Day1

15-7-2024

SOFTWARE

What is Software?

Software is a set of instructions, data or programs used to operate computers and execute specific tasks. It is the opposite of hardware, which describes the physical aspects of a computer. Software is a generic term used to refer to applications, scripts and programs that run on a device. It can be thought of as the variable part of a computer, while hardware is the invariable part.

The two main categories of software are:

- Application software
- System software.

Application Software

Application software refers to software that performs specific functions for a user. When a user interacts directly with a piece of software, it is called application software. An application is software that fulfills a specific need or performs tasks. The most frequently used software is application software, which is a computer software package that performs a specific function for a user or, in some cases, for another application. An application can be self-contained, or it can be a group of programs that run the application for the user.

Examples for Application Software:

- Office Suites
- Graphics Software
- Databases
- Database Management Programs
- Web Browsers
- Word Processors
- Software Development Tools
- Image Editors
- Communication Platforms.
- Application for billing

Need for Application software:

- Helps the user in completing specified tasks
- Manages and manipulates data
- Allows users to effectively organize information

System Software

System software is designed to run a computer's hardware and provides a platform for applications to run on top. These software programs are designed to run a computer's application programs and hardware. System software coordinates the activities and functions of the hardware and software. In addition, it controls the operations of the computer hardware and provides an environment or platform for all the other types of software to work in. An operating system is the best example of system software; it manages all the other computer programs. Other examples of system software include firmware, computer language translators and system utilities.

Example for System software:

- MacOs
- Android
- Chrome OS
- Firmware

Need for System software:

- Hardware Communication
- Resource Management:
- Security
- Application Support

Day 2

16-07-2024

Architectural Styles:

An architectural style is a set of principles and patterns for organization of a software system. It says how the components and modules within the system interact and communicate.

Types of Architectural styles:

- Layered
- Client Server
- Micro Service
- Event Driven
- Service Oriented

Service Oriented Architecture:

Service-oriented architecture (SOA) is a method of software development that uses software components called services to create business applications. Each service provides a business capability, and services can also communicate with each other across platforms and languages. Developers use SOA to reuse services in different systems or combine several independent services to perform complex tasks.

Key principles of SOA:

- Standardized service contract
- Loose coupling
- Abstraction
- Reusability
- Autonomy
- Statelessness
- Discoverability
- Composability

Major components of SOA:

Service Provider:
The service provider creates, maintains, and provides one or more services that others can use. Organizations can create their own services or purchase them from third-party service vendors.

Service consumer:

The service consumer requests the service provider to run a specific service. It can be an entire system, application, or other service. The service contract specifies the rules that the service provider and consumer must follow when interacting with each other. Service providers and consumers can belong to different departments, organizations, and even industries.

Service registry:

A service registry, or service repository, is a network-accessible directory of available services. It stores service description documents from service providers. The description documents contain information about the service and how to communicate with it. Service consumers can easily discover the services they need by using the service registry.

Case study on service oriented architecture:

- **CASE-STUDY SOA DESIGN AND DEVELOPMENT FOR BEPET**

Abstract:

Service Level agreement has given a new idea about the business where two organizations can work together on the based on some contract. The bepet which is an Energy company want to sign an agreement with IT department which can make Bepet services better for the customer and IT Company tries to fulfill all the requirements of Bepet. SOA is an IT control policy follows in institution. The main concept of SOA is to provide services on SOA Governance from beginning to the end. There are many steps to complete the service level agreement which is needed for one organization and it starts from Service Broker which want to take services from the administration and send request and details about the service to the organizations. SOA supervisor which is main body of the service level agreement consult with SOA registry to send all the details about the service because they have established the monitoring system.

Introduction:

The main concept of SOA is to provide services on SOA Governance from beginning to the end approach. The lifecycle of services are development, deployment and retirement of service framed by the process, policies, procedures, roles, and responsibilities for design-time governance and runtime governance. Here we are going to how SOA helped.

Service components:

- Home-based Saving Energy
- Saving Energy Guidelines
- Energy efficient Product
- Gas Boiler
- Repairing Boiler Service

Bepet Service Oriented Architecture Service:

The three major services of Bepet business application services are:

- Human decision services
- Information Services
- Functional Service

Bepet Association Services:

The utility service of the Bepet: The utilities are a part of the organizations which are tiny, minor, and inseparable and frequently used in organization framework. The main objective these services to build a big organization where many services needed for long time.

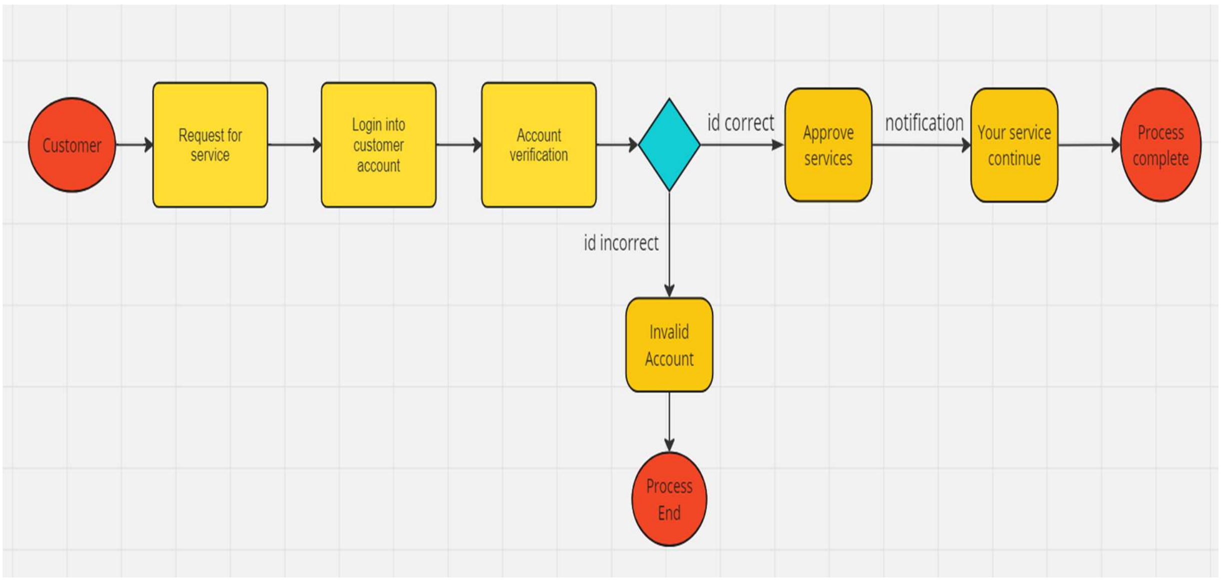
Activities services of Bepet: All those services which comes under the establishment of the building square referred to as a task to the organization. The activities service is independent any individual task of the organization

Conclusion

The overall discussion is based on providing services to the organization and research also set an example of the Bpet which show that how can implement the services and what characteristics should be.

Link for the case study - https://www.ijnrd.org/papers/IJNRD1812007.pdf

UML diagram for SOA :



Event Driver Architecture

An event-driven architecture uses events to trigger and communicate between decoupled services and is common in modern applications built with microservices. An event is a change in state, or an update, like an item being placed in a shopping cart on an e-commerce website. Events can either carry the state (the item purchased, its price, and a delivery address) or events can be identifiers like a notification that an order was shipped.
Event-driven architectures have three key components:

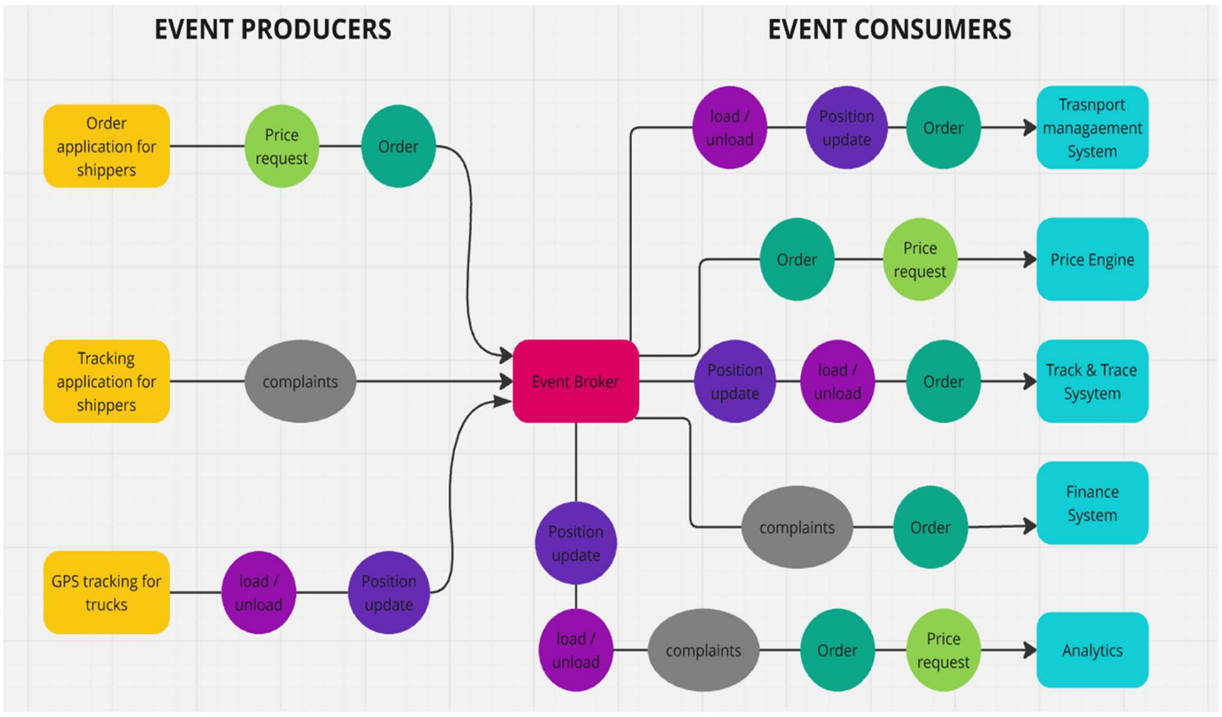
- Event producers
- Event routers
- Event consumers.

Case study on event driven architecture:

- **Event-Driven Architecture in Logistics Company. Case Study of EDA in Modern Supply Chain Management**

Link for the case study - https://nexocode.com/blog/posts/event-driven-architecture-in-logistics-case-study/

UML diagram for above case study on Event driver architecture:



Day 3:

17-7-2024

Presentation on the topic of case study on Service Oriented Architecture (SOA)

Topic of the case study:

- **CASE-STUDY SOA DESIGN AND DEVELOPMENT FOR BEPET**

Presentation link:

https://docs.google.com/presentation/d/1KzBfG1NFnMSBRTPlkGhG4KXjR9IFEUJu/edit?usp=sharing&ouid=110022321659518235286&rtpof=true&sd=true

Day 4 :

18-7-2024

Software design pattern

What is software design pattern?

Software design patterns are essential tools for developers, providing proven solutions to common problems encountered during software development. By understanding and applying these patterns, developers can create more robust, maintainable, and scalable software systems.

Key Characteristics of Design Patterns

- Reusability
- Standardization
- Efficiency
- Flexibility

Types of software design pattern

There are three types of Design Patterns:

- Creational Design Pattern
- Structural Design Pattern
- Behavioral Design Pattern

Creational Design Pattern

Creational Design Pattern abstract the instantiation process. They help in making a system independent of how its objects are created, composed and represented.

Structural Design Pattern

Structural Design Patterns are concerned with how classes and objects are composed to form larger structures. Structural class patterns use inheritance to compose interfaces or implementations.

Behavioral Design Pattern

Behavioral Patterns are concerned with algorithms and the assignment of responsibilities between objects. Behavioral patterns describe not just patterns of objects or classes but also the patterns of communication between them. These patterns characterize complex control flow that’s difficult to follow at run-time.

Model View Controller (MVC)

What is MVC?

The Model View Controller (MVC) design pattern specifies that an application consists of a data model, presentation information, and control information. The pattern requires that each of these be separated into different objects.

The MVC pattern separates the concerns of an application into three distinct components, each responsible for a specific aspect of the application’s functionality. This separation of concerns makes the application easier to maintain and extend, as changes to one component do not require changes to the other components.

Components of MVC

Model:

The Model component in the MVC design pattern represents the data and business logic of an application. It is responsible for managing the application’s data, processing business rules, and responding to requests for information from other components, such as the View and the Controller.

View:

Displays the data from the Model to the user and sends user inputs to the Controller. It is passive and does not directly interact with the Model. Instead, it receives data from the Model and sends user inputs to the Controller for processing.

Controller:

Controller acts as an intermediary between the Model and the View. It handles user input and updates the Model accordingly and updates the View to reflect changes in the Model. It contains application logic, such as input validation and data transformation.

Advantage of MVC

- Separation of Concerns
- Modularity
- Flexibility
- Parallel Development
- Code reusability

Disadvantage of MVC

- Complexity
- Learning Curve

- Potential for Over-Engineering
- Increased File Count
- Overhead

Cloud Computing

What is cloud computing?

Cloud Computing means storing and accessing the data and programs on remote servers that are hosted on the internet instead of the computer's hard drive or local server. Cloud computing is also referred to as Internet-based computing, it is a technology where the resource is provided as a service through the Internet to the user. The data that is stored can be files, images, documents, or any other storable document.

SaaS (Software as a Service)

- Collaboration and Accessibility: Software as a Service (SaaS) helps users to easily access applications without having the requirement of local installations. It is fully managed by the AWS Software working as a service over the internet encouraging effortless cooperation and ease of access.

- Automation of Updates: SaaS providers manage the handling of software maintenance with automatic latest updates ensuring users gain experience with the latest features and security patches.

- Cost Efficiency: SaaS acts as a cost-effective solution by reducing the overhead of IT support by eliminating the need for individual software licenses.

Paas (Platform as a Service)

- Simplifying the Development: Platform as a Service offers application development by keeping the underlying Infrastructure as an Abstraction. It helps the developers to completely focus on application logic and background operations are completely managed by the AWS platform.

- Enhancing Efficiency and Productivity: PaaS lowers the Management of Infrastructure complexity, speeding up the Execution time and bringing the updates quickly to market by streamlining the development process.

- Automation of Scaling: Management of resource scaling, guaranteeing the program's workload efficiency is ensured by PaaS.

IaaS (Infrastructure as a Service)

- Flexibility and Control: IaaS comes up with providing virtualized computing resources such as VMs, Storage, and networks facilitating users with control over the Operating system and applications.
- Reducing Expenses of Hardware: IaaS provides business cost savings with the elimination of physical infrastructure investments making it cost-effective.
- Scalability of Resources: The cloud provides in scaling of hardware resources up or down as per demand facilitating optimal performance with cost efficiency.

Dockers

What is docker?

Docker is a tool designed to make it easier to create, deploy, and run applications by using containers. Containers are lightweight, portable, and ensure that an application runs the same way regardless of where it is deployed.

Docker container

Docker container is a runtime instance of an image. Allows developers to package applications with all parts needed such as libraries and other dependencies. Docker Containers are runtime instances of Docker images. Containers contain the whole kit required for an application, so the application can be run in an isolated way. For e.g.- Suppose there is an image of Ubuntu OS with NGINX SERVER when this image is run with the docker run command, then a container will be created and NGINX SERVER will be running on Ubuntu OS.

Docker Image

It is a file, comprised of multiple layers, used to execute code in a Docker container. They are a set of instructions used to create docker containers. Docker Image is an executable package of software that includes everything needed to run an application. This image informs how a container should instantiate, determining which software components will run and how. Docker Container is a virtual environment that bundles application code with all the dependencies required to run the application. The application runs quickly and reliably from one computing environment to another.

Docker hub

Docker Hub is a repository service and it is a cloud-based service where people push their Docker Container Images and also pull the Docker Container Images from the Docker Hub anytime or anywhere via the internet. Generally, it makes it easy to find and reuse images. It provides features such as you can push your images as private or public registry where you can store and share Docker images.

Mainly DevOps team uses the Docker Hub. It is an open-source tool and freely available for all operating systems. It is like storage where we store the images and pull the images when it is required. When a person wants to push/pull images from the Docker Hub they must have a basic knowledge of Docker. Let us discuss the requirements of the Docker tool.

Kubernetes

What is Kubernetes?

Kubernetes is a container orchestration platform that automates the deployment, scaling, and operations of application containers across clusters of machines. It abstracts the underlying infrastructure and provides a unified API for managing containerized applications.

Key Features of Kubernetes

- Automated Deployment and Scaling: Automatically deploy, manage, and scale applications.
- Self-Healing: Automatically replaces failed containers and restarts applications.
- Load Balancing: Distributes traffic across containers.
- Storage Orchestration: Manages storage resources for applications.
- Service Discovery: Provides DNS names for services and load balances across them.
- Configuration Management: Manages configuration and secrets for applications.
- Rolling Updates: Performs rolling updates to applications with zero downtime.