Enterprise System Important Topics

Unit 2:

1. Distributed Enterprises, why they are important?

Distributed Enterprises

The modern interpretation of a distributed enterprise is a central office space with localized IT staff and networking facilities. This definition also includes several other remote or branch sites that have few or no IT staff. Remote departments and branches usually have similar needs to centralized offices but with varying degrees of scalability.

A distributed enterprise that has a considerable remote hybrid office or workforce generally reproduces the central office's platform stack on a considerably smaller scale. As businesses evolve, prosper, and reach new markets, it's essential to introduce a presence where customers are located. Distributed enterprise strategies evaluate cost and complexity and approximate them against growth potential.

Why Are Distributed Enterprises Important?

Distributed enterprise and geographically dispersed teams are important because of the flexibility and balance associated with remote work. Businesses that adopt hybrid working models garner employees who are on average 30% more productive than their in-office counterparts.

Distributed enterprise solutions are also cheaper to maintain and offer a wealth of hiring options and benefits for organizations. One of the most important advantages is workplace engagement which builds positive associations and connections to work-based environments and cultures.

2. Give the overview of redundancy process and how

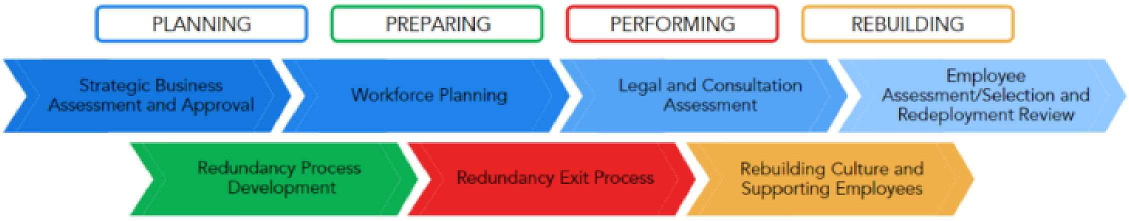
What is a redundancy process?

Generally, a redundancy or layoff is an involuntary termination of employment where a job or role is no longer required. Redundancies are usually caused by factors, such as changes in economic conditions, business efficiencies or technological advancements.

A redundancy process, on the other hand, is the systematic approach to reducing roles as an organisation undergoes restructuring. Taking a people-centred approach to redundancy management can result in better outcomes for you, your business and your workforce.

How to follow Redundancy Process

The People Shift's Pragmatic Redundancy Process


PLANNING
PREPARING
PERFORMING
REBUILDING
Strategic Business Assessment and Approval
Workforce Planning
Legal and Consultation Assessment
Employee Assessment/Selection and Redeployment Review
Redundancy Process Development
Redundancy Exit Process
Rebuilding Culture and Supporting Employees

Caption: Overview of The People Shift's pragmatic redundancy program.

Strategic Business Assessment and Approval

- Inspect the current state of your business / organisation.
- Develop a map of your business goals and strategy to get a clear vantage point.
- When working for a larger organisation, investigate if you are eligible to run the redundancy. If you belong to a smaller business, understand the impacts on your budget.
- In some cases, running the redundancy requires other corporate members.

Workforce Planning

- Analyse your talent against your current goals.
- Consider the future and the skills and experience you have to meet your future strategy.
- Identify the gaps in your workforce and assess the external market.
- Determine the solution that suits your needs and understand if redundancies are required.

Legal and Consultation Assessment

- Assess legal requirements and ensure you know your obligations under the law.
- Assess Industrial Relations/Employee Relations Risks and Union consultation or collaboration requirements under any enterprise agreements.
- Ensure that the people involved sign confidentiality statements and understand the implications of breaching these.

Employee Assessment/Selection and Redeployment Review

- Determine if you even need to run a selection process. If your redundancy program involves reducing the number of people performing the same roles, then the answer will be yes.
- Carry out a sound assessment process—include evidence requirements, relevant training for leaders and assessment procedures.

Redundancy Process Development

- Develop a redundancy and exit process, considering the most dignified approach for impacted employees.
- Develop templates, tools, scripts and training that may be required to ensure consistency and professionalism of your program.
- Using scripts to support all redundancy meetings will ensure conversations stay factual, objective and free of bias.

Redundancy Exit Process

Distribute communications to all relevant stakeholders in line with your communication plan. Consider the following:

- Appropriate Dates and Venue
- Communication to Employees
- Structure of Meetings
- Employee Support
- Exit Checklist
- Government Notification

Rebuilding Culture and Supporting Employees

- Communicate to the workforce about the end of the process
- Acknowledge and honour those that left and create opportunities for people to talk
- Monitor individuals to ensure mental health is intact—be aware of survival guilt

- Coordinate employee assistance program (EAP) or mental health support. Initiate engagement activities over the following weeks

3. Explain the concept of collaboration and how you can classify it in different categories

Collaboration Concept

Enterprise collaboration is defined as the system which is used for communication and collaboration in organizations to improve information gathering and knowledge sharing among employees, despite where they are located. The goal of enterprise collaboration is to make it easier for employees to work together, accomplish their tasks efficiently, and improve the overall business output.

It can be achieved through social networking tools, dedicated collaboration platforms, the company intranet, or even the internet. Successful enterprise collaboration in organizations leads to higher productivity, lesser chances of miscommunication, increased employee engagement, and better-informed employees.

There are two types of enterprise collaboration:

Internal enterprise collaboration

Internal enterprise collaboration happens within the workspace of the organization and it helps in bringing together all the employees. As the main benchmark for enterprise collaboration, it is the key to efficiency in organizations. It allows employees to stay connected and informed by ensuring seamless communication.

External enterprise collaboration

External enterprise collaboration happens mostly on social media platforms among employees, consumers, and brands. It can also take place between companies that need to collaborate with each other in order to work together on products or services.

4. Pros and cons of collaboration.

Benefits of enterprise collaboration

1. Reduced email

While email doesn't have to permanently go away, the fact remains that email wasn't built for today's digital workplaces. We now have an increasing number of remote employees who need real-time collaboration in order to work together seamlessly. An enterprise collaboration tool offers real-time communication which can reduce back and forth emails while increasing productivity.

2. Improved transparency

Enterprise collaboration tools allow employees to easily track and manage projects, drive personal accountability and establish higher levels of trust which directly leads to more transparency within the organization.

3. Better teamwork

Real-time communication in teams results in more effective and efficient collaboration. Moreover, centralized file storage also cuts down on errors and confusion among team members. When all the employees collaborate through a single platform, it also helps support and improves cross-team collaboration.

4. *Fewer apps, more efficiency*

On average, employees are forced to switch between up to 35 applications every minute — that is more than 1100 times in a day. Constant application switching can eat up a lot of time and eventually affect the overall work efficiency.

An enterprise collaboration platform offers a single platform for all of the organization's communication and collaboration needs. It improves efficiency by replacing unnecessary applications with overlapping features.

Major hurdles with enterprise collaboration

1. *No companywide collaboration strategy*

For many organizations, collaboration isn't considered as a companywide cultural value and as a result, there is no collaboration strategy developed.

But the absence of a well-defined collaboration strategy can make it difficult to find the right enterprise collaboration tools that align with your main business requirements and collaboration challenges. In the end, you may even end up with an enterprise collaboration platform that introduces more problems than it solves for your employees.

As a result, it is important for organizations to first gather input and feedback from employees to create a collaboration strategy first and then find the right enterprise collaboration platforms that fit their needs.

2. *Misinterpreting collaboration*

If organizations don't have a clear way of defining what collaboration means to them, it can often get misinterpreted and create a big hurdle in implementing organization-wide enterprise collaboration.

For instance, it is common to confuse cooperation with collaboration. Cooperation occurs when people work closely together and share their resources, ideas, and expertise to support each other's individual goals. Collaboration, on the other hand, is about working together in order to achieve shared goals.

3. *Lack of support for collaboration*

It's good to recognize and reward individual performance, but creating recognition programs for team performance is just as important to create a supportive culture where employees feel encouraged to communicate and share their expertise with their co-workers in order to achieve shared goals.

After all, if your employees look at collaboration with their coworkers as the time that they could have otherwise spent working on their individual tasks, it can be challenging to motivate them to collaborate effectively as a team, even with the most effective enterprise collaboration platforms.

4. *Problems in adopting new technology*

Nobody wants to break something that isn't broken. Introducing a new enterprise collaboration platform when employees believe they have been managing 'just fine' with the existing collaboration tools can be a huge challenge.

As a result, business leaders should first start by initiating discussions with employees to explain the need to shift to an enterprise collaboration platform, how it can help and why it is being implemented.

5. How data transformation can be achieved and what function does it perform?

Extraction and parsing

In the modern ELT process, data ingestion begins with extracting information from a data source, followed by copying the data to its destination. Initial transformations are focused on shaping the format and structure of data to ensure its compatibility with both the destination system and the data already there. Parsing fields out of comma-delimited log data for loading to a relational database is an example of this type of data transformation.

Translation and mapping

Some of the most basic data transformations involve the mapping and translation of data. For example, a column containing integers representing error codes can be mapped to the relevant error descriptions, making that column easier to understand and more useful for display in a customer-facing application.

Translation converts data from formats used in one system to formats appropriate for a different system. Even after parsing, web data might arrive in the form of hierarchical JSON or XML files, but need to be translated into row and column data for inclusion in a relational database.

Filtering, aggregation, and summarization

Data transformation is often concerned with whittling data down and making it more manageable. Data may be consolidated by filtering out unnecessary fields, columns, and records. Omitted data might include numerical indexes in data intended for graphs and dashboards or records from business regions that aren't of interest in a particular study.

Data might also be aggregated or summarized. by, for instance, transforming a time series of customer transactions to hourly or daily sales counts.

BI tools can do this filtering and aggregation, but it can be more efficient to do the transformations before a reporting tool accesses the data.

Enrichment and imputation

Data from different sources can be merged to create denormalized, enriched information. A customer's transactions can be rolled up into a grand total and added into a customer information table for quicker reference or for use by customer analytics systems. Long or freeform fields may be split into multiple columns, and missing values can be imputed or corrupted data replaced as a result of these kinds of transformations.

Indexing and ordering

Data can be transformed so that it's ordered logically or to suit a data storage scheme. In relational database management systems, for example, creating indexes can improve performance or improve the management of relationships between different tables.

Anonymization and encryption

Data containing personally identifiable information, or other information that could compromise privacy or security, should be anonymized before propagation. Encryption of private data is a requirement in many industries, and systems can perform encryption at multiple levels, from individual database cells to entire records or fields.

Modelling, typecasting, formatting, and renaming

Finally, a whole set of transformations can reshape data without changing content. This includes casting and converting data types for compatibility, adjusting dates and times with offsets and format localization, and renaming schemas, tables, and columns for clarity.

Refining the data transformation process

Before your enterprise can run analytics, and even before you transform the data, you must replicate it to a data warehouse architected for analytics. Most organizations today choose a cloud data warehouse, allowing them to take full advantage of ELT. Stitch can load all of your data to your preferred data warehouse in a raw state, ready for transformation.

6. Write the use cases of batch processing system.

Financial services

Financial services organizations, from agile financial technologies to legacy enterprises, have been using batch processing in areas such as high performance computing for risk management, end-of-day transaction processing, and fraud surveillance. They use batch processing to minimize human error, increase speed and accuracy, and reduce costs with automation.

Software as a service

Enterprises that deliver software as a service (SaaS) applications often run into issues when it comes to scalability. Using batch processing, you can scale customer demand while automating job scheduling. Creating containerized application environments to scale demand for high-volume processing is a project that can take months or even years to complete, but batch processing systems help you achieve the same result in a much shorter timeframe.

Medical research

Analysis of large amounts of data—or big data—is a common requirement in the field of research. You can apply batch processing in data analytics applications such as computational chemistry, clinical modelling, molecular dynamics, and genomic sequencing testing and analysis. For example, scientists use batch processing to capture better data to begin drug design and gain a deeper understanding of the role of a particular biochemical process.

Digital media

Media and entertainment enterprises require highly scalable batch processing systems to automatically process data—such as files, graphics, and visual effects—for high-resolution video content. You can use batch processing to accelerate content creation, dynamically scale media packaging, and automate media workload.

7. Modern system can run 100 of thousands of jobs un premises or in the cloud justify this statement with valid example

The ability to run hundreds of thousands of jobs on-premises or in the cloud is a key feature of modern computing systems. This capability is made possible by the availability of powerful hardware, sophisticated software platforms, and scalable cloud infrastructure.

One example of a modern system that can run hundreds of thousands of jobs is the Apache Hadoop platform, which is designed for distributed data processing. Hadoop enables users to process massive volumes of data by breaking it down into smaller pieces and distributing those pieces across multiple computers in a cluster. Each computer in the cluster can then process its assigned piece of the data independently, allowing the system to process large amounts of data in parallel.

Another example is Amazon Web Services (AWS), which provides a range of cloud-based services that can be used to run large-scale workloads. AWS offers a variety of compute services, including EC2 (Elastic Compute Cloud) and Lambda, that allow users to run applications and services on a global network of servers. These services can be configured to automatically scale up or down based on demand, allowing users to run thousands of jobs simultaneously without worrying about infrastructure limitations.

Google Cloud Platform (GCP) is another example of a modern system that can handle hundreds of thousands of jobs. GCP offers a variety of compute services, including Compute Engine and Cloud Functions, that enable users to run applications and services on Google's global infrastructure. GCP also offers managed services like BigQuery and Dataflow, which allow users to process and analyze large volumes of data in a scalable and cost-effective manner.

In summary, modern systems like Apache Hadoop, Amazon Web Services, and Google Cloud Platform are capable of running hundreds of thousands of jobs on-premises or in the cloud. These systems offer powerful hardware, sophisticated software platforms, and scalable cloud infrastructure that enable users to process massive volumes of data and run large-scale workloads with ease.

8. How monolithic architecture is different from microservices, in your opinion for your enterprise which one is most suitable.

Monolithic architecture and microservices are two distinct architectural patterns used in software development. In my opinion, the suitability of one over the other depends on the specific needs and goals of the enterprise.

Monolithic architecture is a traditional approach in which an entire application is built as a single, self-contained unit. This type of architecture typically involves a tightly coupled set of components that are deployed together and share a common codebase, database, and runtime environment. The advantages of monolithic architecture include simplicity, ease of development, and ease of deployment.

On the other hand, microservices architecture is a more modern approach in which an application is broken down into smaller, independent services that communicate with each other through APIs. Each service in a microservices architecture can be developed and deployed independently, allowing for greater flexibility, scalability, and fault tolerance. However, the complexity of managing multiple services can be higher than in a monolithic architecture.

For my enterprise, I would consider the trade-offs between monolithic and microservices architectures based on the specific requirements of our software applications. If our applications are relatively simple and require fast development and deployment cycles, a monolithic architecture may be more suitable. On the other hand, if our applications require high scalability, fault tolerance, and the ability to support a large number of users or transactions, a microservices architecture may be more appropriate.

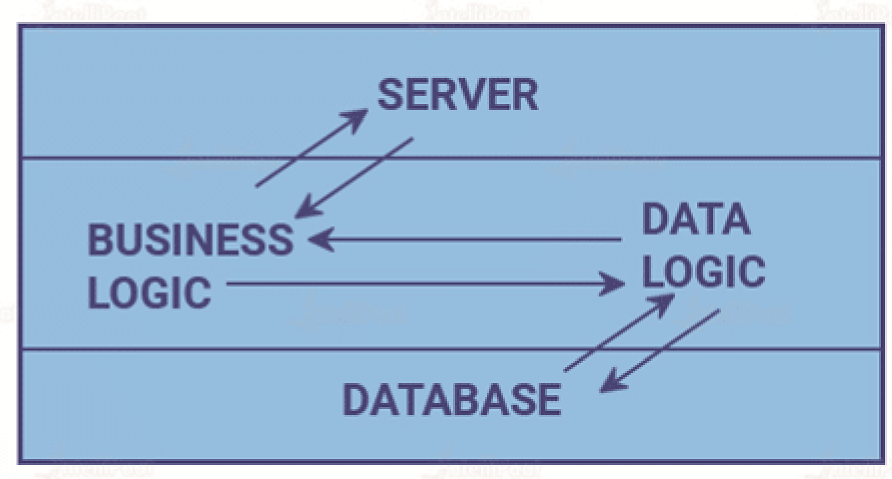
Ultimately, the decision between monolithic and microservices architecture will depend on a range of factors, including the size and complexity of the application, the development team's skills and experience, the level of integration required between different components, and the performance and scalability needs of the application. It is important to carefully evaluate the advantages and disadvantages of each approach before making a decision.

9. How many types of client server architecture are there, write the advantages and disadvantages of client server architecture

Types of client server architecture

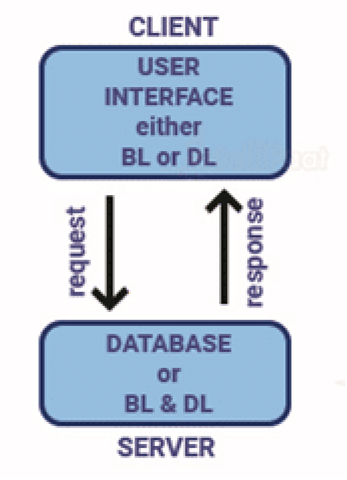
The functionality of client server architecture is in various tiers.

1-tier architecture



In this category of client server architecture, the architecture contains all kinds of settings, such as configuration setting and marketing logic, on a single device. While the diversity of services offered by 1-tier architecture makes it one of the reliable sources, handling such an architecture is difficult. This is primarily due to the data variance. It often results in replication of work. 1-tier architecture consists of several layers, such as presentation layer, business layer, and data layer, that are combined with the help of a unique software package. The data present in this layer is usually stored in local systems or on a shared drive.

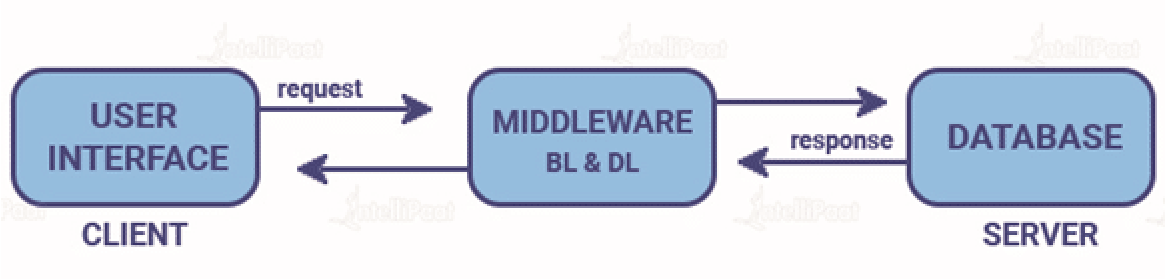
2-tier architecture



his architecture has the best environment. In this architecture, the user interface is stored on the client's side and the database is stored on the server, while database logic and business logic is maintained either on the client's side or on the server's side.

The 2-tier architecture is faster in comparison to the 1-tier architecture; this is because the 2-tier architecture does not have any intermediary between the client and the server. It is often utilized to avoid confusion between clients. One of the popular examples of 2-tier architecture is the online ticket reservation system.

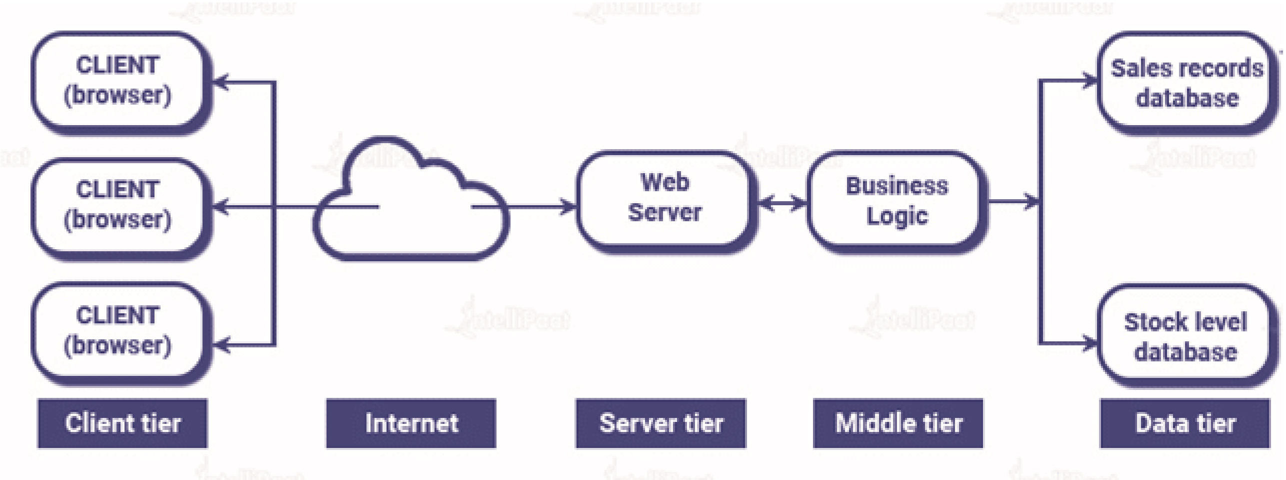
3-tier architecture



Unlike 2-tier architecture that has no intermediary, in 3-tier client server architecture, a middleware lies between the client and the server. If the client places a request to fetch specific information from the server, the request will first be received by the middleware. It will then be dispatched to the server for further actions. The same pattern will be followed when the server sends a response to the client. The framework of 3-tier architecture is categorized into three main layers, presentation layer, application layer, and database tier.

All three layers are controlled at different ends. While the presentation layer is controlled at the client's device, the middleware and the server handle the application layer and the database tier respectively. Due to the presence of a third layer that provides data control, 3-tier architecture is more secure, has invisible database structure, and provides data integrity.

N-tier architecture



N-tier architecture is also called multi-tier architecture. It is the scaled form of the other three types of architecture. This architecture has a provision for locating each function as an isolated layer that includes presentation, application processing, and management of data functionalities.

Advantages and disadvantages of client-server architecture

Advantages

- The centralized network has complete leverage to control the processes and activities.
- All devices in the network can be controlled centrally.
- Users have the authority to access any file, residing in the central storage, at any time.

- It provides a good user interface, easy file finding procedure, and management system for organizing files.
- Easy sharing of resources across various platforms is possible.

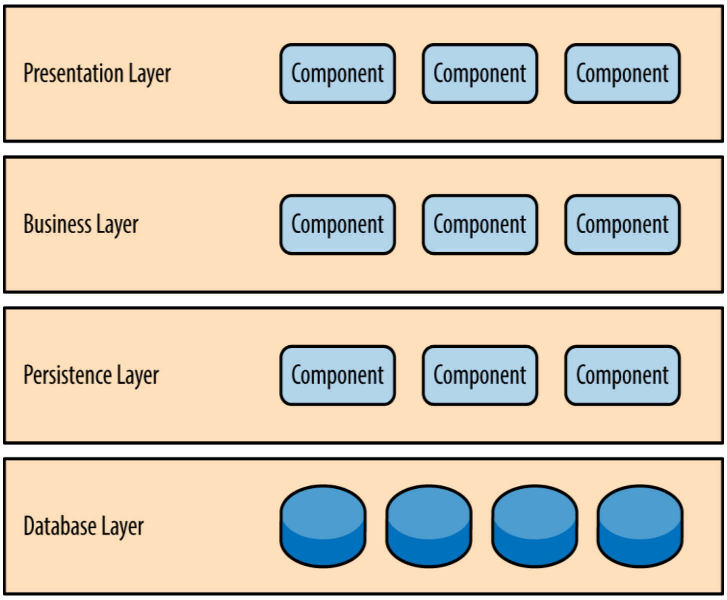
Disadvantages

- If the primary server goes down, the entire architecture is disrupted.
- It is expensive to operate because of the cost of heavy hardware and software tools.
- This architecture requires particular OSs related to networking.
- Too many users at once can cause the problem of traffic congestion.
- It requires highly technical stuff, such as server machines, for maintenance of the network.

Unit 3:

1. Elaborate on concept of layered architecture with suitable diagram

Layered architecture, also known as the n-tier architecture, is a software design pattern that separates the different components of an application into layers. Each layer is responsible for a specific set of functions and communicates with the layers above and below it using well-defined interfaces. This approach makes it easier to maintain, test, and scale the application.



- **Presentation layer** would be responsible for handling all user interface and browser communication logic.
- **Business layer** would be responsible for executing specific business rules associated with the request.
- **The persistence layer**, also called the data access layer, acts as a protective layer. It contains the code that's necessary to access the database layer. This layer also holds the set of codes that allow you to manipulate various aspects of the database.
- **Database layer** is responsible for handling data, databases.

For example, when a user submits a form in the web application***, the presentation layer*** would receive the form data and pass it to the business layer. ***The business layer*** would validate the data, perform any necessary business logic, and coordinate with the persistence layer to store or retrieve data. The business layer would then communicate with the presentation layer to display the results of the request to the user.

When a user requests data from the web application, such as a list of products or a customer record, the presentation layer would receive the request and pass it to the business layer. The business layer would coordinate with ***the persistence layer*** to retrieve the requested data from the database. The business layer would then communicate with the presentation layer to display the data to the user.

Notice in Figure 1-2 that each of the layers in the architecture is marked as being closed. This is a very important concept in the layered architecture pattern. A closed layer means that as a request moves from layer to layer, it must go through the layer right below it to get to the next layer below that one. For example, a request originating from the presentation layer must first go through the business layer and then to the persistence layer before finally hitting the database layer.

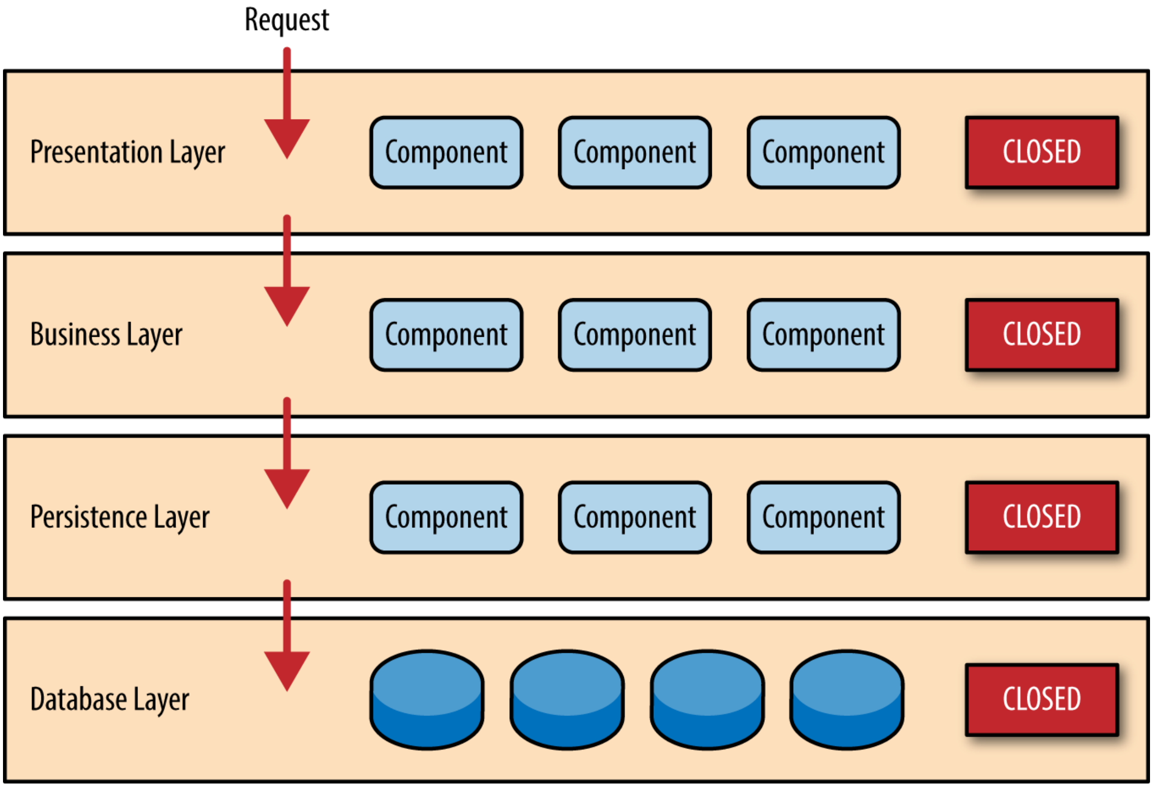
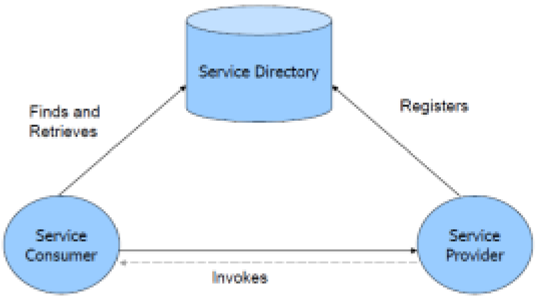


Figure 1-2. Closed layers and request access

2. Service oriented architecture (VVIP)/ Differences could be asked

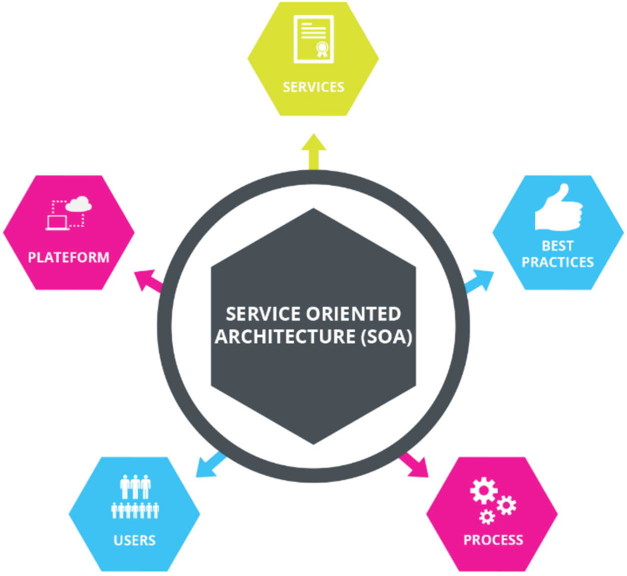
Service Oriented Architecture (SOA) is an architectural style that involves the creation of reusable services that can be accessed by different applications and systems. SOA enables businesses to build flexible and scalable systems that can adapt to changing business requirements.



A SOA has three major parts; **service provider, service consumer, and service directory.** ***Service providers*** are the parties who build service and make available service. ***Service consumers*** are the clients who consume services. ***Service directory*** is the place where service providers register the services and consumer search for services. Service directory provide following services:

1. Scalability of services; can add services incrementally.
2. Decouples consumers from providers.
3. Allows for hot updates of services.
4. Provides a look-up service for consumers.
5. Allows consumers to choose between providers at runtime rather than hard-coding a single provider

There are three roles in each of the Service-Oriented Architecture building blocks: *service provider; service broker, service registry, service repository; and service requester/consumer*.



Advantages of SOA:

1. **Modularity**: SOA allows for the creation of small, independent services that can be reused across multiple applications, reducing the need to develop and maintain duplicate functionality.
2. **Flexibility**: SOA's loosely coupled architecture allows services to be added, removed, and modified without impacting other services, making it easier to adapt to changing business requirements.
3. **Scalability**: SOA allows for the creation of scalable services that can handle large volumes of requests, ensuring that the application remains performant even under heavy loads.
4. **Interoperability**: SOA allows services to be developed using different programming languages, platforms, and technologies, enabling them to communicate with each other seamlessly.

Disadvantages of SOA:

1. **Complexity**: SOA can be complex to design and implement due to its distributed nature, making it difficult to manage and troubleshoot.

2. **Overhead**: The additional layers of abstraction and communication involved in SOA can introduce overhead, slowing down the performance of the application.
3. **Security**: SOA's distributed nature can make it more difficult to ensure the security of the application, requiring additional measures to protect against attacks.
4. **Integration**: Integration with existing systems can be challenging, requiring additional effort to ensure that data is exchanged correctly and consistently.

Feature	Monolith	SOA	Microservices
Architecture	A single, self-contained application that includes all of the application's functionality.	A distributed architecture that uses standardized protocols to enable communication between services.	A distributed architecture that breaks an application down into small, independent services that communicate with each other using lightweight protocols.
Advantages	- Simple to develop and deploy. - Easy to maintain.	- Reusability, as services can be shared between different applications and systems. - Flexibility, as services can be modified and updated without impacting other parts of the system. - Interoperability, as services can be accessed using different technologies and platforms. - Scalability, as services can be scaled independently of each other.	- Scalability, as each service can be scaled independently of each other. - Flexibility, as each service can be modified and updated without impacting other services. - Resilience, as failures in one service do not necessarily impact the entire application. - Easier to update and deploy, as changes can be made to individual services without re-deploying the entire application.
Disadvantages	- Limited scalability, as the entire application must be scaled as a single unit. - Difficult to update and deploy, as any changes require re-deploying the entire application.	- Complex to design and implement, as the architecture requires careful planning and coordination. - Potential for service dependency issues, as changes to one service can impact other services that depend on it.	- Complexity of managing multiple services, as the architecture requires careful coordination and monitoring. - Potential for service dependency issues, as changes to one service can impact other services that depend on it - Increased overhead, as there is additional network communication overhead between services.

3. List down the companies which uses SOA

Service-Oriented Architecture (SOA) is a design pattern that emphasizes the use of loosely coupled services to support the integration and sharing of business processes and data. Many companies across a range of industries have adopted SOA to improve their agility, flexibility, and efficiency. Some of the notable companies that use SOA include:

1. **Amazon**: Amazon uses SOA to manage its vast e-commerce platform, which includes millions of products and millions of customers. SOA enables Amazon to rapidly develop and deploy new services, as well as scale its infrastructure to meet demand.

2. **IBM**: IBM uses SOA to provide a wide range of enterprise software solutions, including its WebSphere platform. SOA enables IBM to provide a flexible, modular architecture that can support a variety of business processes and workflows.

3. **FedEx**: FedEx uses SOA to support its logistics and supply chain management operations, which involve tracking and delivering packages to millions of customers worldwide. SOA enables FedEx to integrate with a wide range of systems and services, as well as manage complex workflows and business rules.

4. **Verizon**: Verizon uses SOA to support its telecommunications services, including its wireless, broadband, and TV offerings. SOA enables Verizon to provide a flexible and scalable infrastructure that can support a wide range of devices and platforms.

5. **JPMorgan Chase**: JPMorgan Chase uses SOA to support its banking and financial services, including its online banking platform. SOA enables JPMorgan Chase to integrate with a wide range of systems and services, as well as manage complex workflows and business rules.

6. **UPS**: UPS uses SOA to support its logistics and supply chain management operations, which involve tracking and delivering packages to millions of customers worldwide. SOA enables UPS to integrate with a wide range of systems and services, as well as manage complex workflows and business rules.

7. **Cisco**: Cisco uses SOA to support its networking and communications solutions, including its Unified Communications Manager platform. SOA enables Cisco to provide a flexible and scalable architecture that can support a wide range of devices and platforms.

These are just a few examples of the many companies that use SOA to improve their operations and achieve business agility.

4. How do you achieve concurrency.

Concurrency refers to the ability of a system to execute multiple tasks or processes simultaneously. Achieving concurrency is essential for improving system performance, increasing throughput, and reducing response times. There are several techniques that can be used to achieve concurrency, including:

1. **Multi-threading**: This is a technique in which multiple threads of execution are created within a single process. Each thread can run independently of the others and perform a different task or function. Multi-threading is commonly used in applications that require high performance and responsiveness, such as web servers and video games.

2. **Parallel processing**: This involves dividing a task into multiple sub-tasks that can be executed simultaneously on different processors or cores. Parallel processing is commonly used in applications that require high computational power, such as scientific simulations and data analysis.

3. **Asynchronous programming**: This is a programming model in which tasks are executed independently and can continue to run in the background while other tasks are performed. Asynchronous programming is commonly used in applications that require high responsiveness and scalability, such as web applications and messaging systems.

4. **Locking and synchronization**: This involves using locks and synchronization mechanisms to control access to shared resources, such as data structures or files. Locking and synchronization is commonly used in multi-threaded applications to prevent race conditions and other concurrency-related issues.

5. **Non-blocking, I/O**: This involves using non-blocking I/O operations to perform I/O operations without blocking the execution of other tasks. Non-blocking I/O is commonly used in applications that require high concurrency and scalability, such as web servers and network applications.

These techniques can be used in combination to achieve concurrency and improve the performance and scalability of a system. However, it is important to carefully design and implement concurrency-related features to avoid issues such as deadlocks, race conditions, and resource contention.

5. Event driven Archi and plugin Archi

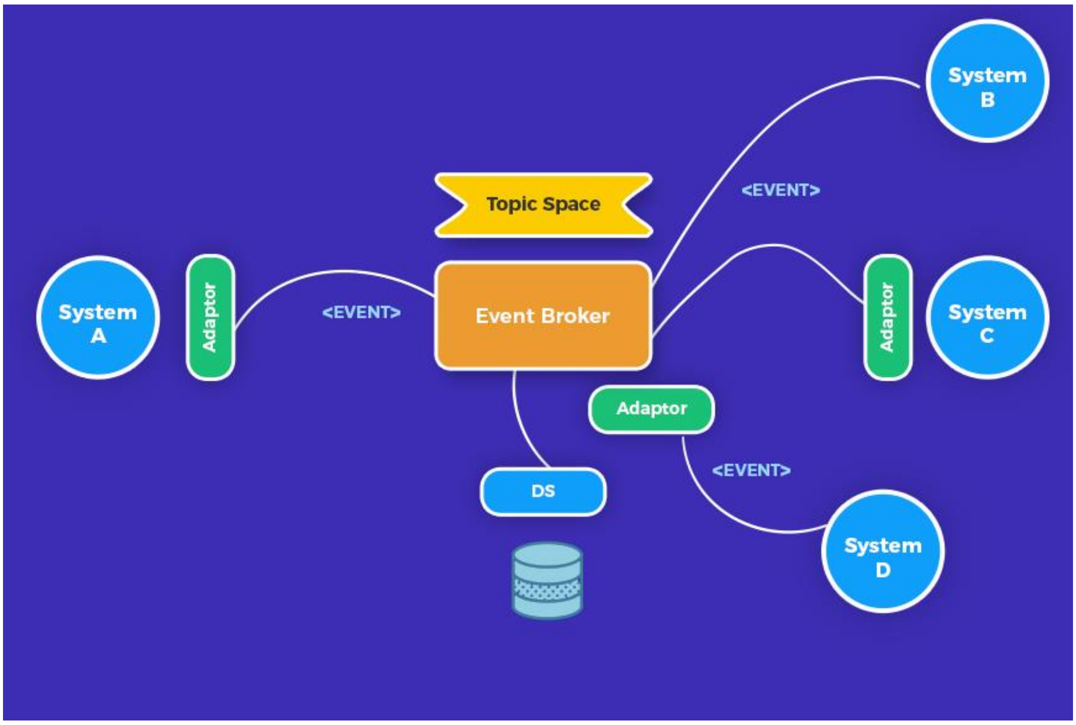
Event Driven Architecture

Event-driven architecture models your business systems as a flow of events — when an important business event happens, your systems are alerted to that change of state. A simple example could be a customer changing their address: once that state change is registered, your billing systems get notified of the new address. This is in stark contrast to traditional request-based architectures.

A major benefit of this architectural pattern is that it is both scalable and relatively easy to change. EDA’s inherently loosely coupled nature means that it’s relatively easy to make changes in one particular part of your systems, without breaking anything else.

A well-designed EDA will be based on events that are meaningful to the business. The events could be triggered by user activity, external inputs, such as sensor activity, or outputs from an analytics system. What’s important is the way you define those events, so that you’re capturing something important to your organization.

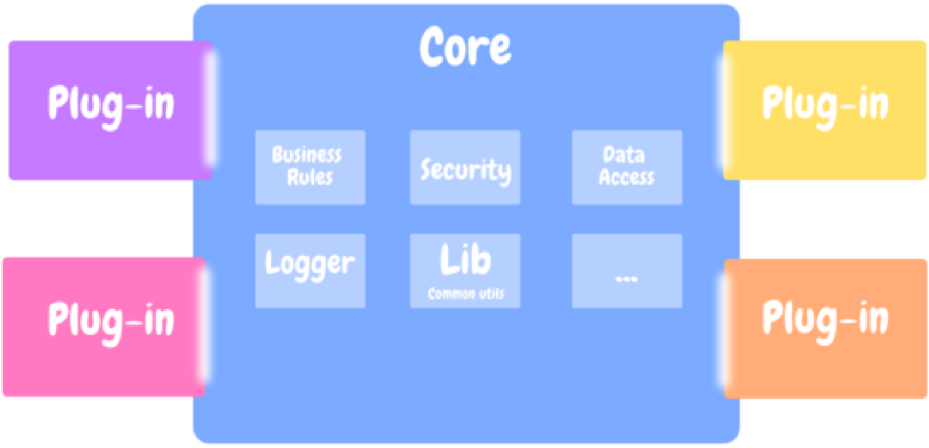
By basing your designs on these triggering events, you gain flexibility; you’re able to add new behaviours without having to redesign the entire system.



Plugin Architecture

A plug-in architecture is a software design pattern that allows applications to be extended with new functionality by loading and executing external code modules, called plug-ins or extensions. In this architecture, the application provides a framework that defines a set of interfaces or extension points, which the plug-ins can implement or extend to provide additional features.

Here is a diagram that illustrates the plug-in architecture:



Advantages of Plug-in Architecture:

1. **Extensibility**: Plug-in architecture allows the application to be easily extended with new functionality without modifying the core code.
2. **Modularization**: Plug-in architecture promotes the modularization of code, making it easier to maintain and debug.
3. **Flexibility**: Plug-in architecture provides greater flexibility, as users can choose which plug-ins to install and use.
4. **Reusability**: Plug-in architecture allows for the reuse of code across multiple applications.

Disadvantages of Plug-in Architecture:

1. **Complexity**: Plug-in architecture can introduce additional complexity, as the application must manage the loading and unloading of plug-ins.
2. **Compatibility**: Plug-in architecture can introduce compatibility issues, as different plug-ins may have conflicting dependencies or requirements.
3. **Security**: Plug-in architecture can introduce security risks, as plug-ins may be provided by third parties and may not be trustworthy.

Unit 4:

1. Why is enterprise integration important and what key features Enterprise integration have?

Why it's Important

Most businesses use several software products from different vendors in their day-to-day operations. These applications can have different uses, data stores and methods of operation. This can create data silos where data is either duplicated between databases or is available in one but not another. That leads to data inconsistency when users need to manually copy and paste data between applications. Users may also need to consult several applications for their daily work.

EAI provides a methodology that helps these disparate applications share data and workflows to reduce manual steps and errors and to provide better insight into organizational data. This becomes more important as an organization grows and as technology advances.

In the past, all business workflows were manual and carried out on paper. As computers automated work for users, the day-to-day assignments of information workers quickly became formatting data and transferring it from one system to another. This need to quickly make data available between applications will only increase as more organizations adopt microservices, software as a service (SaaS) and platform as a service (PaaS) tools.

EAI can help to remove manual steps and reduce user error.

Features

1. **Scalability**: Enterprise integration solutions should be able to scale to handle growing volumes of data and increasing numbers of users and applications.
2. **Security**: Enterprise integration solutions should provide robust security features to protect sensitive data and prevent unauthorized access.
3. **Flexibility**: Enterprise integration solutions should be flexible and adaptable to changing business needs and requirements, such as adding new systems or data sources.
4. **Reliability**: Enterprise integration solutions should be highly reliable and available, with minimal downtime or service disruptions.
5. **Interoperability**: Enterprise integration solutions should be able to integrate with a wide range of systems, applications, and data sources, regardless of platform or technology.
6. **Ease of use**: Enterprise integration solutions should be easy to use and manage, with intuitive interfaces and user-friendly features.
7. **Performance**: Enterprise integration solutions should provide high performance and low latency, with minimal impact on system resources and response times.
8. **Monitoring and analytics**: Enterprise integration solutions should provide robust monitoring and analytics capabilities to track system performance and detect issues or anomalies.

9. **Standards compliance**: Enterprise integration solutions should comply with industry standards and best practices, such as XML, SOAP, REST, and others.

2. What are the goals of enterprise integration

- **Database integration** is a common and relatively easy goal for EAI. Removing duplication of data and ensuring consistency between applications can reduce mistakes and liability. EAI can be achieved by setting different applications to use the same database or using synchronization tools to keep different databases up to date with changes. Data warehouse designs can make this possible. All these can also be more easily fed into big data systems for better business intelligence (BI).
- **Workflow integration** can get separate tools to work together to increase productivity by making common tasks easier to accomplish. As an example of EAI, onboarding an employee may require human resources to update payroll, personnel records, desk assignments on a floor plan, door access controls and IT resources. With EAI, all these different tasks can be accomplished as a single workflow instead of in separate programs.
- **Interface integration** allows organizations to present a single consistent user interface and user experience (UI and UX) to operators instead of multiple interfaces from different software packages. This can be the most difficult form of EAI to achieve as minor changes to the underlying software can result in redesign work.

3. Soap, rest rest-full

SOAP (Simple Object Access Protocol)

SOAP is a messaging protocol for exchanging information between two computers based on XML over the internet. SOAP messages are purely written in XML which is why they are platform and language independent.

A SOAP message contains:

- An Envelope that indicates the start and end of the message
- A Header that includes attributes used to process the message and is an optional element
- A Body that holds the XML data that is to be sent and it cannot be left out
- A Fault which provides error messages when processing and it is an optional element

Working with SOAP requests and responses could get very complex. Some languages make efficient use of the SOAP shortcuts to reduce the level of complexity and the .Net platform for example hides the XML to a large extent. This is made possible due to the Web Service Definition Language (WSDL). WSDL is an XML file that defines and describes the services that are available in the web service of interest. It describes the naming of services, the specifications and structure of the response sent back. The services in the WSDL are described as a compilation of network ports/endpoints. With the WSDL, the .Net platform is able to auto-generate the proxy classes and functions which can be called from the application.

- The types describe the datatypes the web service use.
- The message defines each operation’s data element
- The portType describes what operations can be done and the messages involved in this operation. This could be one of four types: One way (receives messages but no response), Request-Response (receives request and responds), Solicit-Response (sends request and waits for response) or Notification (sends request, does not require response
- Binding tells the format of data for the type of each port and the protocol.

REST (REpresentational State Transfer)

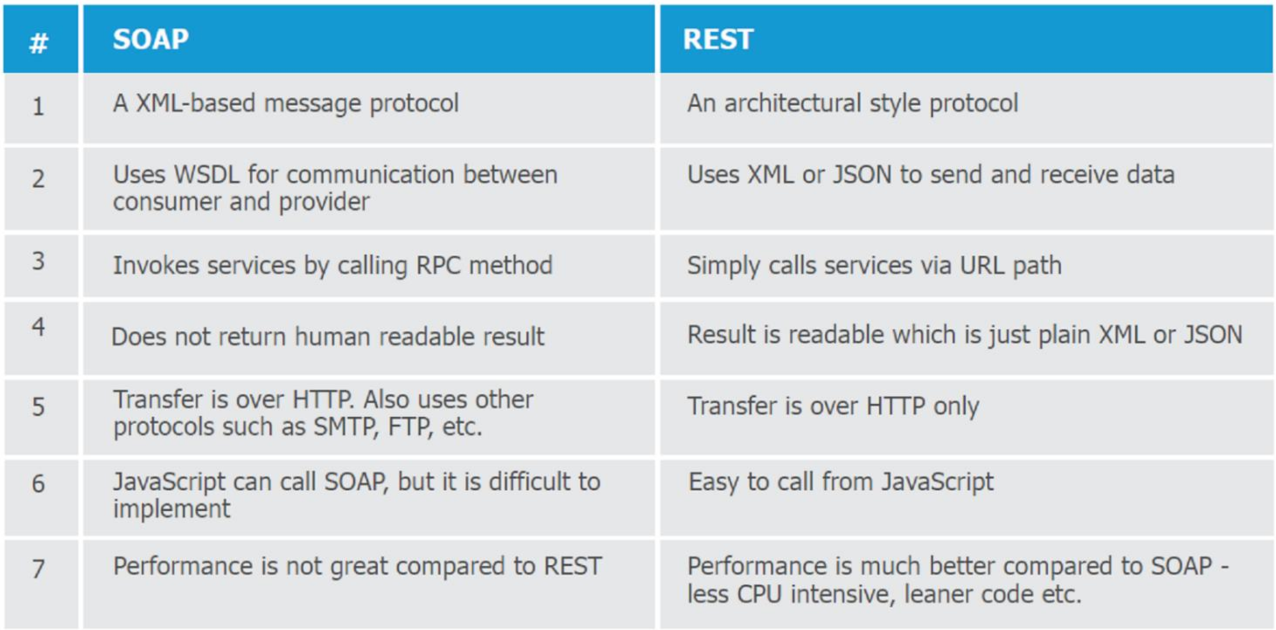
REST is a web standard architecture that achieves data communication using a standard interface such as HTTP or other transfer protocols that use standard Uniform Resource Identifier (URI). The design is such that each component in a RESTful web service is a resource that can be accessed using standard HTTP methods (if the chosen protocol is HTTP). Resources which can be thought of as objects in the concept of Object oriented programming (OOP) are identified by URIs and the resources are represented in several ways such as JSON, XML, Text etc. though JSON is currently the more favoured choice.

RESTful services have the following properties: Representations, Messages, URIs, Uniform interface, Stateless, Links between resources and caching. A quick look into these properties below using HTTP

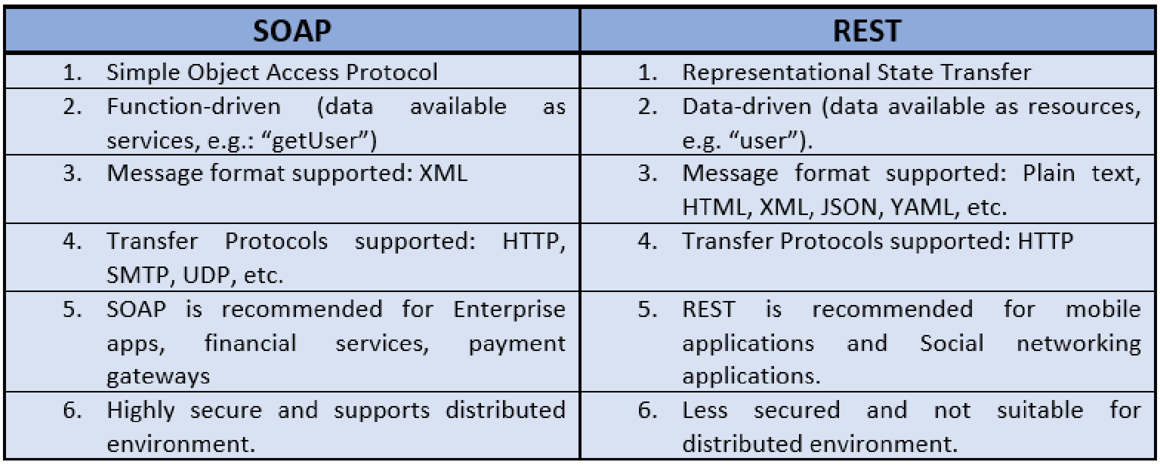
- **Representation**- resources are represented in different formats as earlier stated and should be a complete representation of the resource.
- **Messages**- This is how the client and server interact. Along with the data, messages contain metadata bout the message. When accessing a RESTful resource using HTTP, the commonly used methods are GET (reads a resource), PUT (creates a resource), DELETE (removes a resource) and POST (updates an existing resource)
- **URIs** – Each resource needs at least one URI to identify a resource(s) and the operation is determined by the HTTP verb/action. Hence a URI can be called with different actions.
- **Stateless**- Restful web services are stateless and any session state is held on the client not server. This ensures that every client to server request has the necessary information to understand the request and handles each request independently.
- **Links Between Resources**- the representation of a resource can have links to other resources.
- **Caching**- the data produced when a request is made the first time is stored and used the next time in order to stop regenerating same information for the same request and improves performance. The HTTP headers help to control caching such as age which tracks how long ago the data was fetched from the server, expires date and time the resource representation expires etc.

The above gives an oversight of the six properties of REST and it is important to remember that:

- REST is not coupled to HTTP and is actually protocol independent. It is simply not a mapping of CRUD to the HTTP methods.
- REST makes it relatively easy to integrate with websites and are exposed using XML (one of many ways) for easy consumption.

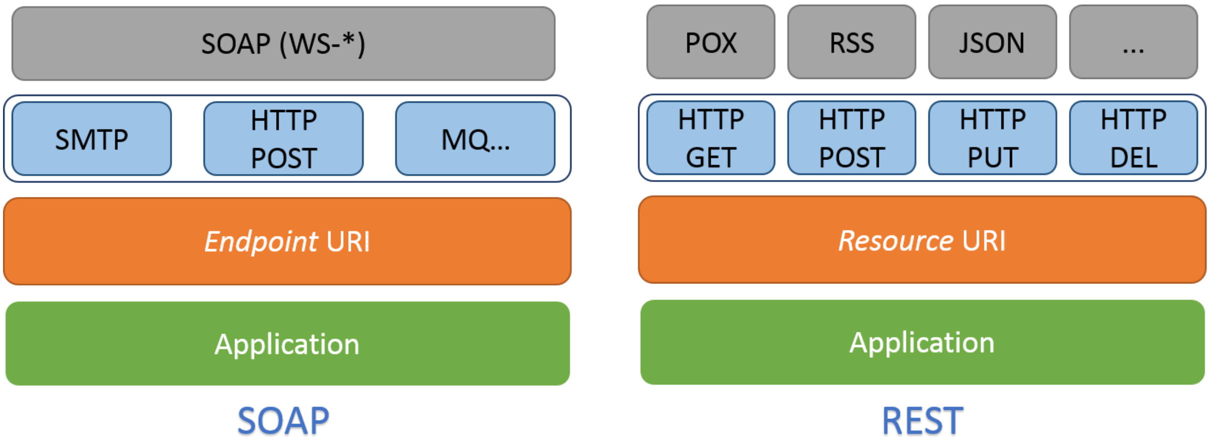


#	SOAP	REST
1	A XML-based message protocol	An architectural style protocol
2	Uses WSDL for communication between consumer and provider	Uses XML or JSON to send and receive data
3	Invokes services by calling RPC method	Simply calls services via URL path
4	Does not return human readable result	Result is readable which is just plain XML or JSON
5	Transfer is over HTTP. Also uses other protocols such as SMTP, FTP, etc.	Transfer is over HTTP only
6	JavaScript can call SOAP, but it is difficult to implement	Easy to call from JavaScript
7	Performance is not great compared to REST	Performance is much better compared to SOAP - less CPU intensive, leaner code etc.



SOAP	REST
1. Simple Object Access Protocol	1. Representational State Transfer
2. Function-driven (data available as services, e.g.: "getUser")	2. Data-driven (data available as resources, e.g. "user").
3. Message format supported: XML	3. Message format supported: Plain text, HTML, XML, JSON, YAML, etc.
4. Transfer Protocols supported: HTTP, SMTP, UDP, etc.	4. Transfer Protocols supported: HTTP
5. SOAP is recommended for Enterprise apps, financial services, payment gateways	5. REST is recommended for mobile applications and Social networking applications.
6. Highly secure and supports distributed environment.	6. Less secured and not suitable for distributed environment.

Fig. SOAP vs REST



Unit 5:

1. Deployment wala part Pros and cons

Feature	Group Policy	PDQ Deploy	CITRIX
Definition	A feature in Microsoft Windows that allows administrators to manage and configure settings on multiple computers in an organization	A software deployment tool that simplifies the process of installing and updating software across multiple computers	A virtualization solution that allows users to access applications and desktops from anywhere
Deployment	Centralized control	Centralized control	Decentralized control
Targeted Devices	Windows OS devices	Windows OS devices	Various OS and devices
Scalability	Limited to Windows environment	Limited to Windows environment	Scalable to multiple devices
Security	Secure communication	Secure communication	Secure virtualization
Automation	Automatic application	Automatic application	Automatic deployment
Management	Limited to Windows environment	Limited to Windows environment	Management of virtualization
Cost	No additional cost	Paid software	Paid software
Pros	Secured way of deploying over a network	Applications features can be edited before deploying	Suitable for apps and mobile applications
Cons	Organizing and handling large number of GPOs is difficult	Features like automated Windows update repository are missing	It is a costly solution

2. What is capacity test and how can you relate to

Capacity testing is a type of performance testing that evaluates the ability of a system, application, or infrastructure to handle a specified level of workload or user traffic. The goal of capacity testing is to determine the maximum capacity of the system and identify any bottlenecks or performance issues that may arise under different load conditions.

During capacity testing, a system is subjected to a gradual increase in workload or user traffic until it reaches its maximum capacity. The test typically measures various metrics, such as response time, throughput, and resource utilization, to determine how well the system performs under different levels of load.

Capacity testing is important for ensuring that a system can handle the expected volume of traffic or workload without degrading performance or causing downtime. It can also help organizations identify potential issues and bottlenecks before they impact the system in production.

3. How can you achieve the secure – your releases of enterprise deployment

To achieve secure releases of enterprise deployment, organizations should follow a set of best practices and security measures that can help protect against potential security threats and vulnerabilities. Some of these practices include:

- **Secure coding practices**: Developers should follow secure coding practices when developing and testing software, such as avoiding known vulnerabilities, using secure coding frameworks, and following coding standards.

- **Vulnerability testing and scanning**: Organizations should perform vulnerability testing and scanning on all code and software releases before they are deployed to production. This can help identify potential vulnerabilities and security issues that may need to be addressed.
- **Access control and authentication**: Organizations should implement access control and authentication measures to ensure that only authorized personnel have access to production environments and deployment processes. This can include using role-based access control (RBAC), multi-factor authentication, and other security measures.
- **Encryption and data protection**: Organizations should use encryption and other data protection measures to secure sensitive data and communications during deployment and in production.
- **Incident response planning**: Organizations should have an incident response plan in place in case of a security incident or breach. This plan should include procedures for detecting, containing, and responding to security incidents, as well as communication and notification procedures.
- **Regular updates and patches**: Organizations should regularly update and patch their software and systems to ensure that they are protected against known security vulnerabilities and threats.

4. What is the role of CI CD pipeline

The role of a CI/CD (Continuous Integration/Continuous Deployment) pipeline is to automate the software delivery process from development to production, with the goal of reducing errors, improving speed, and increasing reliability. The CI/CD pipeline is a set of practices and tools that allow developers to quickly and efficiently test and deploy code changes in a controlled and automated way.

The CI/CD pipeline typically consists of the following stages:

1. **Continuous Integration**: This stage involves integrating code changes from developers into a shared repository on a regular basis, typically several times a day. Automated tests are run to ensure that the code changes integrate successfully and do not introduce new errors or issues.
2. **Continuous Testing**: In this stage, automated testing is performed to ensure that the code changes meet the required quality standards and do not introduce any new errors or regressions.
3. **Continuous Deployment**: In this stage, the code changes are automatically deployed to production, either in a rolling release or a phased rollout, depending on the organization's requirements and policies.
4. **Continuous Monitoring**: This stage involves monitoring the production environment for any issues or errors that may arise. If any issues are detected, the pipeline may be paused or rolled back until the issues are resolved.

The role of the CI/CD pipeline is to automate and streamline the software delivery process, allowing organizations to deploy new features and updates more quickly and with less risk. By automating testing and deployment, the pipeline helps ensure that code changes are thoroughly tested and validated before they are deployed to production, reducing the risk of errors or downtime. It also allows for greater collaboration and visibility among teams, as code changes are integrated and tested in a shared environment, and issues can be identified and resolved more quickly.

5. Network availability- serial transport kaise achieve karoge explain it with a diagram?

The best-known metric of network availability is known as "five nines". What five nines means is that the end-user perceives that their application is available 99.999% of the time. This permits only 5.26 minutes of downtime a year. Depending on the application and network topology, this can be a very stringent standard.

Consider Figure 1 below which shows serially connected routers, switches, access points, servers, and transited clouds. When these ten elements are connected without any redundancy, each of these elements must be up and available 99.9999% (or six nines) of the time for the end-user to perceive five nines of availability. As six nines allows only 32 seconds of downtime, having a single reboot a year could prove problematic.

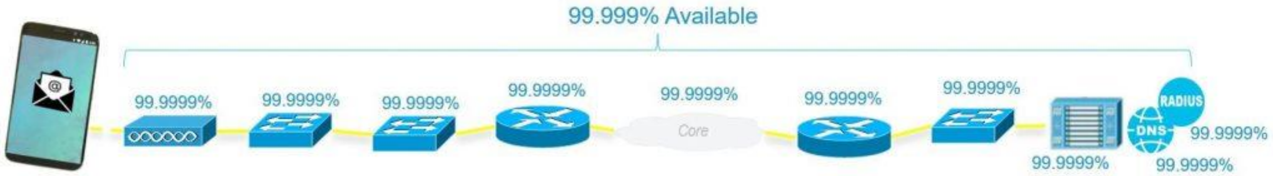


Figure 1: Serial Transport Availability

The good news is that with the proper network, application, and services architecture, the individual devices making up the Internet do not need to support six nines of availability. All we need to do is add some redundancy. The following network design includes such a well-architected redundancy-based design. For this network design, if each element is fully independent, and if each element is available just 99.9% of the time, then the end-user will experience 99.999% availability

Unit 6: Puri Important hai

Unit 7:

1. What is the purpose of enterprise architecture model?

The main goals of enterprise architecture may be to create a map or blueprint of the structure and operations of an organization. This blueprint should include information such as a map of IT assets and business processes.

Other common goals include promoting team alignment and standardization. This can be done in part by unifying environments across teams and organizations. Guidance is normally based on an organization's business requirements.

2. zachman framework TOGAF framework

ZACHMAN FRAMEWORK

What is Zachman Framework?

Enterprise Architecture (EA) is a discipline which has evolved to structure the business and its alignment with the IT systems. 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.

Why Zachman Framework?

In today's complex business environments, many large organizations have great difficulty responding to changes. Part of this difficulty is due to a lack of internal understanding of the complex structure and components in different areas of the organization, where legacy information about the business is locked away in the minds of specific employees or business units, without being made explicit.

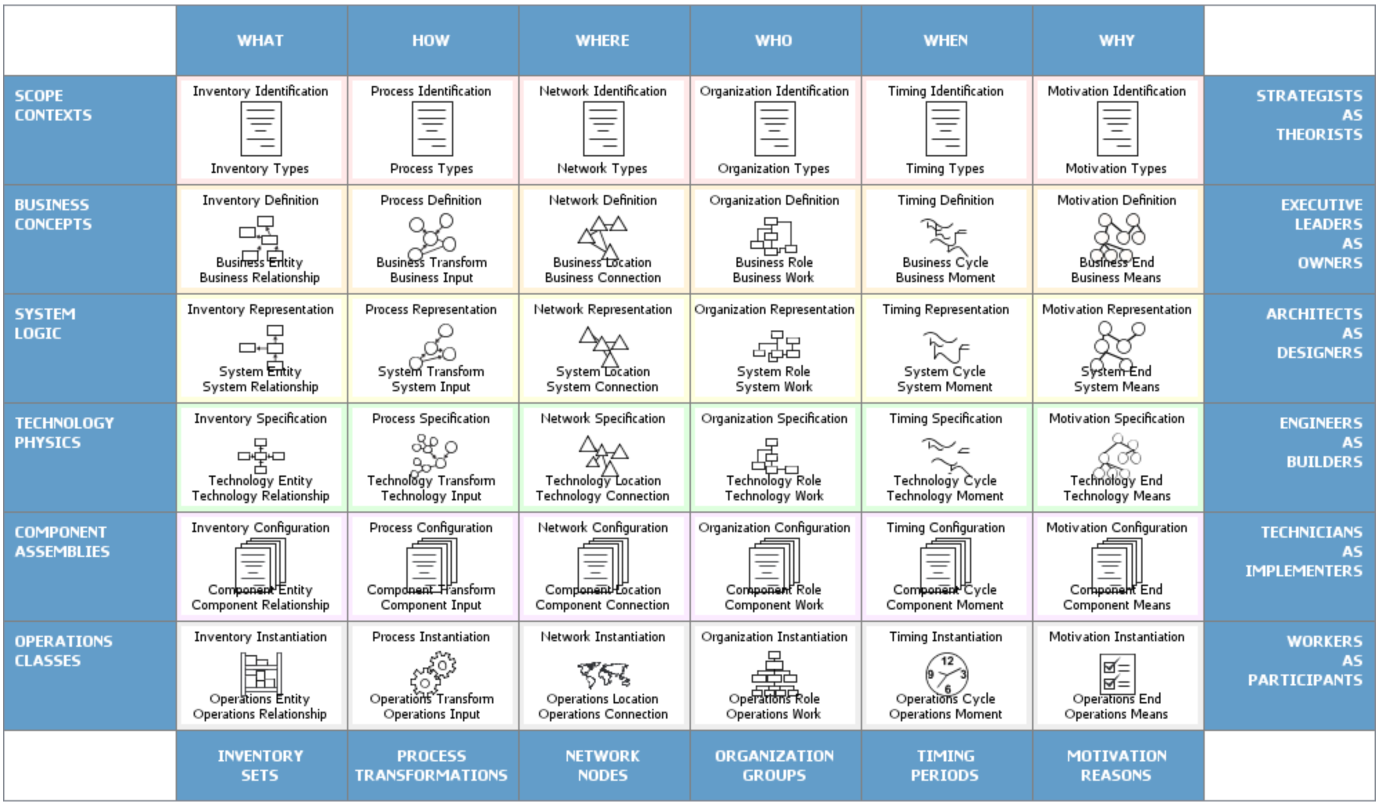
The Zachman framework provides a means of classifying an organization's architecture. **It is a proactive business tool, which can be used to model an organization's existing functions, elements and processes - and help manage business change**. The framework draws on Zachman's experience of how change is managed in complex products such as airplanes and buildings.

Structure of Zachman Framework

Zachman Framework is a two-dimensional classification scheme for descriptive representations of an Enterprise that is structured as a matrix containing **36 cells**, each of them focusing on one dimension or perspective of the enterprise. Rows are often presented as different viewpoints involved in the systems development process, while columns represent different perspectives of the stakeholders involved in the organization.

The rows of Zachman Framework focus on describing the enterprise from six viewpoint perspectives of the stakeholders. These six perspectives are based on English language interrogatives 'what', 'where', 'who', 'when', 'why', and 'how' (known as W5H).

The columns of the framework consist of a set of artifacts which are description of the enterprise from specific viewpoint of a group of stakeholders. The stakeholders are generally grouped as planners, owners, designers (architects), implementers, sub-constructors, users, or sometimes represented as viewpoints: scope context, business concepts, system logic, technology, physics, component assembles and operations classes.



	WHAT	HOW	WHERE	WHO	WHEN	WHY	
SCOPE CONTEXTS	Inventory Identification Inventory Types	Process Identification Process Types	Network Identification Network Types	Organization Identification Organization Types	Timing Identification Timing Types	Motivation Identification Motivation Types	STRATEGISTS AS THEORISTS
BUSINESS CONCEPTS	Inventory Definition Business Entity Business Relationship	Process Definition Business Transform Business Input	Network Definition Business Location Business Connection	Organization Definition Business Role Business Work	Timing Definition Business Cycle Business Moment	Motivation Definition Business End Business Means	EXECUTIVE LEADERS AS OWNERS
SYSTEM LOGIC	Inventory Representation System Entity System Relationship	Process Representation System Transform System Input	Network Representation System Location System Connection	Organization Representation System Role System Work	Timing Representation System Cycle System Moment	Motivation Representation System End System Means	ARCHITECTS AS DESIGNERS
TECHNOLOGY PHYSICS	Inventory Specification Technology Entity Technology Relationship	Process Specification Technology Transform Technology Input	Network Specification Technology Location Technology Connection	Organization Specification Technology Role Technology Work	Timing Specification Technology Cycle Technology Moment	Motivation Specification Technology End Technology Means	ENGINEERS AS BUILDERS
COMPONENT ASSEMBLIES	Inventory Configuration Component Entity Component Relationship	Process Configuration Component Transform Component Input	Network Configuration Component Location Component Connection	Organization Configuration Component Role Component Work	Timing Configuration Component Cycle Component Moment	Motivation Configuration Component End Component Means	TECHNICIANS AS IMPLEMENTERS
OPERATIONS CLASSES	Inventory Instantiation Operations Entity Operations Relationship	Process Instantiation Operations Transform Operations Input	Network Instantiation Operations Location Operations Connection	Organization Instantiation Operations Role Operations Work	Timing Instantiation Operations Cycle Operations Moment	Motivation Instantiation Operations End Operations Means	WORKERS AS PARTICIPANTS
	INVENTORY SETS	PROCESS TRANSFORMATIONS	NETWORK NODES	ORGANIZATION GROUPS	TIMING PERIODS	MOTIVATION REASONS	

The framework enables complex subjects to be distilled into systematic categories in the column headers, using these six basic questions (known as 5WH). The answers to these questions will differ, depending on the perspective or audience (represented in the rows).

Each view is a description from a particular perspective and has a representation (a model or functioning system), as indicated in the Table above. Here is a brief description of each view and model/functioning system:

Columns of Zachman Framework

The columns represent the interrogatives or questions that are asked of the enterprise. These are:

- **What** (data) - what is the business data, information or objects?
- **How** (function) - how does the business work, i.e., what are the business' processes?
- **Where** (network) - where are the businesses operations?
- **Who** (people) - who are the people that run the business, what are the business units and their hierarchy?
- **When** (time) - when are the business processes performed, i.e., what are the business schedules and workflows?
- **Why** (motivation) - why is the solution the one chosen? How was that derived from? What motivates the performance of certain activities?

Rows of Zachman Framework

Each row represents a distinct view of the organisation, from the perspective of different stakeholders. These are ordered in a desired priority sequence. A row is allocated to each of the following stakeholders:

- **Planner's View** (Scope Contexts) - This view describes the business purpose and strategy, which defines the playing field for the other views. It serves as the context within which the other views will be derived and managed.
- **Owner's View** (Business Concepts) - This is a description of the organization within which the information system must function. Analyzing this view reveals which parts of the enterprise can be automated.
- **Designer's View** (System Logic) - This view outlines how the system will satisfy the organization's information needs. The representation is free from solution specific aspects or production specific constraints.
- **Implementer's View** (Technology Physics) - This is a representation of how the system will be implemented. It makes specific solutions and technologies apparent and addresses production constraints.
- **Sub-Constructor's View** (Component Assembles) - These representations illustrate the implementation-specific details of certain system elements: parts that need further clarification before production can begin. This view is less architecturally significant than the others because it is more concerned with a part of the system than with the whole.
- **User's View** (Operations Classes) - This is a view of the functioning system in its operational environment.

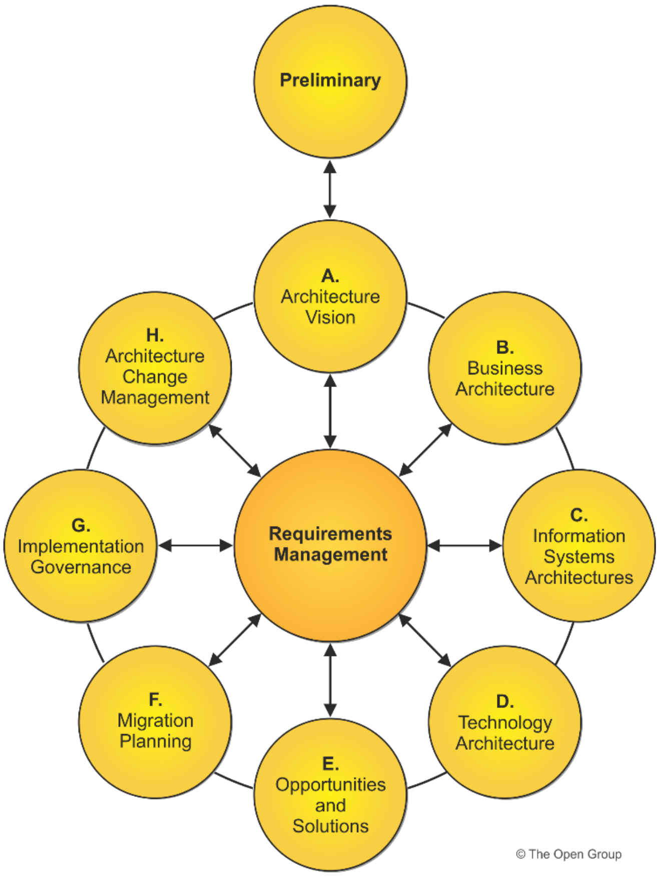
Rules of Zachman Framework

The framework offers a set of descriptive representations or models relevant for describing an enterprise.

- Each cell in the Zachman Framework must be aligned with the cells immediately above and below it.
- All the cells in each row also must be aligned with each other.
- Each cell is unique.
- Combining the cells in one row forms a complete description of the enterprise from that view.

TOGAF Framework

TOGAF is an enterprise architecture framework that helps define business goals and align them with architecture objectives around enterprise software development.


Preliminary
A. Architecture Vision
B. Business Architecture
C. Information Systems Architectures
D. Technology Architecture
E. Opportunities and Solutions
F. Migration Planning
G. Implementation Governance
H. Architecture Change Management
Requirements Management
© The Open Group

TOGAF definition

The Open Group Architecture Framework (TOGAF) is an enterprise architecture methodology that offers a high-level framework for enterprise software development. TOGAF helps organize the development process through a systematic approach aimed at reducing errors, maintaining timelines, staying on budget, and aligning IT with business units to produce quality results.

The TOGAF approach to EAFs

The Open Group defines the TOGAF as the “de factor global standard for enterprise architecture”. The framework is intended to help enterprises organize and address all critical business needs through four goals:

- **Ensuring all users, from key stakeholders to team members, speak the same language**. This helps everyone understand the framework, content, and goals in the same way and gets the entire enterprise on the same page, breaking down any communication barriers.
- **Avoiding being “locked in” to proprietary solutions for enterprise architecture**. As long as the company is using the TOGAF internally and not towards commercial purposes, the framework is free.
- **Saving time and money** and utilizing resources more effectively.
- **Achieving demonstrable return on investment (ROI).**

3 pillars of TOGAF

If the four goals are the theoretical outcome of using TOGAF, then the three pillars are the way to achieve the goals. These pillars help create a systematic process to organize and put software technology to use in a structured way that aligns with governance and business objectives. Because software develop relies on collaboration across various business departments inside and outside of

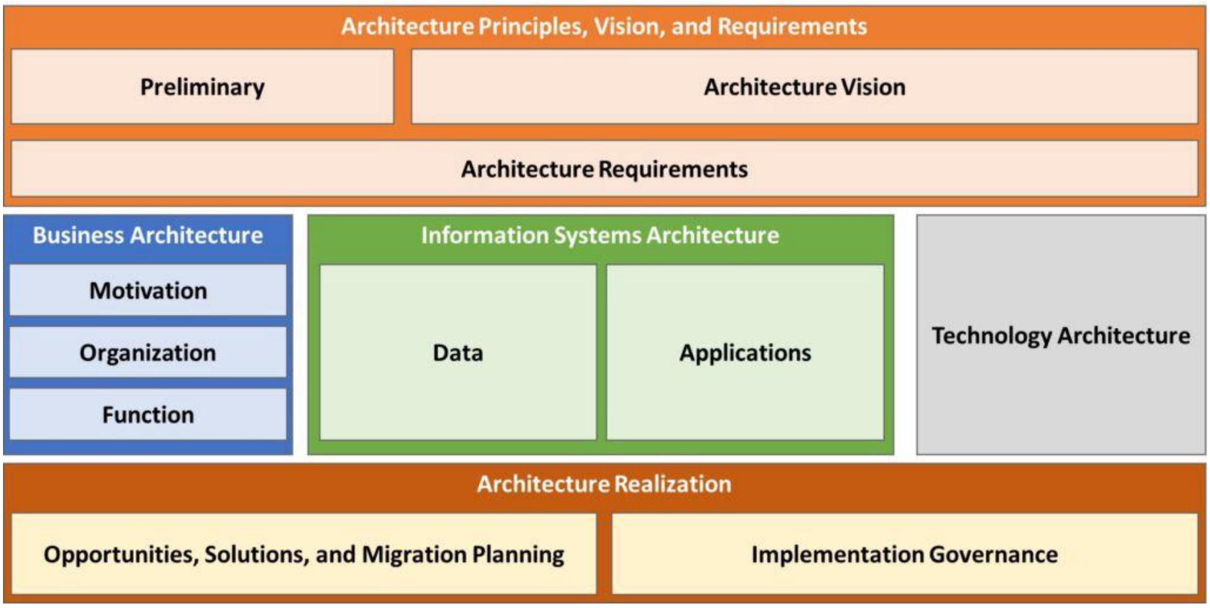
IT, TOGAF's goal of speaking the same language encourages and assists t he various stakeholders to get on the same page, something that may not otherwise happen in business environments.

The TOGAF is divided into three main pillars:

Enterprise architecture domains

Enterprise architecture domains divide the architecture into four key areas (sometimes shortened to 'BDAT areas'):

- **Business architecture**, which defines business strategy and organization, key business processes, and governance and standards.
- **Data architecture**, which documents the structure of logical and physical data assets and any related data management resources.
- **Applications architecture**, which provides a blueprint for deploying individual systems, including the interactions among application systems as well as their relationships to essential business processes.
- **Technical architecture** (also known as technology architecture), which describes the hardware, software, and network infrastructure necessary to support the deployment of mission-critical applications.


Architecture Principles, Vision, and Requirements
Preliminary
Architecture Vision
Architecture Requirements
Business Architecture
Motivation
Organization
Function
Information Systems Architecture
Data
Applications
Technology Architecture
Architecture Realization
Opportunities, Solutions, and Migration Planning
Implementation Governance

TOGAF Framework

The heart of TOGAF

Proponents say that ADM is the heart of TOGAF: it's this pillar that makes TOGAF both very effective and a standout from other frameworks. The Architecture Development Method offers eight steps as guidance to figure out where the enterprise currently is and determine where the enterprise wants and needs to be in each of the four enterprise architecture domains.

Once business processes are established through the entire lifecycle, the ADM helps the enterprise to:

1. Identify the gaps between current status and long-term goals.
2. Collate these gaps into smaller actionable and understandable packages that the team can then implement.

Two other areas are sometimes included in TOGAF's main pillars:

- TOGAF certified tools
- Qualifications

The Open Group offers two certifications for individuals:

- The first level is known as the Foundation, teaching basic tenets of enterprise architecture and rolling out TOGAF.
- Level 2 Certified involves business analysis and application.

The Open Group also certifies tools that align with TOGAF. For the most recent version, eight tools from eight organization are available.

Benefits of using TOGAF

The benefits of ADM are that it is customizable to organizational need—there's no need to create a structure that doesn't serve your business. These smaller packages are also scalable, so if one team rolls it out, it can successfully be rolled out to other teams without much tweaking. This helps the enterprise establish a process with multiple check points, so that there are few errors the wider the architecture is implemented.

There can also be benefits to individuals who certify in TOGAF. A study of industry employees indicates that enterprise architects, software architects, and IT directors, among others, who choose to earn a certification in TOGAF often see an average yearly pay bump of $10,000 to $20,000 over similarly placed colleagues who aren't certified.

Some experts in enterprise architecture point out that while TOGAF may appear very logical, it's actually quite a shake up to traditionally educated technology consultants today – but perhaps this will change has TOGAF adoption continues along steadily.

3. Kind/Main of pillars TOGAF

Enterprise architecture domains

TOGAF is based on four interrelated areas of specialization called architecture domains:

- **Business architecture** which defines the business strategy, governance, organization, and key business processes of the organization
- **Data architecture** which describes the structure of an organization's logical and physical data assets and the associated data management resources
- **Applications architecture** which provides a blueprint for the individual systems to be deployed, the interactions between the application systems, and their relationships to the core business processes of the organization with the frameworks for services to be exposed as business functions for integration
- **Technical architecture**, or technology architecture, which describes the hardware, software, and network infrastructure needed to support the deployment of core, mission-critical applications

Architecture Development Method

The Architecture Development Method (ADM) is applied to develop an enterprise architecture which will meet the business and information technology needs of an organization. It may be tailored to the organization's needs and is then employed to manage the execution of architecture planning activities.

The process is iterative and cyclic. Each step checks with Requirements. Phase C involves some combination of both Data Architecture and Applications Architecture. Additional clarity can be added between steps B and C in order to provide a complete information architecture.

Performance engineering working practices are applied to the Requirements phase, and to the Business Architecture, Information System Architecture, and Technology architecture phases. Within Information System Architecture, it is applied to both the Data Architecture and Application Architecture.

Enterprise Continuum

The Enterprise Continuum is a way of classifying solutions and architectures on a continuum that range from generic foundation architectures through to tailored organization-specific both within and outside the Architecture Repository. These include architectural models, architectural patterns, architecture descriptions, and other artifacts. These artifacts may exist within the enterprise and also in the IT industry at large.

The Enterprise Continuum consists of both the Architecture Continuum and the Solutions Continuum. The Architecture Continuum specifies the structuring of reusable architecture assets and includes rules, representations, and relationships of the information systems available to the enterprise. The Solutions Continuum describes the implementation of the Architecture Continuum by defining reusable Solution Building Blocks (SBBs).