2m Questions

1. What is Microservice

It modularizes complex applications into distributed parts that run parallel without harming another part of the applications. The team of developers composes the overall applications by resulting in the upgradable, interchangeable, and critical scalable parts of the application. When it comes to the rapid application development era, such a modular architectural style surely helps accelerate business growth by enabling the agile deployment of rapid client functionalities.

2. What is Plugin Architecture

A plug-in is a bundle that adds functionality to an application, called the host application, through some well-defined architecture for extensibility. This allows third-party developers to add functionality to an application without having access to the source code.

The plug-in architecture consists of two components: a core system and plug-in modules.

The main key design here is to allow adding additional features as plugins to the core application, providing extensibility, flexibility, and isolation of application features and customs processing logic.

3. What is mapping of the database

Data mapping is the process of matching fields from one database to another. It's the first step to facilitate data migration, data integration, and other data management tasks.

Data migration

Data migration is the process of moving data from one system to another as a one-time event.

Data integration

Data integration is an ongoing process of regularly moving data from one system to another.

Data transformation

Data transformation is the process of converting data from a source format to a destination format.

Data warehousing

If the goal is to pool data into one source for analysis or other tasks, it is generally pooled in a data warehouse.

4. What is a web presentation

Enterprise Web Applications allow you to manage and record the internal and external operations and processes of your company or organization.

5. What is concurrency

Concurrency is also synonymous with boundary-less organization as it has been demonstrated with concurrent engineering implementation that breaks down barriers between disciplines to enable collaboration.

6. What is an enterprise integration

Enterprise integration is the use of multiple integration approaches, including API management, application integration and messaging to leverage enterprise services and assets in order to expose them as APIs or connect them as services. This enables organizations to seamlessly integrate, unify and standardize core business capabilities across diverse IT environments.

7. What is message-based integration

Message-based Integration is one of the common categories of patterns for connectivity solutions.

An Enterprise Service Bus can extend an existing messaging infrastructure by providing an environment for building and deploying message-based applications at the infrastructure level. Examples of these applications include routing and transformation services, and logging services.

8. What is WSDL

Web Services Description Language (WSDL) is a standard specification for describing networked, XML-based services. It provides a simple way for service providers to describe the basic format of requests to their systems regardless of the underlying run-time implementation.

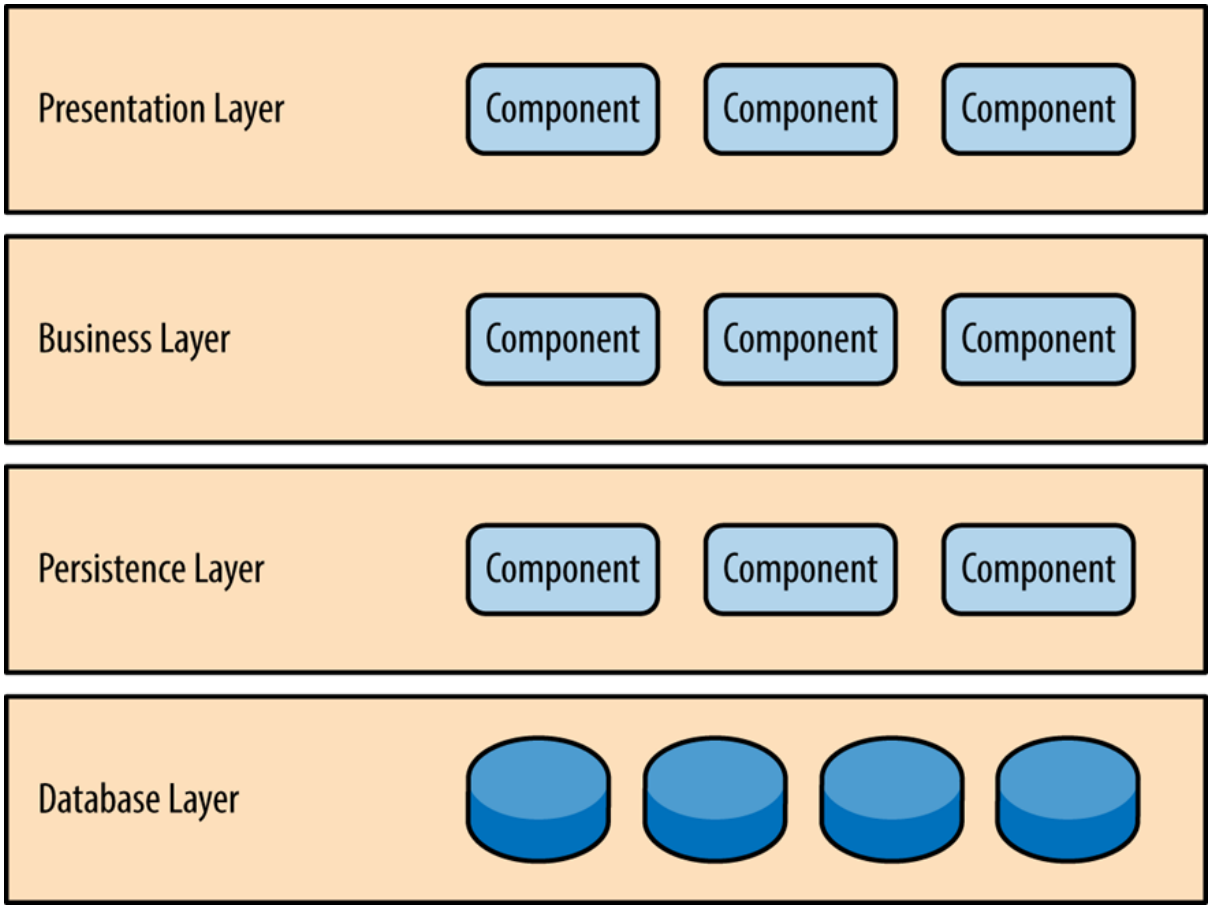
WSDL defines an XML format for describing network services as a set of endpoints that operate on messages that contain either document-oriented or procedure-oriented information.

4m Questions

1. Explain Layered Architecture

The most common architecture pattern is the layered architecture pattern, otherwise known as the n-tier architecture pattern. This pattern is the de facto standard for most Java EE applications and therefore is widely known by most architects, designers, and developers. The layered architecture pattern closely matches the traditional IT communication and organizational structures found in most companies, making it a natural choice for most business application development efforts.

Each layer of the layered architecture pattern has a specific role and responsibility within the application. For example, a presentation layer would be responsible for handling all user interface and browser communication logic, whereas a business layer would be responsible for executing specific business rules associated with the request. Each layer in the architecture forms an abstraction around the work that needs to be done to satisfy a particular business request. For example, the presentation layer doesn't need to know or worry about how to get customer data; it only needs to display that information on a screen in particular format. Similarly, the business layer doesn't need to be concerned about how to format customer data for display on a screen or even where the customer data is coming from; it only needs to get the data from the persistence layer, perform business logic against the data (e.g., calculate values or aggregate data), and pass that information up to the presentation layer.



2. What is an event-driven architecture

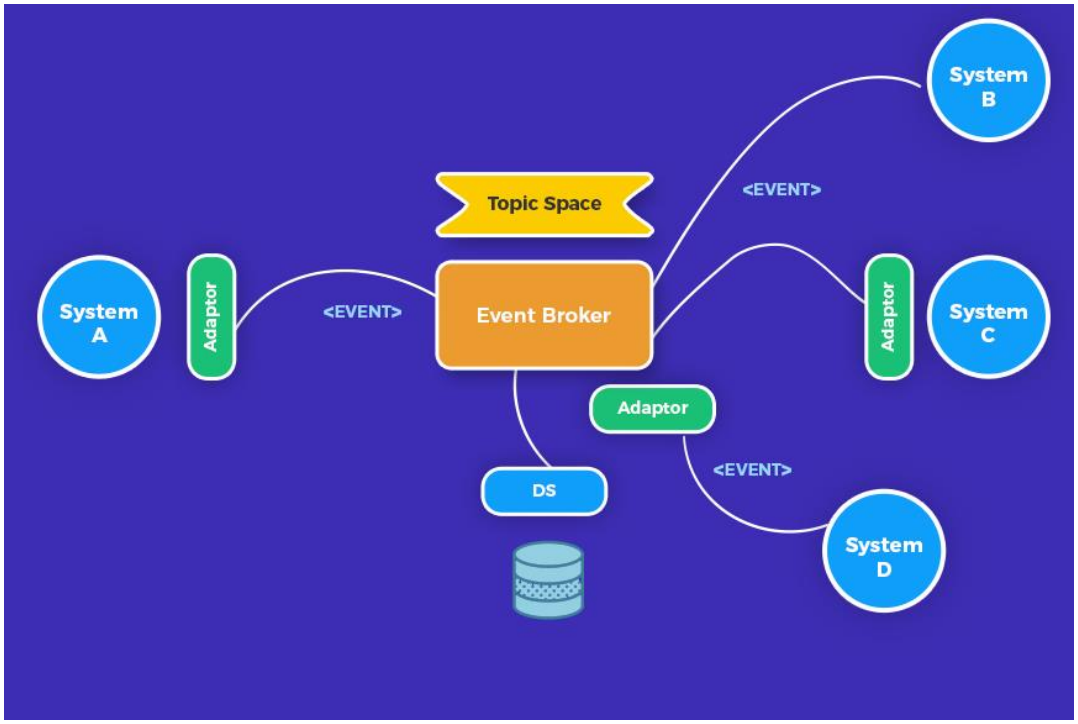
Event-driven architecture (EDA) is a design paradigm in which a software component executes in response to receiving one or more event notifications.

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.

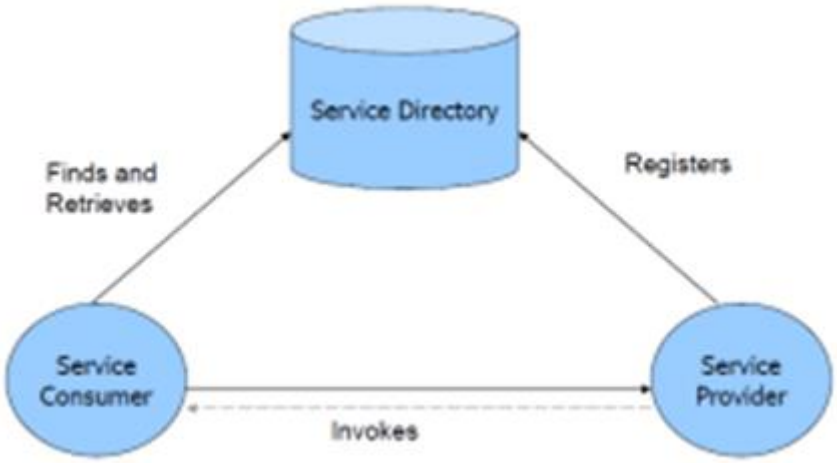


3. What is Service-oriented architecture

- Service-Oriented Architecture (SOA) is an architectural approach for designing and developing applications that are based on loosely-coupled services.
- The primary benefit of SOA is the reusability of services, which enables rapid development and modification of software that supports business operations, leading to enterprise agility.
- SOA also allows for flexible, combined business processes, business process optimization, and Real Time Enterprise (RTE).
- SOA involves organizing all software into business services that are network accessible and executable, with service interfaces based on public standards for interoperability.
- SOA enables distributed deployment by exposing enterprise data and business logic as loosely coupled, discoverable, structured, standards-based, coarse-grained, stateless units of functionality called services.
- SOA facilitates reusability by allowing businesses to choose a service provider and access existing resources exposed as services, thereby leveraging existing investments.
- SOA also enables composability by allowing the assembly of new processes from existing services exposed at a desired granularity through well-defined, published, and standards-compliant interfaces.
- SOA provides interoperability by sharing capabilities and reusing shared services across a network irrespective of underlying protocols or implementation technology.

Key characteristic of SOA:

1. quality of service- response time
2. security and performance
3. service is catalogued and discoverable
4. data are catalogued and discoverable
5. protocols use only industry standards



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.

4. What are Data Transformation and Data Warehousing, and what are the steps of data mapping

Data transformation

Data transformation is the process of converting data from a source format to a destination format. This can include cleansing data by changing data types, deleting nulls or duplicates, aggregating data, enriching the data, or other transformations.

Data Warehouse

If the goal is to pool data into one source for analysis or other tasks, it is generally pooled in a data warehouse.

Step 1: Define — Define the data to be moved, including the tables, the fields within each table, and the format of the field after it's moved. For data integrations, the frequency of data transfer is also defined.

Step 2: Map the Data — Match source fields to destination fields.

Step 3: Transformation — If a field requires transformation, the transformation formula or rule is coded.

Step 4: Test — Using a test system and sample data from the source, run the transfer to see how it works and make adjustments as necessary.

Step 5: Deploy — Once it's determined that the data transformation is working as planned, schedule a migration or integration go-live event.

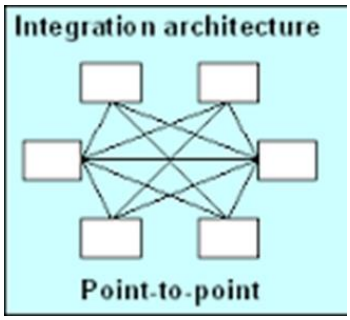
Step 6: Maintain and Update — For ongoing data integration, the data map is a living entity that will require updates and changes as new data sources are added, as data sources change, or as requirements at the destination change.

5. Types of enterprise application integration

Point-to-point integration.

This is the simplest form of EAI. In it, data is taken from one source, perhaps reformatted, and then ingested by the next application. These are often simple to implement for small workflows and a few tools. They can quickly grow large and difficult to manage as more applications and integrations are added, however, and can become slow as a backlog or slowdown in one system affects others in the line.

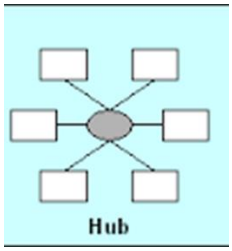
As an example, an organization may need to update a human resources database with information from an ERP system.



Hub-and-spoke integration.

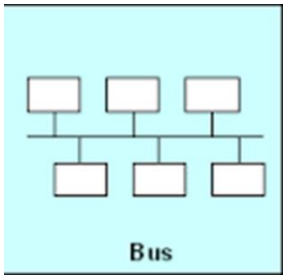
This approach uses a central program to facilitate the data and steps between the participation applications. The program can handle the data reformatting and keep workflows moving in the event of an application slowdown. Hub-and-spoke is therefore faster and more reliable than point-to-point but requires development time and effort to set up and maintain.

For example how airlines operate out of a centralized hub and use regional airports as the spokes from which they offer flights



Bus integration.

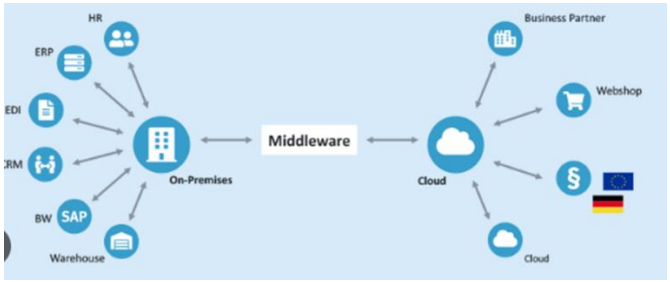
This is an evolution of hub-and-spoke design EAI. It is also called an enterprise service bus (ESB). In a common bus design, all participating applications use a set of standards to send and receive data or workflows. This allows for quick and easy integration but requires work during the planning and product selection phase.



Middleware integration.

This involves an intermediary program that sits between the end user and the underlying application. Middleware supports interface integration and may have an underlying hub-and-spoke or bus design.

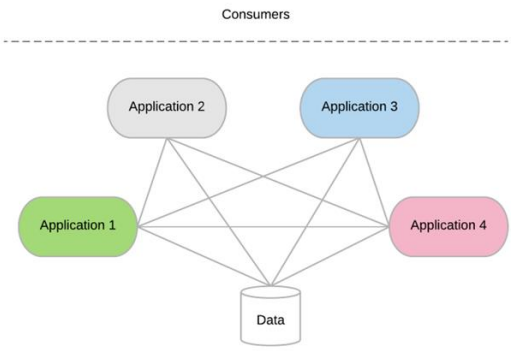
Common middleware examples include database middleware, application server middleware, message-oriented middleware, web middleware, and transaction-processing monitors.



Microservices

These are small, single-purpose tools that support EAI initiatives. These can be serverless functions or dedicated apps designed to integrate easily or quickly connect programs. Microservices can often be easily offloaded as cloud workloads.

Some of the most innovative and profitable microservices architecture examples among enterprise companies in the world — like Amazon, Netflix, Uber, and Etsy



6. What are modern service integration techniques (Explanation and Pros and Cons of the same)

1. API Integration

Application Programming Interface (API) is the most common tool for connecting different applications for service management software. There are many different types of API that are either public, partner, or private. What they all have in common is how they enable interaction between applications. An API uses a common code language to specify functionality and set protocols. This gives your applications the ability to transfer data.

Pros:

- Highly Flexible: Even though you are dependent on the developer resources, specific data becomes highly flexible because the integration uses product code.
- App Changes Aren't Disruptive: Service providers offer better functionality that goes uninterrupted since APIs are often limited in scope.
- Widely Available: As stated earlier, API is the most common tool for third-party integration. So, it will be unlikely that you run into a service that won't offer API integration options.

Cons:

- Dependent on Vendor: Vendors are responsible for creating APIs. So, you are reliant on the vendor to create APIs for the specific type of information you are trying to pull.
- Code-Intensive: Because they are code-based, APIs need an understanding of programming languages to install.

2. Web-hooks

Web hooks or HTTP call backs are an alternative to API integration. They are both tools that link to a web application but have two key differences. For web hooks, implementation is often not code-based. They often have modules that are programmable within a web application. Instead of being request-based, web hooks are event-based. They only trigger when specific events occur within a third-party service.

Pros:

- Real-Time Data: Web hooks don't use a request-based system. They allow your team to view data on a real-time scale.
- Supports Automation Efforts: Because data requests are event-based, you don't have to set up poll timings to your data centres. This can help streamline data flow and automation.

Cons:

- Limits Data Manipulation: A webhook requires the service to trigger a data transfer based on an update. In contrast to webhooks, APIs can list, create, edit, or delete an item without triggering a transfer.

3. ISC

Integration Services Component (ISC) lives on a local server unlike code-based integrations. The ISC creates a bridge with on-premise tools such as directories, asset management tools, and BI tools without the need for file imports.

Pros:

- (Near) Out-of-the-Box Solution: The ISC immediately offers many data synchronization options you would likely use.
- Wider Range of Functionality: With an ISC, you have complete data access that you can do anything with. Any data that you can access on the backend with your cloud service will be available.

Cons:

- Knowledge of Database Architecture Necessary: If you are unfamiliar with how your local database is set up, implementing an ISC will be challenging.
- Requires Access to the Backend of Your Applications: There will be many cases where backend access isn't there for your team, so you won't be able to use an ISC in those situations.

4. ORCHESTRATION

The most automated integration option is orchestrations. If you are not familiar with orchestrations, they refer to the process of automating multiple systems and services together. Teams will often use software configuration management tools such as PowerShell to build orchestrations. Software configuration management tools offer various methods such as snap-ins or hosting APIs to connect with applications to manage the automation workflow.

Pros:

- Full Automation: Automation across all processes.
- Manages Multiple Systems: Ability to manage the integrations of multiple systems collectively.

Cons:

- Code-Intensive: You need to have coding skills to manage your software configuration management tool.
- Labour-Intensive: Because the workflows are quite complex, the setup can be a drawn-out process. Also, any asset or process changes force you to check how it will affect your orchestrations.

7. Difference between SOAP and REST

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

8. What are the Key elements of enterprise integration

Application programming interfaces (APIs)

process data transfers between different systems. Situated between an application and web server, they enable companies to share the data and functionality of their applications with third-party developers, business partners and internal departments. With APIs increasingly used to access and expose real-time data, this can be extended to more sources, such as data published as events.

Application integration

is the enablement of individual applications — each designed for a specific purpose — to work collaboratively. By making it easier to share data and combine workflows and processes, organizations can benefit from integrations that modernize infrastructures without rework. Furthermore, application integration helps on-premises systems and cloud-based enterprise systems like CRMs and ERPs interact successfully without major changes to existing applications.

Messaging

helps provide resilience and performance to IT environments spanning cloud and on-premises systems. Messaging must cross network boundaries to provide reliable delivery while preserving network-wide message integrity, data protection and regulatory compliance via security-rich functions.

Events

are records of action or change. When one application or service performs an action or undergoes a change relative to the functionality of another application or service, the first one publishes an event. Other applications or services can detect the event publication. They can then process the event, perform one or more reciprocal action or simply ignore the event.

Data

specifically real-world operational data, enables continuous improvement (CI) of enterprise architecture. Data is also used to assess the criticality and usage of integrations and determine their target state. When analysed, data reveals recommended target integration patterns (e.g., service-oriented architecture (SOA), event-driven, message-driven, etc.), consolidation possibilities and other inputs that help define the target integration state.