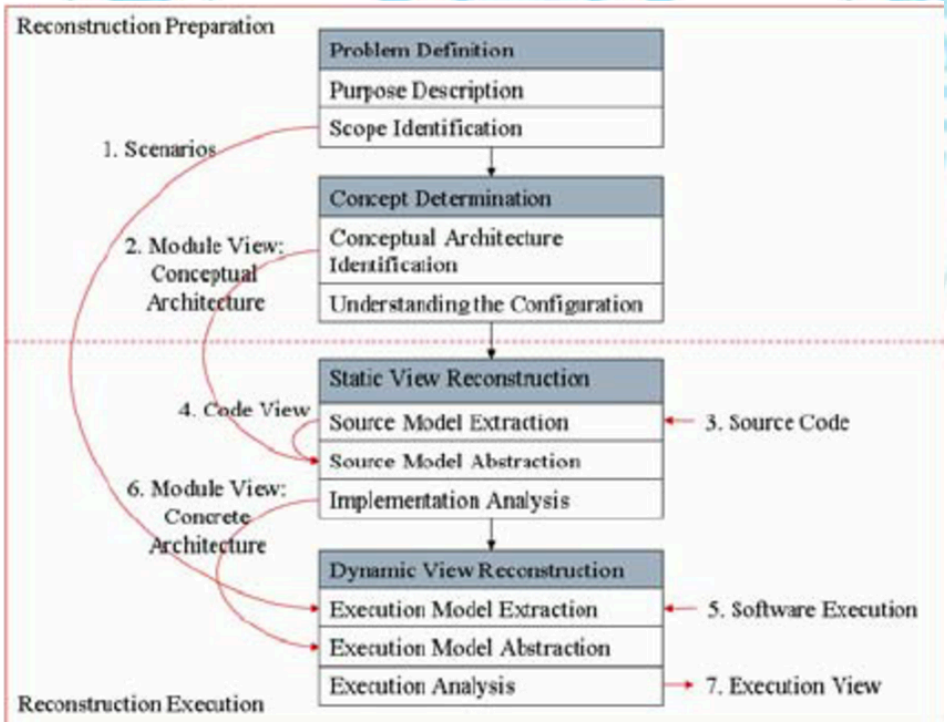
## Case Study: Architecture reconstruction

**Aim:** To modernize the legacy e-commerce platform of ShopVerse by reconstructing its monolithic architecture into a scalable, efficient, and maintainable microservices-based system, ensuring improved performance, real-time capabilities, and future readiness with minimal disruption to ongoing operations.

**Theory:**

An established e-commerce company, **ShopVerse**, had been operating a successful online platform for over a decade. Initially, the platform was built using monolithic architecture and traditional technologies like PHP, MySQL, and a simple front-end framework. Over time, as the company expanded, the system became harder to maintain, suffered from performance bottlenecks, and could not scale effectively to handle peak traffic. With new business requirements, including the introduction of mobile applications, real-time inventory management, and personalized user experiences, the company decided to reconstruct the architecture of its platform. The aim was to transition from a monolithic to a microservices-based architecture to address scaling issues and future-proof the platform.



**The Problem**

ShopVerse's legacy system had become difficult to maintain and scale for the following reasons:

1. **Performance Issues**: The system could not handle large numbers of concurrent users effectively, leading to slow response times, particularly during high-demand periods like Black Friday.
2. **Limited Scalability**: The platform was hosted on a single, vertically scaled server. It could not be easily distributed, making it difficult to scale horizontally as traffic and data increased.
3. **High Maintenance Cost**: The monolithic architecture led to a tightly coupled codebase, which made implementing new features or fixing bugs a slow and error-prone process.
4. **Lack of Flexibility**: Integrating new technologies like mobile apps and AI-based product recommendations was challenging due to the rigidity of the existing architecture.
5. **No Real-Time Capabilities**: Features such as real-time inventory updates and personalized shopping experiences were difficult to implement in the current architecture.

**Objective**

The goal of the architectural reconstruction was to:

- Improve performance and scalability.
- Enable real-time features and personalized experiences.
- Reduce the time and cost of maintenance.
- Support future integrations with new technologies.
- Ensure a smooth transition with minimal downtime for the business.

**Approach: Architecture Reconstruction**

**Architecture reconstruction** involves reverse-engineering the existing architecture to analyze and understand the current system, followed by refactoring and reorganizing the components into a new architectural style. The following steps were taken to reconstruct the architecture for ShopVerse:

**Application:**

## 1. Assessment of the Existing Architecture

The first step was to assess and document the current state of the system. This involved:

- **Dependency Analysis**: Identifying key modules and their interdependencies in the monolithic system. Tools like *SonarQube* were used to visualize the system's dependency structure and pinpoint tightly coupled components.

- **Performance Bottleneck Identification**: Using APM (Application Performance Monitoring) tools like *New Relic* and *Datadog*, the team identified performance bottlenecks, such as slow database queries, heavy API calls, and inefficient code loops.
- **Data Flow Mapping**: Mapping out how data flowed between various modules, databases, and external systems like payment gateways.

This assessment revealed that the platform's checkout, search, and inventory systems were the most problematic in terms of performance and scalability.

## 2. Defining the New Architecture

The team decided to adopt a **microservices-based architecture**. Each key feature of the e-commerce platform (e.g., product catalog, user management, inventory, checkout) would be transformed into a separate microservice that could be independently developed, deployed, and scaled.

Key components of the new architecture included:

- **Microservices**: Each business function (search, payment, recommendation engine, etc.) was divided into independent services, allowing for better scalability and maintainability.
- **API Gateway**: To unify communication between the frontend (web and mobile apps) and backend services, an API gateway was introduced. This allowed secure and efficient communication with microservices.
- **Message Broker**: Real-time updates for inventory and order processing were facilitated by a message broker like *Apache Kafka*.
- **Containers and Orchestration**: The team used **Docker** for containerizing services and **Kubernetes** for orchestrating and managing these containers.
- **Database Optimization**: A NoSQL database (*MongoDB*) was introduced for product catalog and session management, while relational databases like *PostgreSQL* were retained for transactional data like orders.

## 3. Incremental Migration Strategy

To ensure minimal disruption to ongoing business operations, the migration from monolithic to microservices was carried out incrementally:

- **Phase 1: Search and Product Catalog**: The first components to be migrated were the product catalog and search functionality. This allowed ShopVerse to experiment with microservices on non-critical features while avoiding risks to checkout and order management.
- **Phase 2: User Management and Checkout**: After the success of the initial phase, the team migrated the user authentication and checkout modules.

- **Phase 3: Inventory and Real-Time Features**: The final phase involved migrating the inventory system and implementing real-time stock updates using Kafka.

During the migration, traffic was routed through the API gateway, allowing certain requests to be handled by the new microservices while legacy components still handled others. This ensured that users experienced minimal downtime and disruptions.

## 4. Performance and Load Testing

Once the critical services were migrated, the team conducted extensive performance and load testing. Tools like **JMeter** and **Gatling** were used to simulate high-traffic scenarios, such as holiday sales, to ensure the platform could handle the increased load. The results showed significant improvements in response times and scalability:

- **Response Times**: Improved by 30%, with average page load times decreasing from 3.2 seconds to 1.8 seconds.
- **Scalability**: Horizontal scaling with Kubernetes allowed the platform to handle 5x the number of concurrent users as before.

## 5. Post-Migration Monitoring and Optimization

After migration, the system was continuously monitored using tools like *Prometheus* and *Grafana* to ensure the new architecture met the business's performance and reliability goals. Logs and metrics were analyzed, and the microservices architecture was fine-tuned accordingly.

**Results**

The architecture reconstruction had a significant positive impact on ShopVerse:

- **Improved Performance**: The microservices-based system was able to handle a much higher volume of traffic, reducing the incidence of system slowdowns during peak times.
- **Scalability**: With the new architecture, ShopVerse could easily scale individual services based on demand. For example, during high sales periods, the search service and checkout service could be scaled independently without affecting other parts of the platform.
- **Reduced Maintenance Costs**: The modular nature of microservices made it easier for developers to maintain and update individual components without worrying about breaking other parts of the system.
- **Real-Time Inventory Management**: By integrating Kafka and real-time processing, ShopVerse was able to implement live inventory updates, preventing overselling and enhancing customer satisfaction.

**Learning Outcome:**

- **Incremental Migration Minimizes Risk**: The incremental approach to transitioning from monolith to microservices ensured that the business experienced minimal disruption during the reconstruction.
- **Comprehensive Monitoring is Critical**: Continuous monitoring of both the legacy and new systems was vital in ensuring the platform's stability throughout the process.
- **Importance of Team Training**: Shifting to a microservices architecture required the development team to be trained on new technologies such as Kubernetes, Docker, and Kafka, which was a significant undertaking but paid off in terms of performance improvements.

## Conclusion

Architecture reconstruction was essential for ShopVerse to modernize its e-commerce platform and continue growing in an increasingly competitive market. By adopting microservices and modern deployment strategies, they overcame the challenges of scalability, performance, and flexibility, positioning themselves to integrate new features and technologies with ease. This case study demonstrates how architectural transformation, when done incrementally and systematically, can provide significant long-term benefits.

**For Faculty Use:**

| **Correction Parameters** | **Formative Assessment [40%]** | **Timely completion of Practical [ 40%]** | **Attendance / Learning Attitude [20%]** | |
|---|---|---|---|---|
| **Marks Obatined** | | | | |