**ZACHMAN FRAMEWORK**

The purpose of **enterprise architecture** is to show a **complete picture** of how all the parts of a business work together. These parts include:

* **Physical servers** (real computers in data centers),
* **Cloud systems** (like Google Cloud or AWS), and
* **Applications** used by clients and employees (such as websites, apps, or internal tools).

All these systems must **work together** to help the business run properly.

To plan and manage these systems, a **framework** is used. A framework is like a **guide** that helps organize and explain how everything is connected.

**What is the Zachman Framework?**

The **Zachman Framework** is one of the **oldest and most well-known** enterprise architecture frameworks. It was first introduced in **1987** and is now in **version 3.0**.

This framework helps **organize information** about a business from **different points of view**. It helps show how each part fits into the whole system and how they relate to each other.

**What does it look like?**

The Zachman Framework looks like a **table or matrix** made up of **columns and rows**.

Across the **top (columns)** are six simple questions:

* **What?**
* **How?**
* **Where?**
* **Who?**
* **When?**
* **Why?**

These are often called the **W5 questions**.

Along the **side (rows)** are six levels of detail:

1. **Contextual** – gives the big picture or general scope of the business
2. **Conceptual** – shows basic ideas and models
3. **Logical** – includes diagrams and system planning
4. **Physical** – shows technology details like software and hardware
5. **Detailed / As-Built** – shows how the system currently works
6. **Functioning Enterprise** – shows the complete and working business system

Each **cell** in the matrix shows information that answers **one question** at **one level of detail**.

**Example**

* The cell in the **“Why” column** and **Contextual row** may list the business goals.
* The cell in the **“How” column** and **Conceptual row** might describe the basic steps of a business process.

**Rules of the Framework**

There are a few simple rules when filling out the matrix:

* All cells in each row should be **connected and make sense together**.
* Each cell should also connect to the cell **above and below it**.
* The full matrix should give a **complete and clear view** of the business from each point of view.

**Benefits of the Framework**

* It is **easy to understand**.
* It is **flexible** and can be used for many types of businesses and projects.
* It helps teams **organize important information** about systems and processes.

**How the Rows Work (from top to bottom)**

* The **first row** gives a general overview.
* The **second row** shows simple models to explain how things work.
* The **third row** includes more detailed diagrams.
* The **fourth row** talks about specific technology used.
* The **fifth row** shows how the system works right now.
* The **sixth row** shows the actual business running with all its systems in place.

Each row adds **more detail** to the one above it.

**Limitations of the Framework**

* Some answers might be **too broad** or **unclear**, since the framework is very general.
* It can be **hard to decide** where to put certain information.
* The framework does **not include steps or methods** to build systems — it only helps **organize what needs to be considered**.
* It is a tool for **structuring data**, not a full guide for creating solutions.

**Summary:**

The **Zachman Framework** is useful for **organizing enterprise information**, but it should be **used with other tools**. Newer frameworks like **TOGAF (The Open Group Architecture Framework)** offer **step-by-step methods** that help in actually designing and building business systems.

**THE ONE GROUP ARCHITECTURE FRAMWORK (TOGAF)**

**What is TOGAF?**

TOGAF stands for **The Open Group Architecture Framework**. It is a **tool or guide** used by companies to help them **improve the way their business works**.

When we talk about improving a business, there are three important things to focus on:

1. **People** – These are the employees who work in the company, and also the customers who buy from the company.
2. **Processes** – These are the steps or methods the company uses to do things, like how it sells products or delivers services.
3. **Technology** – These are the tools and systems that support the business, such as computers, software, apps, data, and networks.

A big mistake some companies make is using **new technology** without thinking about how it affects the people or the processes. This can make things worse instead of better.

In fact, studies show that **70% of digital projects fail**, and many **cloud upgrades don’t succeed either**. This is why TOGAF is helpful — it gives a **step-by-step method** to make sure new systems will actually **help the business**.

**What is the Purpose of TOGAF?**

The purpose of TOGAF is to help companies **design systems** (like IT systems) that are built to:

* Support the **business goals**
* Help the **people** using them
* Match the **way the business works**
* Use the right **technology**

Instead of just picking a new tool or system because it’s popular, TOGAF makes sure the system fits the company’s real needs.

**How Does TOGAF Work?**

TOGAF uses a process called the **Architecture Development Method (ADM)**. This is a series of **phases** (or steps) that guide how to plan and build a system.

Here are the **phases of TOGAF ADM**, explained in very simple words:

**1. Architecture Vision Phase**

* This phase is about **understanding what the company wants to achieve**.
* It helps make sure the future system will **support the company’s goals**.

**2. Business Architecture Phase**

* This step looks at how the business works **right now** (called the current business architecture).
* Then it plans how the business **wants to work in the future** (called the target business architecture).

**3. Information Systems Architecture Phase**

This phase has two parts:

**a. Data Architecture**

* Looks at how the company **uses its data now**, and how it wants to use data in the future.

**b. Application Architecture**

* Looks at what **software or apps** the company uses now, and what apps it **needs in the future**.

**4. Technology Architecture Phase**

* This step checks what **technology tools** (like servers, systems, networks) the company has now.
* Then it decides what **new technology is needed** to support the business goals.

**5. Opportunities and Solutions Phase**

* This phase finds possible **solutions** to build the system.
* It also lists the **technology parts** (like virtual machines, storage, firewalls) that may be used. These are called **architecture building blocks**.

**6. Migration Planning Phase**

* This step is about **planning how to move** from the current system to the new one.
* It includes things like the **schedule**, how many **people are needed**, and what **skills** they need.

**7. Implementation Governance Phase**

* This is where the company picks **who is in charge** of the project.
* It also checks who will **make decisions** and **make sure everything is done properly**.

**8. Change Management Phase**

* This final step is about **managing changes** that may happen.
* It checks how changes will **affect the people, processes, and technology**.
* It also does an **impact assessment** to understand the risks and effects of those changes.

**Summary:**

TOGAF is not just about technology. It is about designing the **right system** for a company by looking at the **people**, the **processes**, and the **technology** all together. It helps avoid costly mistakes and ensures that the final system will truly **help the business grow and succeed**.

**C4 ARCHITECTURE MODEL**

**What is the C4 Model?**

The **C4 model** is a way to **design and explain software systems** using just **four simple diagrams**. These diagrams help show how a system works, from a big picture view down to the details of the code.

The goal of the C4 model is to make software architecture **easy to understand**, even for people who didn’t build the system. A lot of older architecture diagrams were confusing, hard to read, or not consistent. That’s why the **C4 model** was created — to make everything **clear, simple, and structured**.

The “C4” name comes from the **four levels of diagrams**:

1. **Context**
2. **Containers**
3. **Components**
4. **Code**

Let’s look at each of these levels in very simple terms.

**1. Context Diagram – The Big Picture**

This is the **first diagram**, and it shows the **overall system** at a high level.

* It shows **what the system is**,
* **Who uses it** (like customers or other systems), and
* **How the system connects** with other outside systems.

**Example:**  
In a **banking application**, the context diagram would show:

* A **personal banking customer**
* Accessing the **Internet Banking System**
* Which connects to other systems like an **email system** or a **mainframe backend system**

This diagram helps people understand **what the system is for**, and who or what it **interacts with**.

**2. Container Diagram – What's Inside the System**

The **second diagram** is the **container diagram**.

Note: This does **not** mean Docker containers. In the C4 model, "containers" mean the **different parts** of the software system — like apps or databases.

This diagram **zooms in** and shows the **main parts** (or “containers”) inside the system.

**Example:**  
Inside the **Internet Banking System**, there might be:

* A **Web App** (for browsers)
* A **Mobile App**
* An **API App** (for connecting systems)
* A **Database**
* A **Single-Page Application (SPA)**

This diagram shows **how the parts of the system talk to each other**.

**3. Component Diagram – The Services or Parts Inside a Container**

The **third diagram** is the **component diagram**.

It focuses on **one container** (like the API app) and shows the **smaller parts** inside it.

**Example:**  
Inside the **API Application**, there could be:

* A **Sign-in Service**
* A **Reset Password Service**
* An **Account Summary Component**
* A **Security Component**
* An **Email Component**
* A connection to the **Mainframe Processing System**

This diagram gives a **closer look** at what **each part does** inside a single container.

**4. Code Diagram – The Detailed Code View**

The **fourth diagram** is the **code diagram**.

This one is the **most detailed**. It shows **how the code is structured**, like a **class diagram** for a specific component or service from the previous step.

**Example:**  
If we look closer at the **Sign-in Service**, the code diagram might show:

* The **classes** or **methods** used
* How the code is **organized**
* What files or logic are inside

This helps **developers understand how the service is built** in actual code.

**Summary of the C4 Model (in simple terms):**

* **Context** – Shows the system and who uses it
* **Containers** – Shows the big parts inside the system (like apps, APIs, databases)
* **Components** – Shows the services inside a container (like login, email, etc.)
* **Code** – Shows how the services are built in code (like class diagrams)