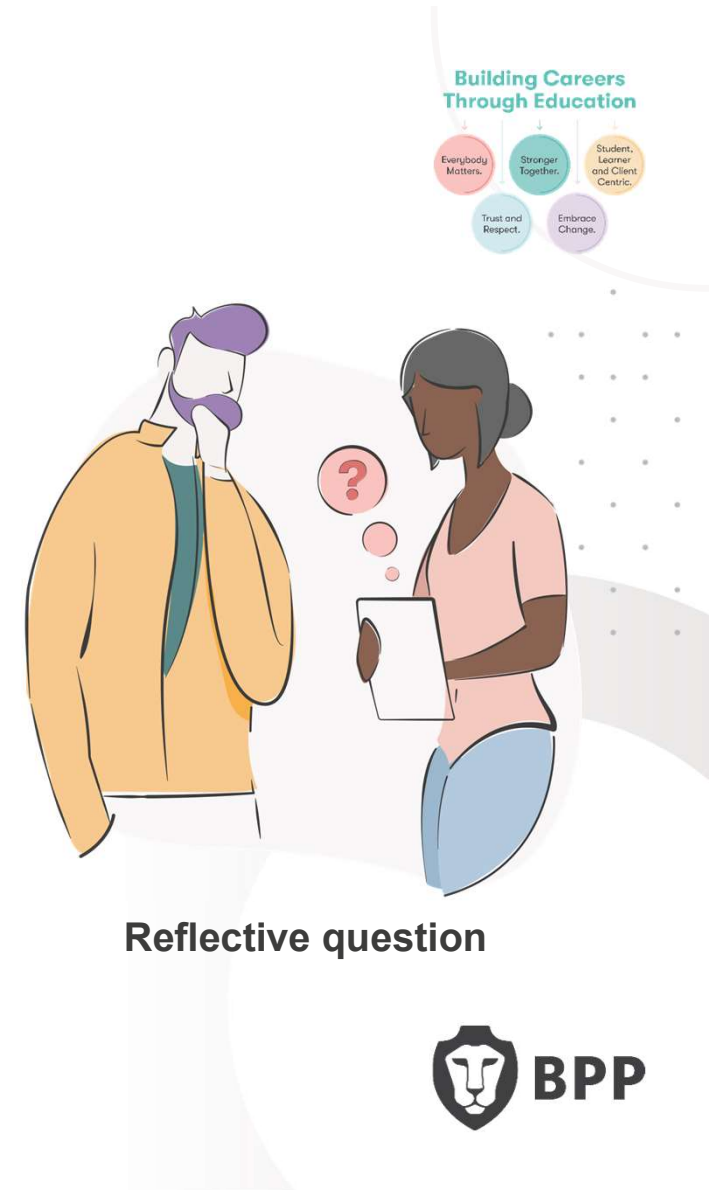
Enhancing system comprehensibility, maintainability, and adaptability



Why use diagrams at all?

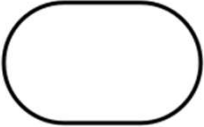


Diagrams are useful for the following reasons:

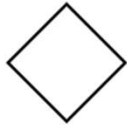


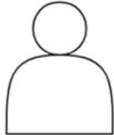
- Diagrams convey key system aspects
- No diagram is flawless; you select focal points
- Diagrams initiate discussions
- System architecture can be viewed through various lenses based on:
 - Job
 - Role
 - Expertise
 - Project stage



Exploring flowcharts

One of the most basic types of diagrams you can make...

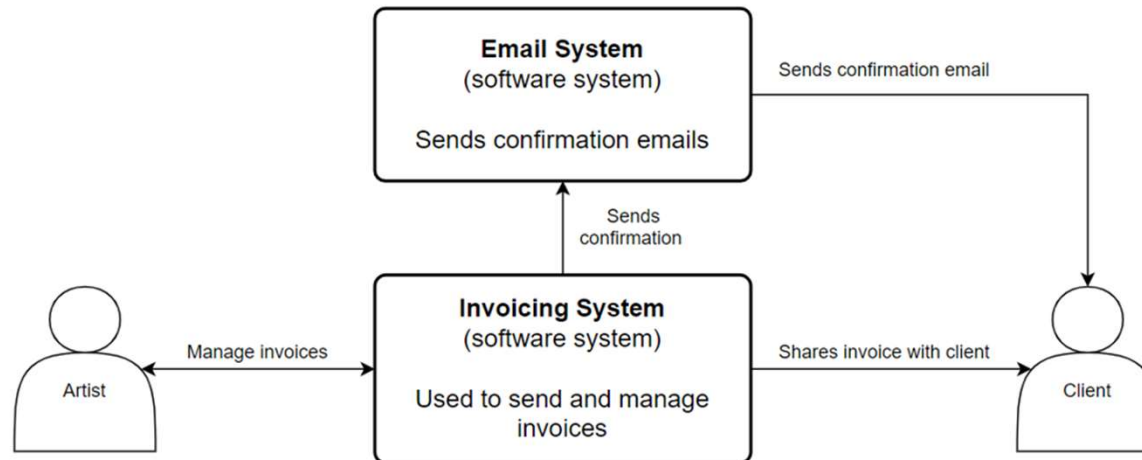
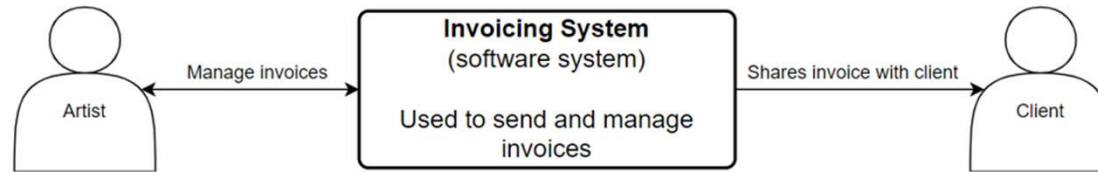
Start / End		Used to represent the starting point or terminal point of a flowchart
Flow lines		Connects components in a flowchart and indicates flow direction
Input / Output		Represents information or data that is transmitted or received

Decision		Represents checkpoints to evaluate conditions for making decisions
Process		Represents processes (e.g., mathematical operations)
Database		Represents databases
Person		Represents actors or users or a software system

Components of flow charts

Document the actors

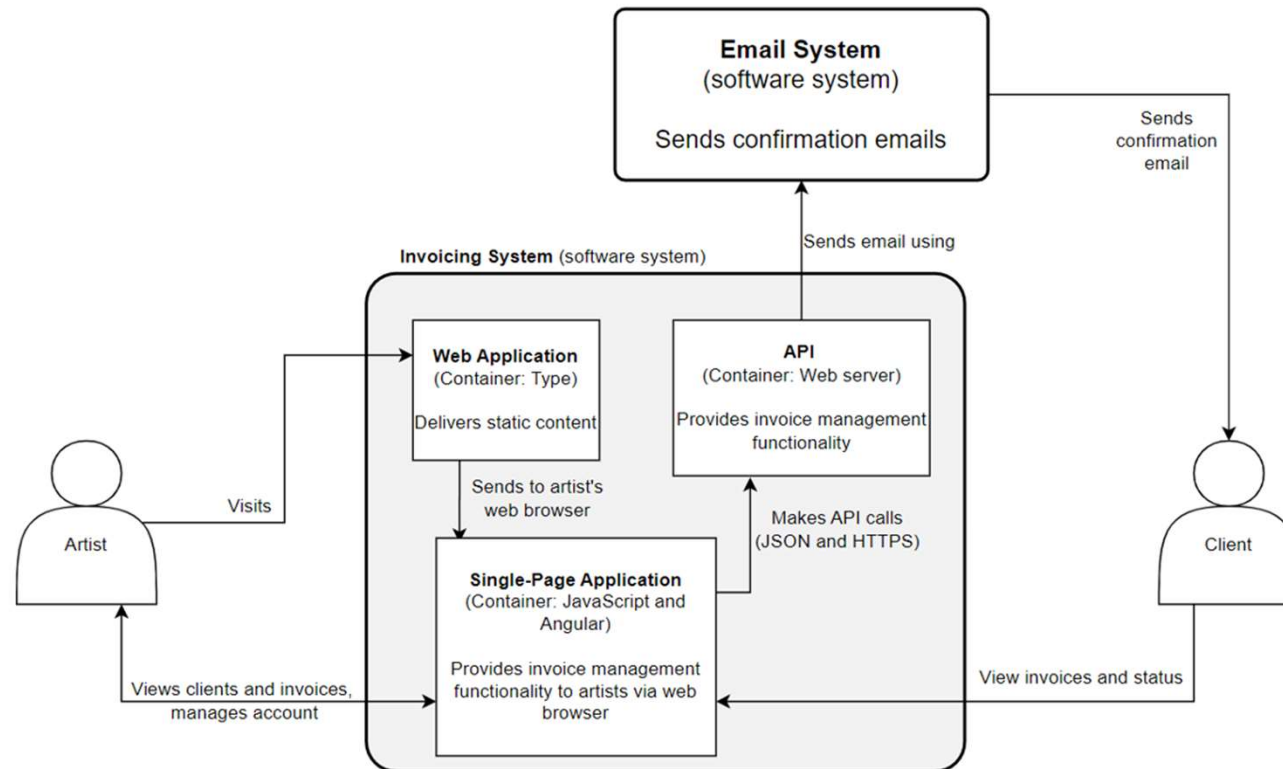
Identify actors and document external system interactions for software clarity



Now that you know who your actors are, document any **external systems** interacting with the software system

Adding containers

The benefits of adding containers in flowcharts



Improving flowchart clarity with containers for grouping and hierarchy

Misuse deployment diagram

Threat orientated diagrams

- "Misuse Deployment Diagram" identifies security threats in data architecture
- Shows how attackers exploit weaknesses
- Includes servers, databases, networks, and controls
- Aids in proactive risk mitigation
- Ensures data protection and system integrity

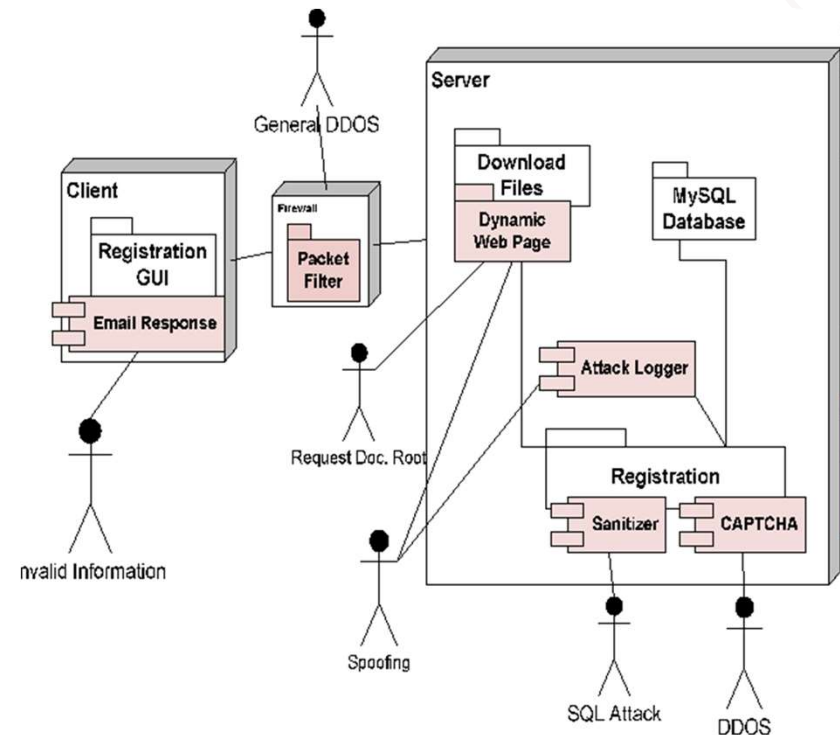


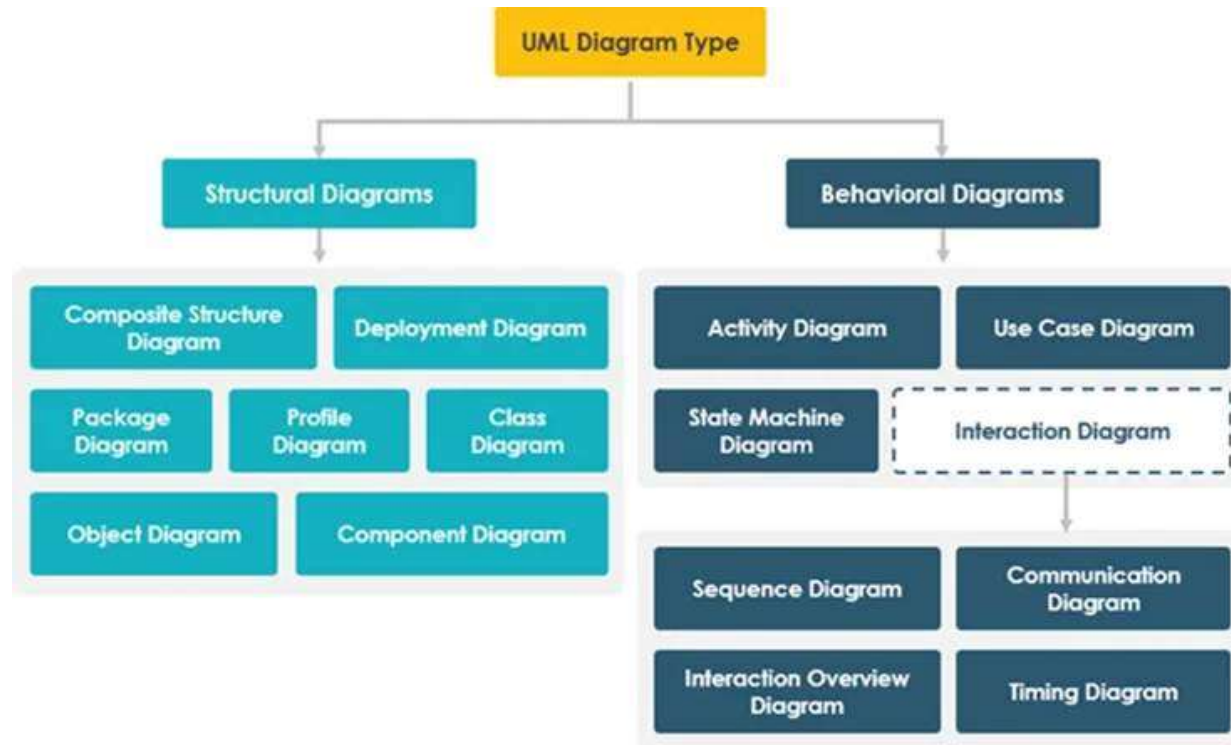
Figure 3. MDD for Web Registration

Proactively safeguarding data integrity with misuse deployment diagrams



UML component diagrams

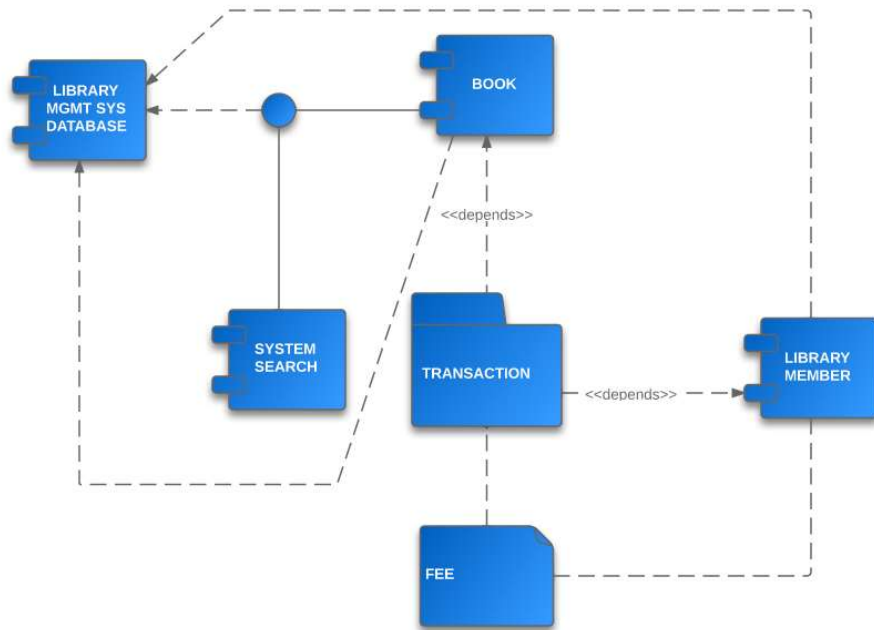
What is their purpose?



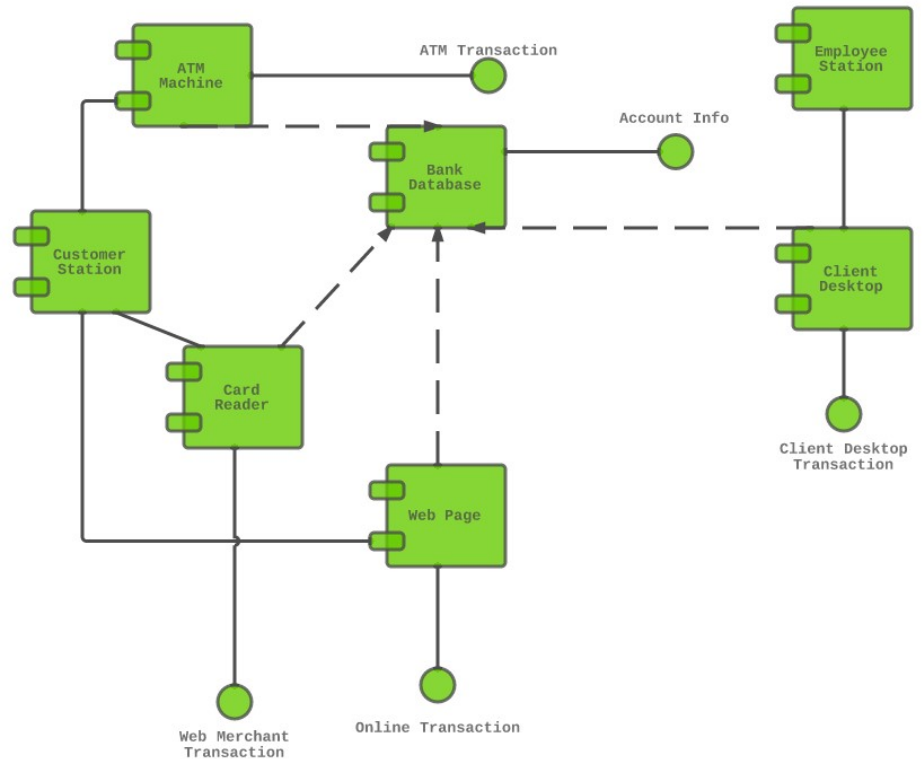
There is a wide variety of UML diagrams, and this chart shows you the different UML diagram types

System component diagrams

How do they work?



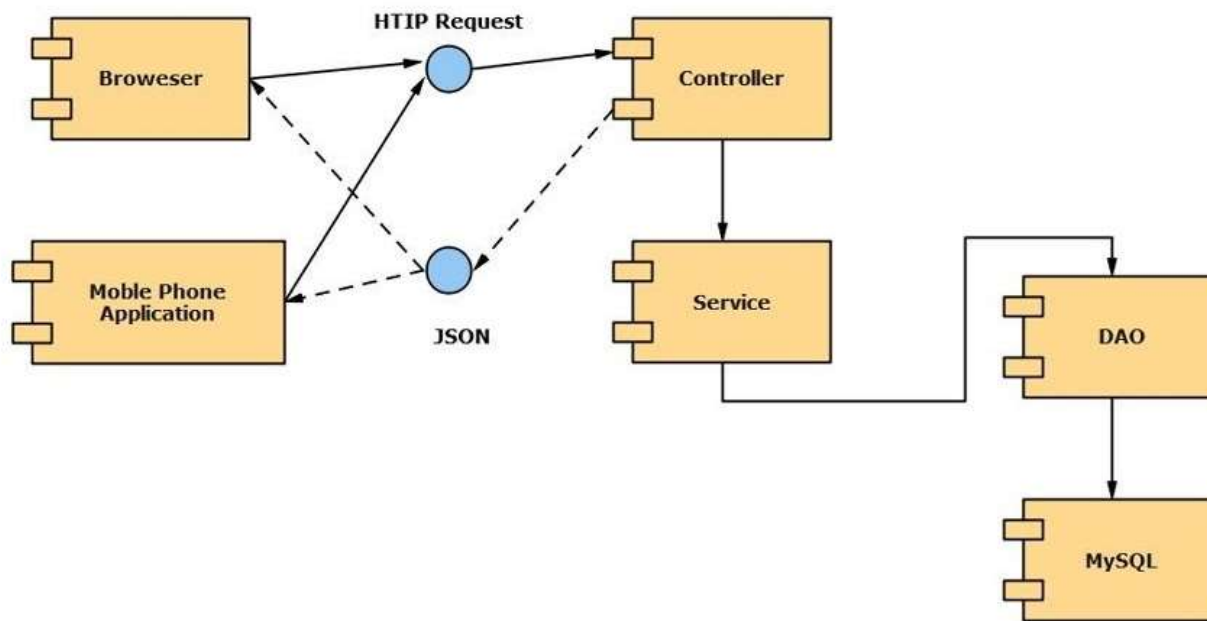
System component diagram example 1



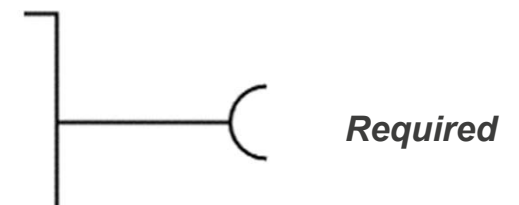
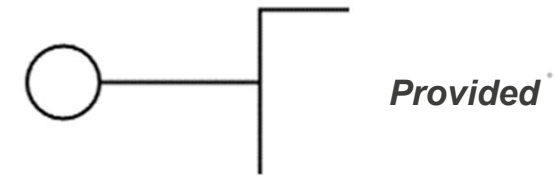
System component diagram example 2

System component diagrams

How do they work?



System component
diagram example 3



System component
diagram key

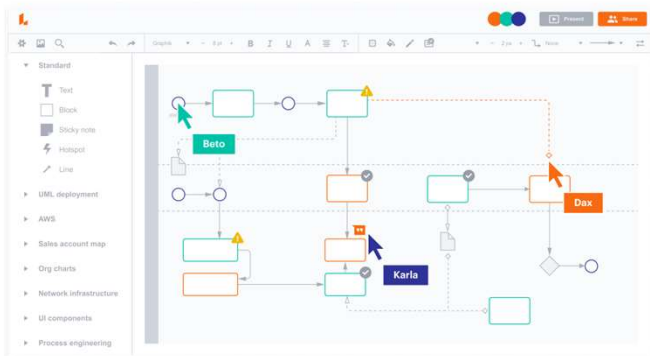
Group discussion

How to create diagrams...

Diagrams can be created using the following:

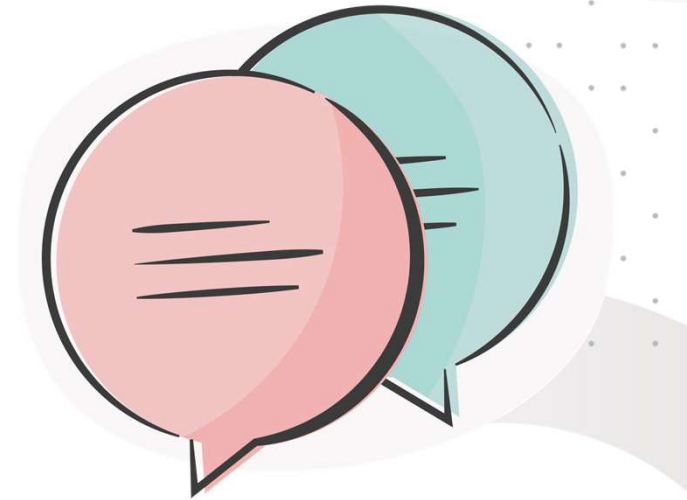
- LucidChart (online)
- Microsoft Visio (part of MS Office)
- Dia (free software)
- Draw.io (cloud based)

What do you use to create yours?



 **Lucidchart**

Building Careers
Through Education



Group discussion



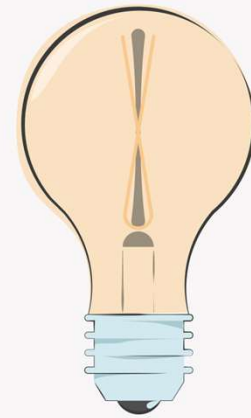
Tips for creating architecture diagrams

Remember the following...

Diagrams can be created using the following:

- Version Control your diagrams – they will evolve
- Simplify when possible, split when necessary
- Create logical groupings using polygons and colours
- Complement diagrams with descriptions
- Avoid too many acronyms
- You will normally have to explain the symbols you use

Building Careers
Through Education

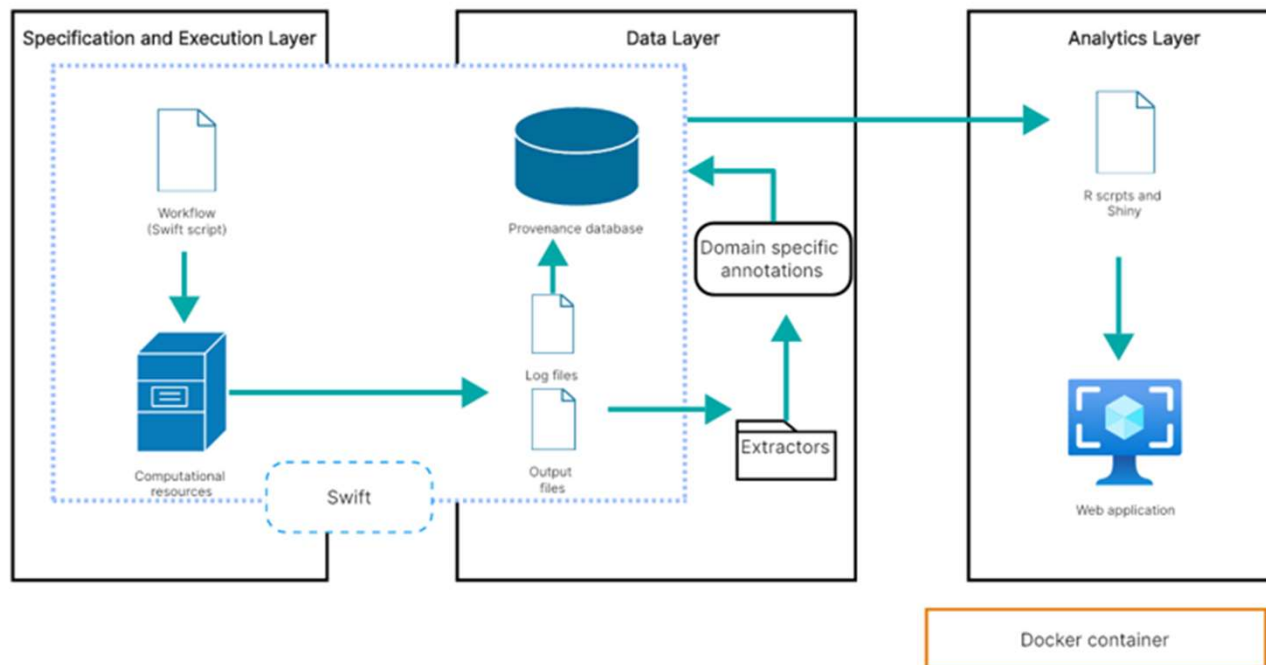


Mike's handy hints!



Applying lenses, layers and chunks

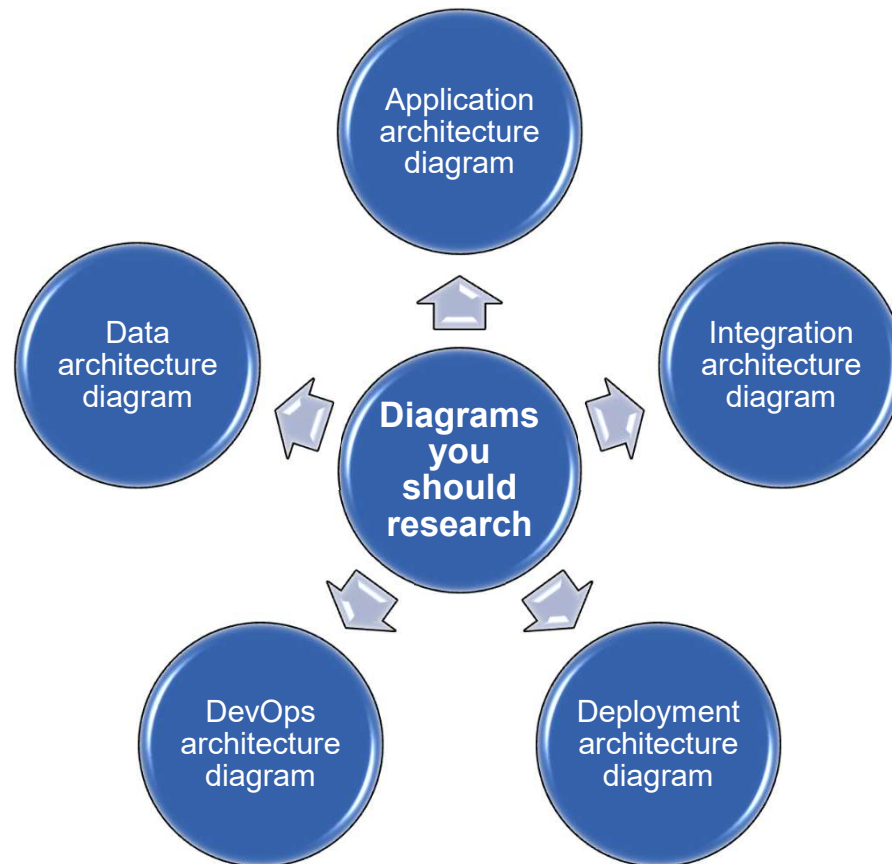
Exploring system architecture: Choosing lenses and layers



An example of conceptual architecture diagram

Other types of diagrams

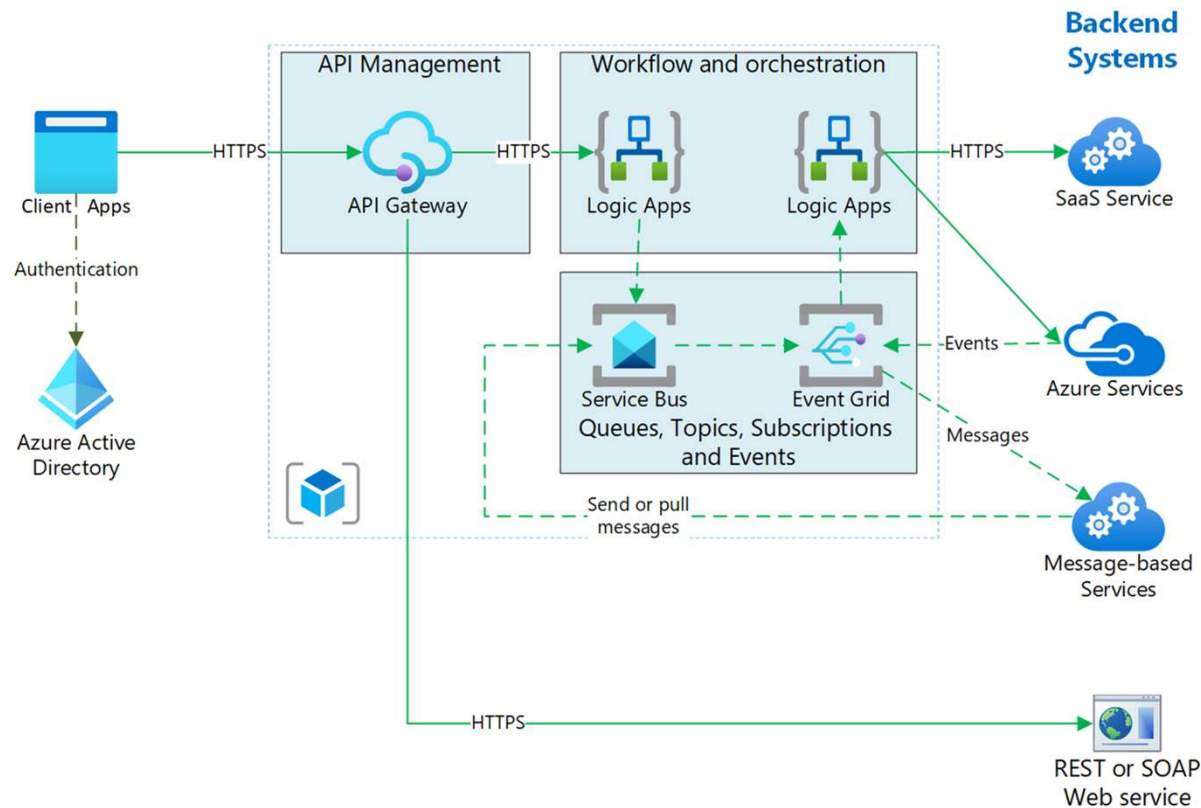
You should research the following...



Research suggestions!

Integration architecture diagram

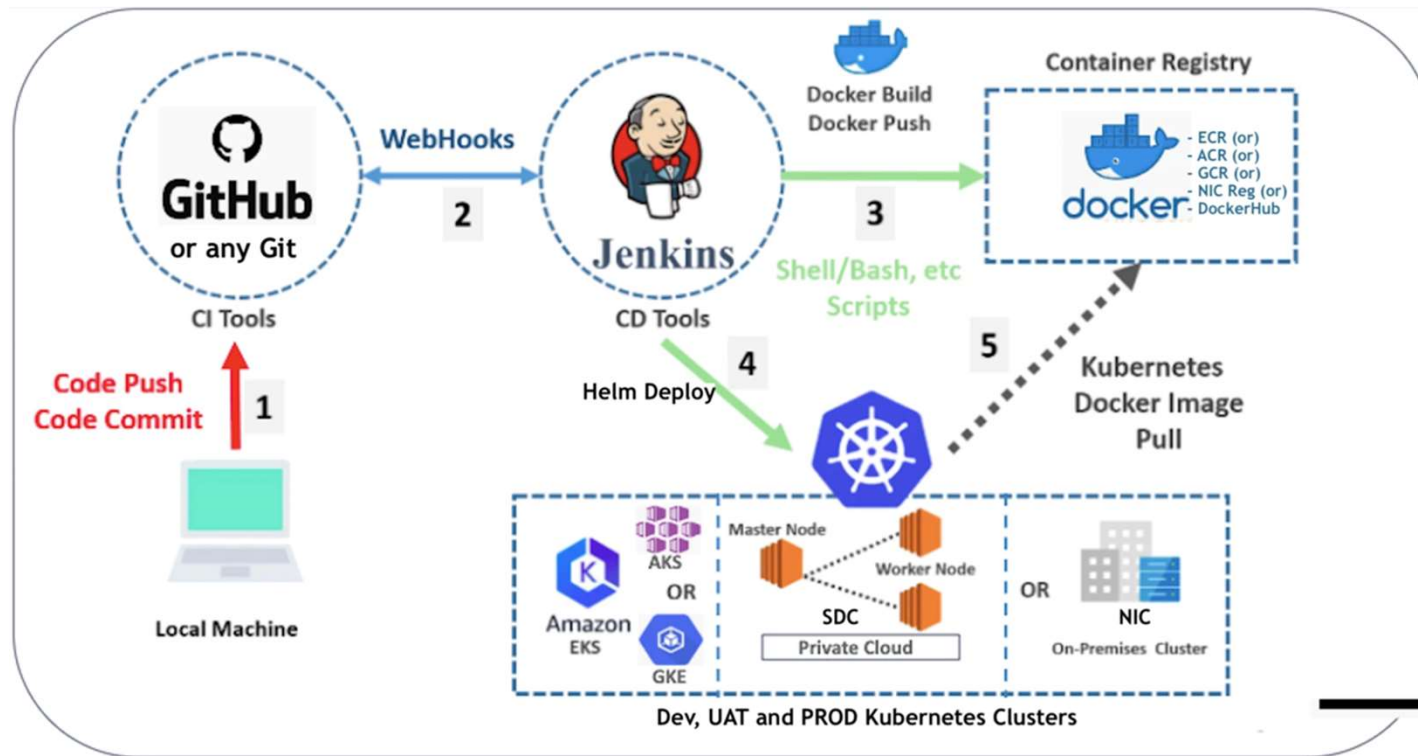
Exploring system architecture: Choosing lenses and layers



An example of conceptual architecture diagram

Deployment architecture diagram

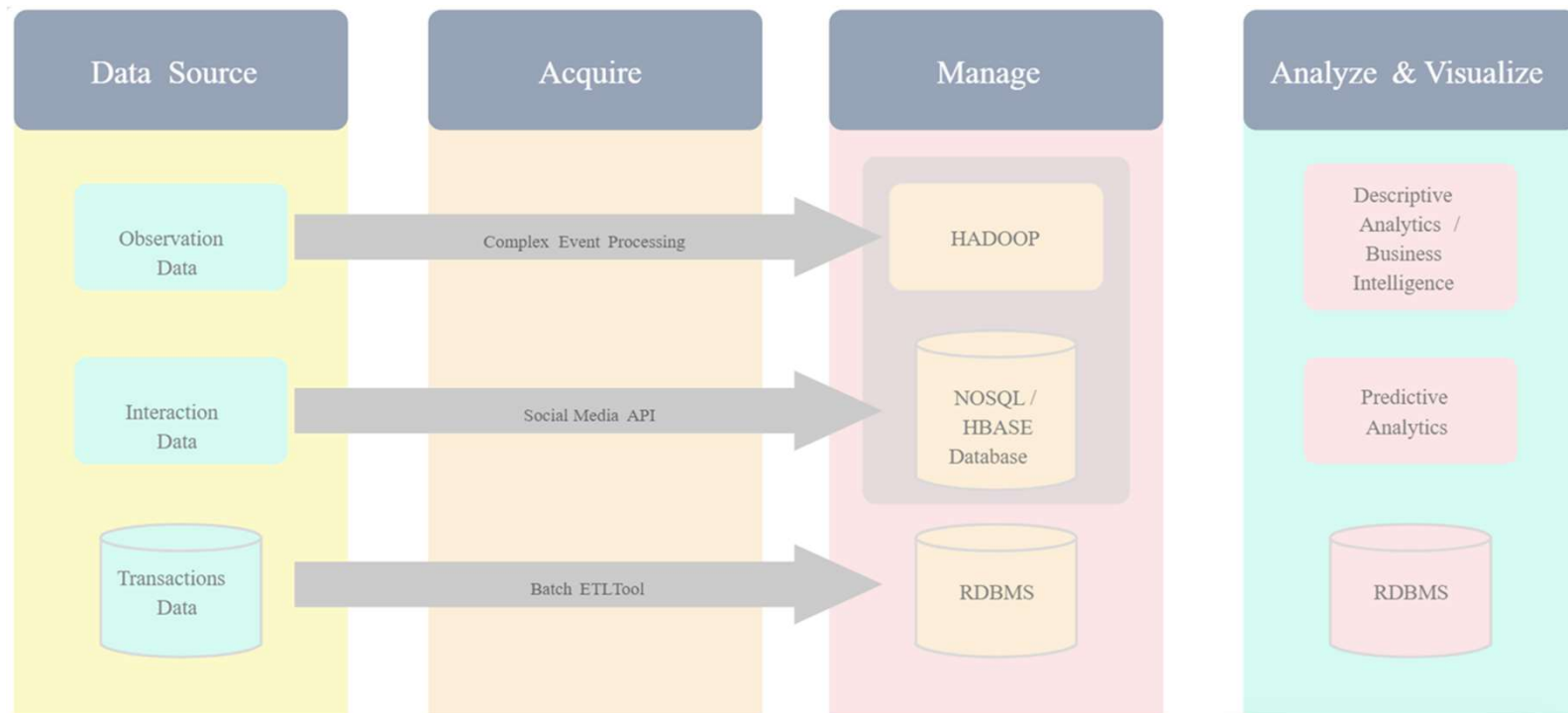
Exploring system architecture: Choosing lenses and layers



Visualising deployment architecture for scalability and security

Data architecture diagram

Understanding data architecture diagrams



Understanding the flow from acquisition to analysis in data architecture diagrams

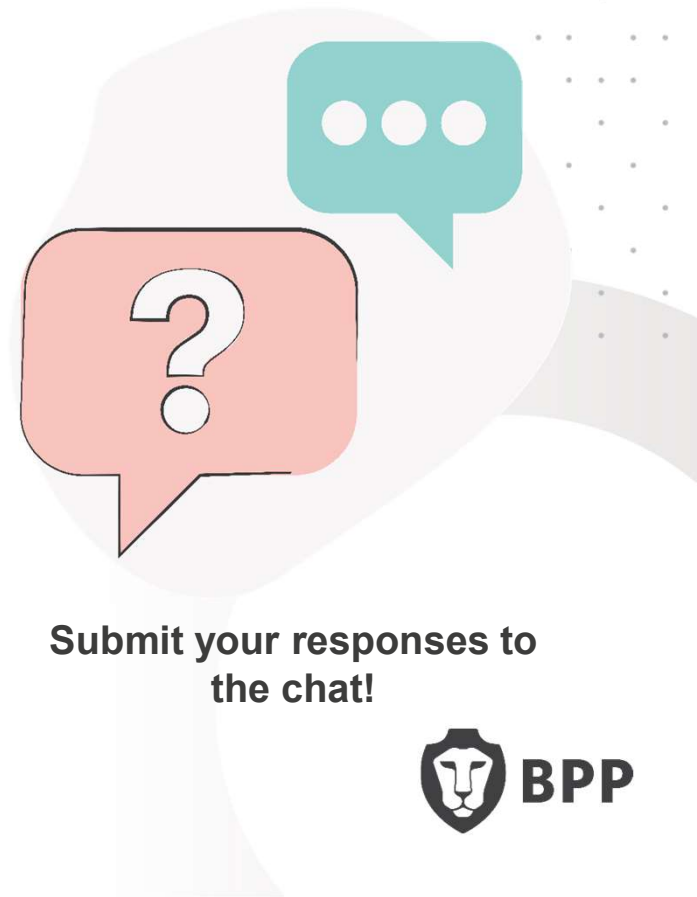
Knowledge Check Poll

What is the primary purpose of UML component diagrams?

- A. To show the physical deployment of the system
- B. To represent the system's functional requirements
- C. To illustrate the relationship between different components in a system
- D. To describe the system's user interface

Feedback: C – The purpose of a UML component diagram is to show the relationship between different components in a system

Building Careers
Through Education



Collaborate task

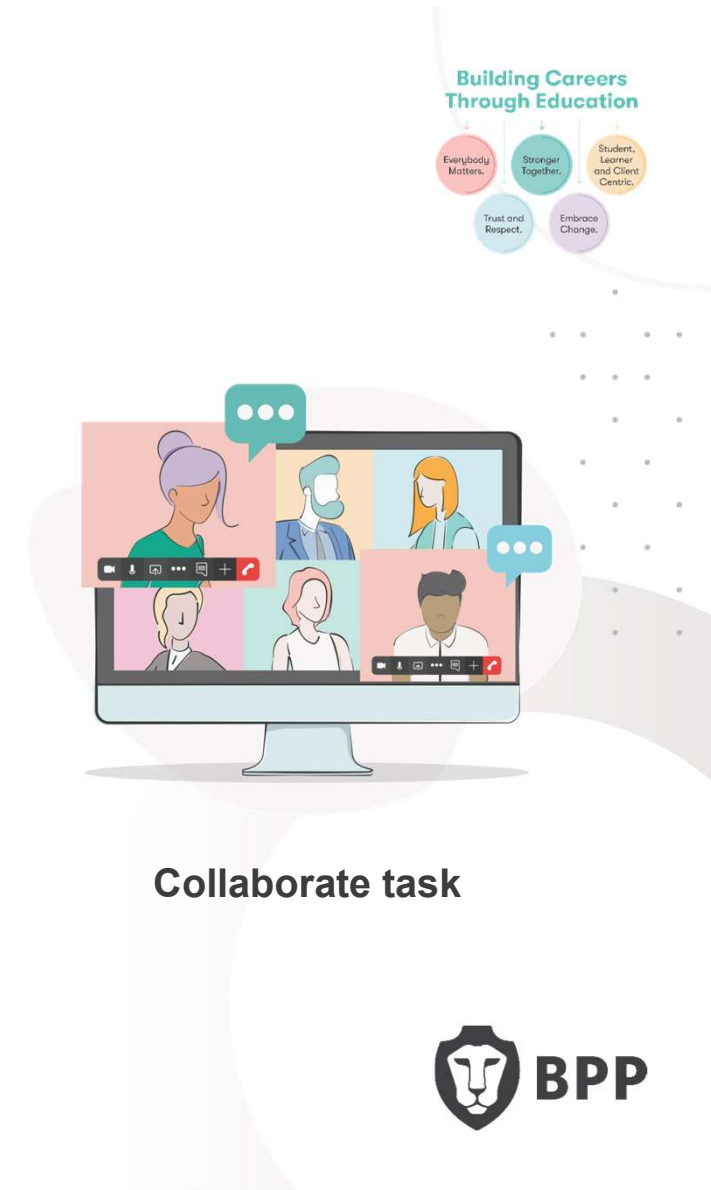
Scenario:

NextGen Fitness provides personalised fitness tracking and coaching services via a mobile app and wearable devices.

The system collects wearable data, stores it, and uses ML for coaching advice. Users can also connect with personal trainers.

The brief:

1. Create a basic flowchart/architecture diagram for NextGen Fitness
2. Include actors: Users, Personal Trainers, Wearable Devices, Mobile App, ML Algorithms
3. Include containers: UI, Data Storage, Data Processing, Personalized Coaching, Trainer Communication
4. Show data flow and interactions
5. Use diagramming tools (LucidChart, Visio, etc.).

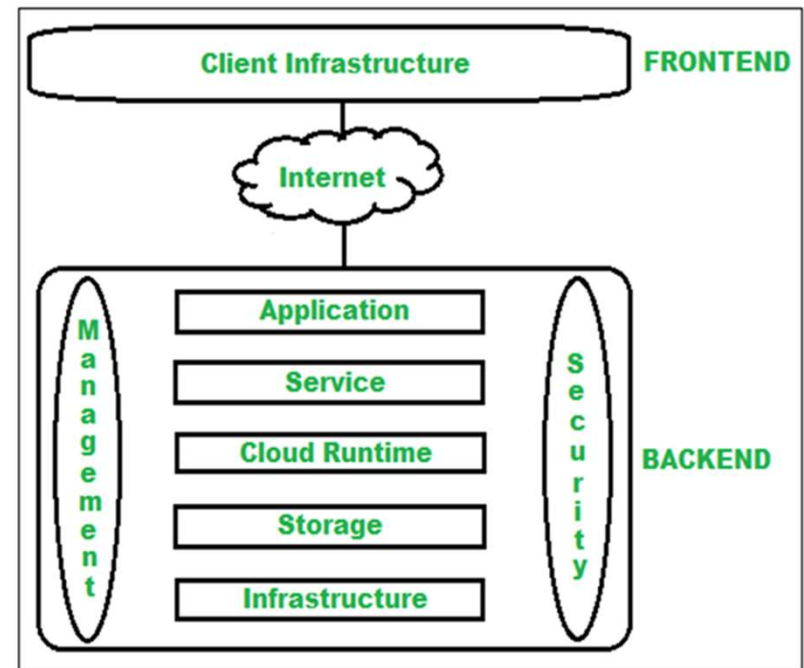


Architecture of public cloud computing

Working with public cloud computing for reliable data architectures

The following considerations are important:

- Cloud storage
- Data processing
- Data integration
- Data security and compliance
- Scalability and performance
- Cost optimisation
- Automation and infrastructure as code
- Monitoring and logging



*Unlocking the power of public cloud computing
for reliable data architectures*

Architecture problem space

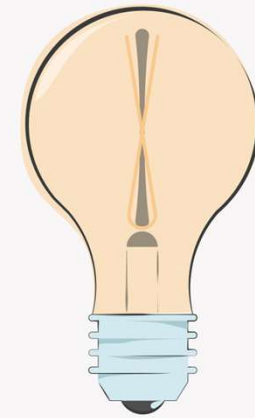
Helpful guidance...

BEFORE making decisions such as what technology is going to be used and the physical designs, you need to articulate the problem space for your architecture.

This is achieved by teams asking the following questions of the users:

- What is the current user experience or product?
- What are the problems?
- How can it be improved?
- What assumptions have you made?
- New product or user experience? How do you think your proposed design ideas support, change or extend current ways of doing things?

Building Careers
Through Education



Mike's handy hints!



Enterprise architecture

A framework

Problem

How to fully describe a project?

Difficulty

Work with an SME

1. An **Enterprise Architecture** (EA) shows project details, from business understanding to enterprise deployment.
2. An EA's artifacts includes models, documents, and specifications.
3. Example EAs include DoDAF (52 artifacts in 8 categories) the Zachman Framework (36 artifacts), and TOGAF.
4. Usually, only a subset of an EA's artifacts are created.

- Project concept
- Subject matter experts

Enterprise Architecture Process

Project details at all levels for all customers

1. Select an Enterprise Architecture
2. Decide on which elements in the EA to create
 - A minimal list of artifacts could include
 - AV-1 : Overview and Summary Information
 - AV-2 : Integrated Dictionary
 - OV-1 : High Level Operational Concept Graphic – **most common**
 - OV-2 : Operational Node Connectivity Description
 - StdV-1 Standards Profile
 - SV-1 : System Interface Description
3. Create the artifacts and review with stakeholders

The **Zachman Framework** has (example instantiations shown below)

- **6 descriptive areas**: data, function, network, people, time, motivation
- **6 perspectives**: planner, owner, designer, builder, subcontractor, enterprise
- The 36 elements are arranged in a 6-by-6 grid

TOGAF (The Open Group Architecture Framework) uses 4 architecture domains: Applications, Business, Data, and Technical

Enterprise Architecture – Example – Phone App

- Consider creating a phone application
- Use the Zachman framework to show all needed artifacts.
- The 6 perspectives (rows) can be interpreted in several different ways; three are shown.
 - For example: “Objective /Scope” / “Contextual layer” / “Role: Planner”
- The cells in the 6-by-6 grid below contain only some of the items that would be in that cell.

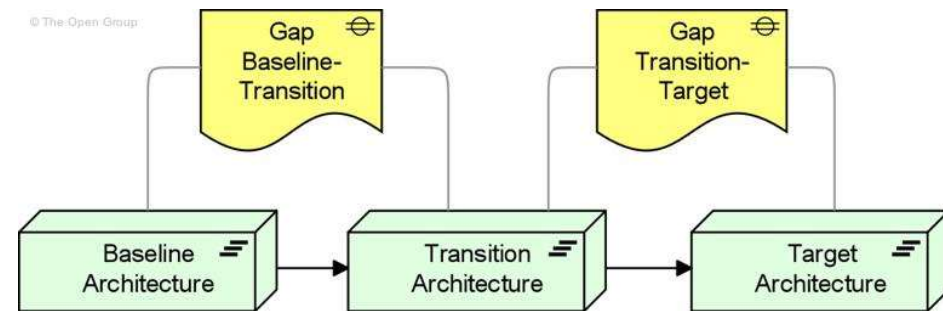
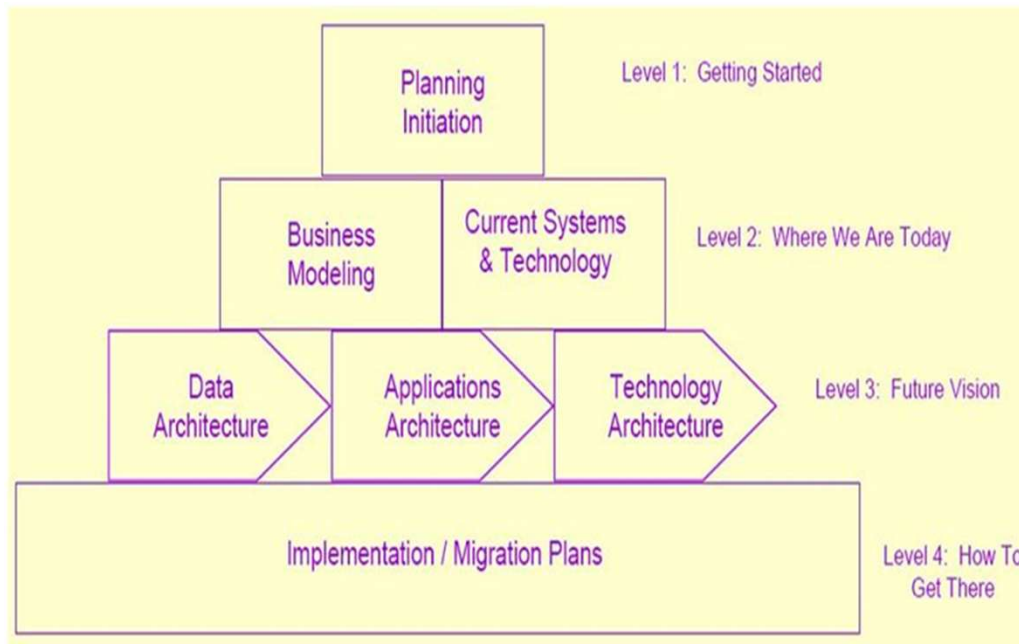
6 perspectives –
must be in this top
down order

6 descriptive areas – can be in any order

	What	How	Where	Who	When	Why
	<i>Data</i>	<i>Function</i>	<i>Network</i>	<i>People</i>	<i>Time</i>	<i>Motivation</i>
(1) Objective/Scope Contextual layer <i>Role: Planner</i>	Business vision & goals	Business processes	Business locations	Departments involved	Future products road map	User needs, app business case
(2) Enterprise model Conceptual layer <i>Role: Owner</i>	Short term goals	App financing, hiring, training	Project locations	Stakeholders buy-in plan	Product release timeline	App alignment with other offerings
(3) System logic Logical layer <i>Role: Designer, Architect, or General Manager</i>	App look and feel	System architecture (e.g., support capabilities)	System connectivity	User interface design	Master schedule	App functionality
(4) Technology model Physical model <i>Role: Builder, General Contractor, or Local Manager</i>	Platform description, wireframe model	App requirements	Technology architecture (e.g., component libraries)	Skill identification	Development milestones	Define function capabilities
(5) Detailed representation Detailed model <i>Role: Scientist, Engineer, Subcontractor, or Programmer</i>	Interface definitions, database schema, code	App design	Communications architecture	Security design	Implementation model (e.g., scrum)	Motivate team to create successful product
(6) Functioning result Enterprise release <i>Role: End user</i>	User data needs	Usage instructions	User locations (e.g., sales roll-out plan)	Market segmentation	App responsiveness	Motivation for end-users to obtain and use app

Time-based partitions

Building the foundation for scalable data architecture across architectural levels



Evolving from the current state of the system to the desired future state

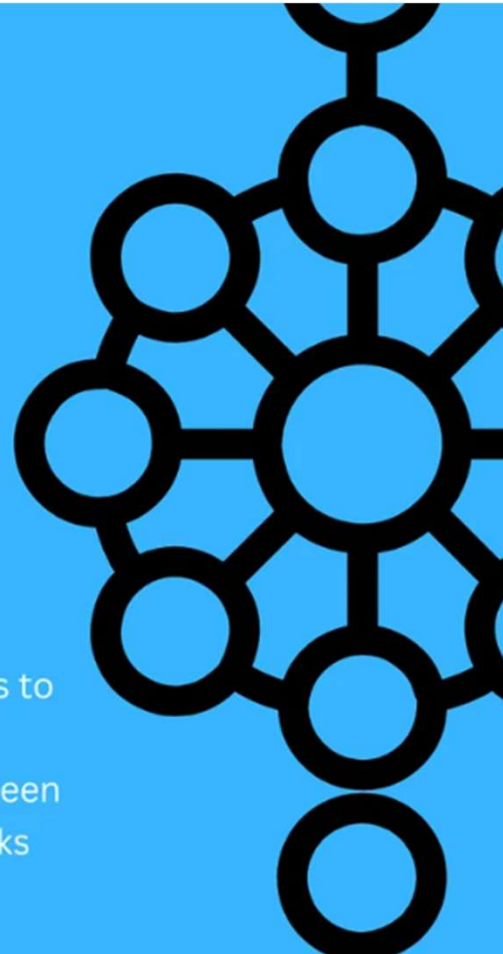
What is TOGAF?

TOGAF (The Open Group Architecture Framework), is a toolkit for planning, designing, implementing, and managing your organization's technology game plan. Think of it as a common language that brings different teams together under one roof to achieve your business goals.

Why is it useful?

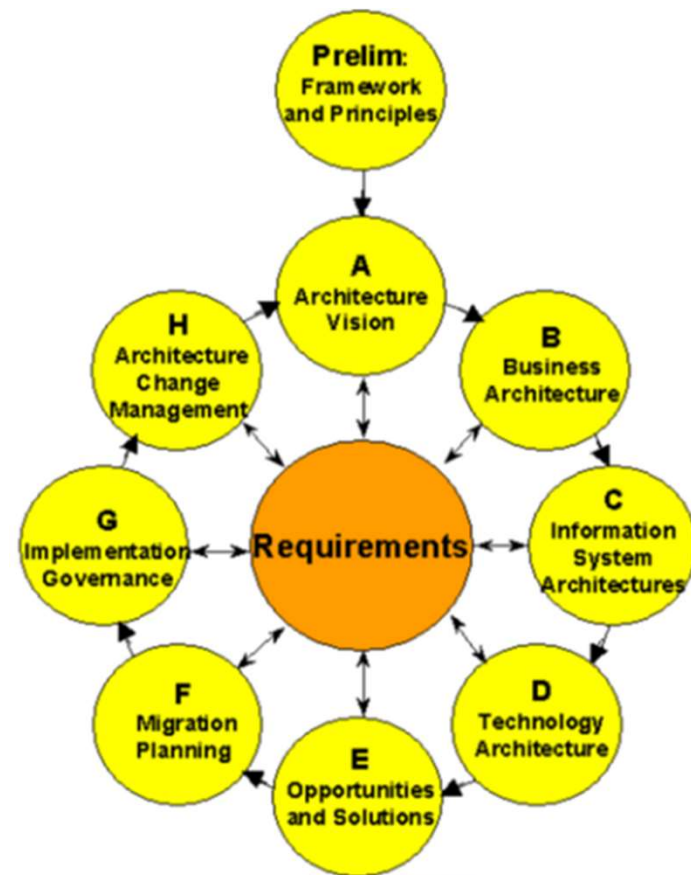
This framework is like a blueprint for organizations to align their IT strategy with business goals.

TOGAF also acts as a communication lifeline between IT pros and stakeholders, ensuring everyone speaks the same language.



The Open Group Architecture Framework

- TOGAF, developed by The Open Group, stems from the US Department of Defense's Technical Architecture
- Each version is publicly available on The Open Group's website, emphasising transparency in architectural standards development
- TOGAF breaks the ADM (Architecture Development Method) into phases, where each phase has its own:
 - Inputs
 - Objective
 - Approach
 - Steps
 - Outcomes
- For example, phase E will rely on the inputs from phases A, B, C and D



A visual representation of TOGAF Phases

TOGAF Architecture Domains

Business architecture:

strategy, governance, processes

Data architecture:

logical and physical data assets and data management resources

Applications architecture:

blueprints for individual systems to be deployed and their interactions

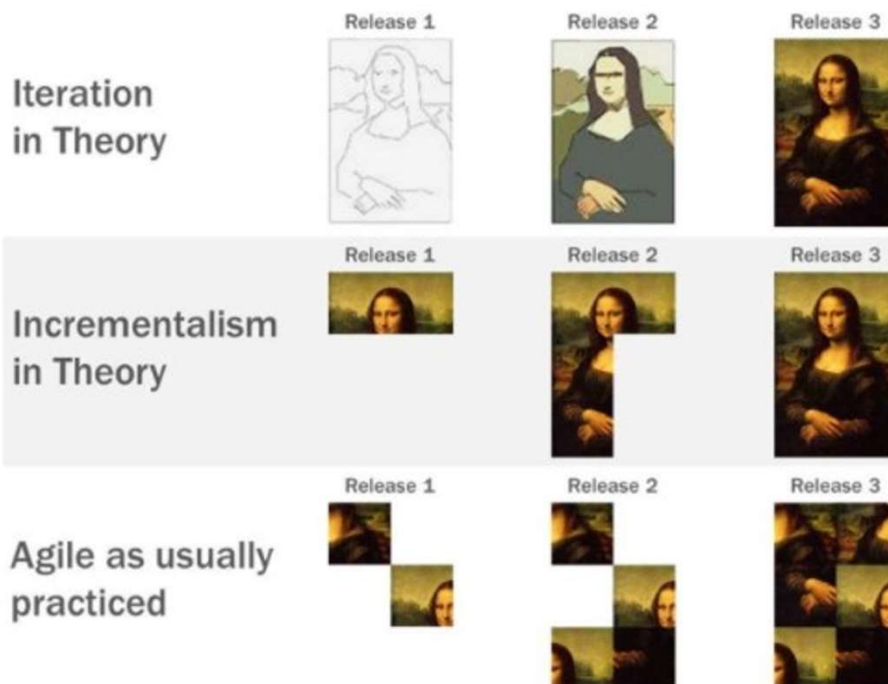
Technology architecture:

hardware, software and network infrastructure



The Open Group Architecture Framework

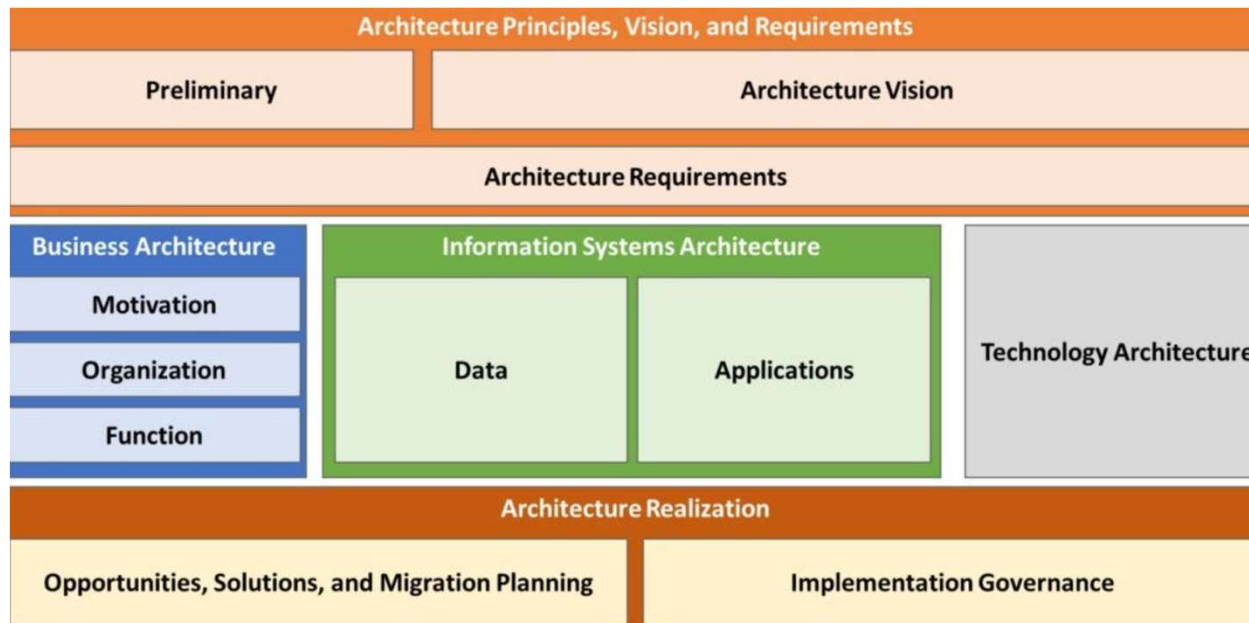
Is it relevant in an Agile world?



- ✓ TOGAF supports agility
- ✓ TOGAF is iterative
- ✓ Just another style of architecture

Iterative work still requires a vision to follow!

TOGAF – from vision to realisation



Architecture is a complex beast and stakeholders may only have interest in certain aspects of it based on their role/responsibilities. TOGAF uses the concepts of **Concerns**, **Viewpoints**, **Views** and **Artifacts** to support clear communication of the architecture



Stakeholder Concern

Represents the interests, objectives and goals of the stakeholder with interest in the system



Architecture Viewpoint

Is the framework that guides creation of views. Viewpoints can be used across projects and are not specific to a piece of work



Architecture View

An artifact or collection of artifacts that address stakeholders concerns by using the Architecture Viewpoint as a framework for their creation



Artifacts

These may be Catalogs, Matrixes or diagrams that collectively can “paint a picture” or architecture view of the system



Stakeholder

A business stakeholder may have concerns related to cost and revenue, while an IT stakeholder may be concerned with system performance and scalability.



Architecture

For a stakeholder that has concerns related to the security aspects of the project the viewpoint would describe the concerns and specify the artifacts that the view may include such as:

- Data security diagram
- Risk analysis
- Data/Information handling business rules



Architecture View

The view would be a combination of the actual/tangible artifacts that were described in the Viewpoint

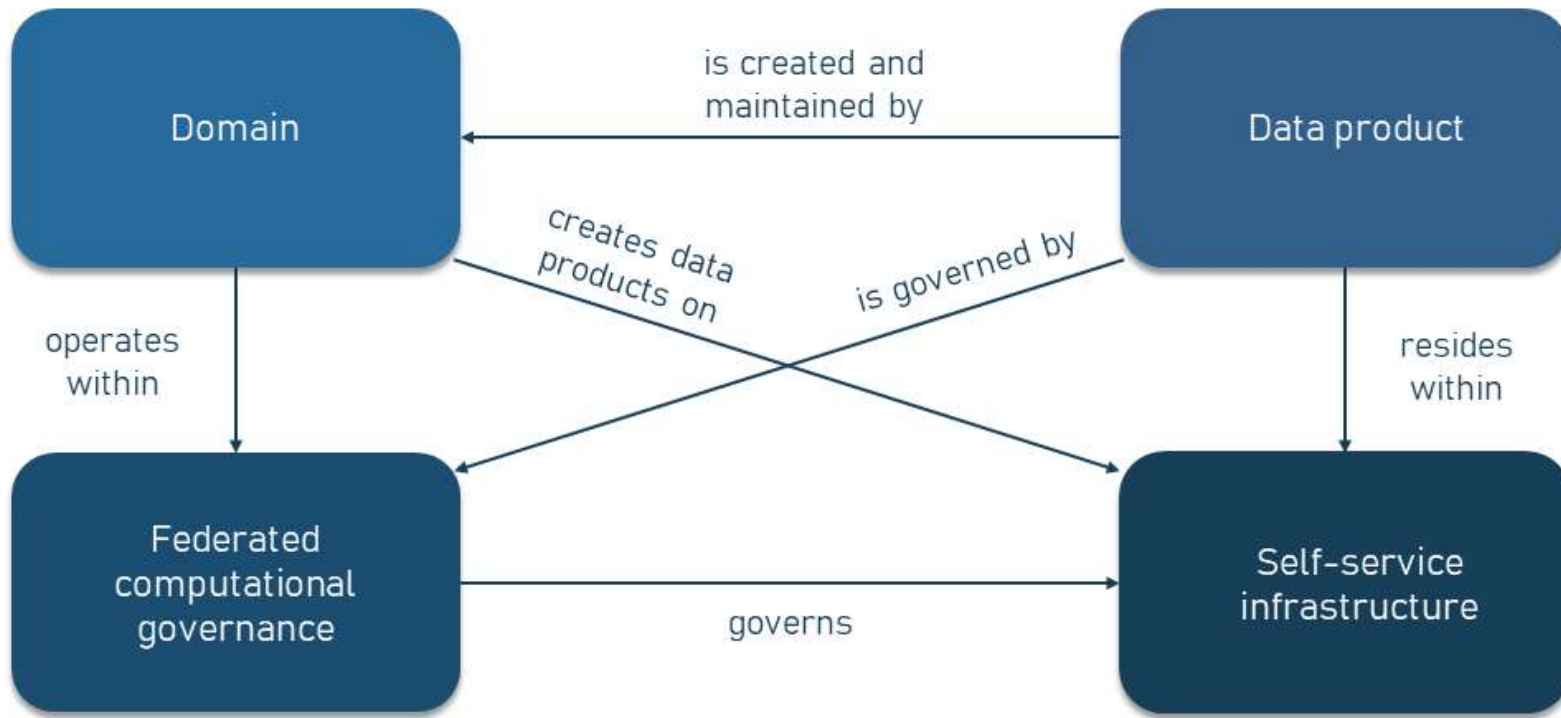


Artifacts

Each of the following is an “artifact” that collectively paints a “view”

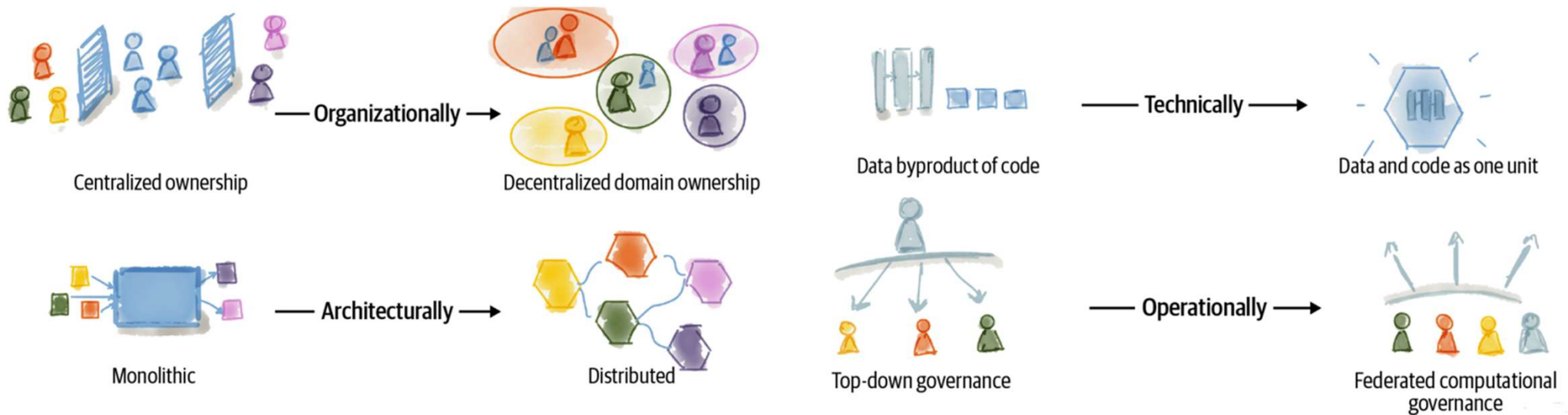
- Data security diagram
- Risk analysis
- Data/Information handling business rules

Data Mesh



Data Mesh

Embracing data decentralisation



Embracing data decentralisation with data mesh: Shifting ownership, architecture, technology, and operations

Practical research activity

In your breakout rooms read the following 5 scenarios of large-scale data quality issues impacting big businesses negatively:

<https://www.montecarlodata.com/blog-bad-data-quality-examples/>

Discuss which of these 3 strategies (described in the article) would help prevent each of the scenarios:

- Employing Testing
- Employing a Governance framework
- Increasing data observability

What other principles from today's topic would you apply?

Apply hands-on exercise briefing

Your homework (off-the-job)

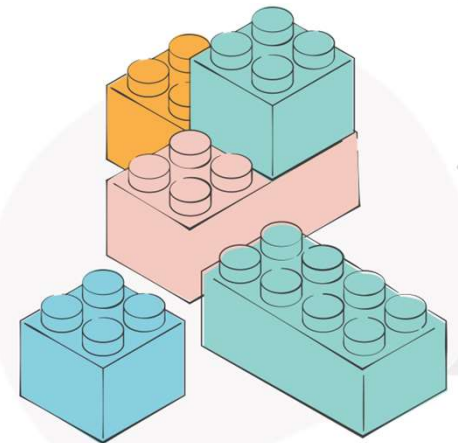
Your off-the-job apply activities for this topic are as follows:

1. Creating database diagrams with LucidChart
2. Reflection on designing a data product
3. Applying TOGAF principles to data governance



Further details can be found [here](#) and in the Hub for this topic.

Building Careers
Through Education



Apply exercises



Key Learning Summary

The key takeaways from this session are as follows:

- Architecture diagrams are crucial for managing complexity and change in data systems
- Diagrams help convey key system aspects, initiate discussions, and provide different perspectives
- Applying lenses, layers, and chunks enhances system comprehensibility and adaptability
- Public cloud computing offers powerful capabilities for reliable data architectures
- Data Mesh promotes decentralised data ownership and empowers domain-driven data management

Building Careers
Through Education



Module consolidation



Building Careers
Through Education





Thank you

**Do you have any questions,
comments, or feedback?**

