Software Design & Integration

Continuous Integration (CI)

Overview: Continuous Integration (CI) is a practice where developers frequently integrate code into a shared repository. Each integration triggers automated builds and tests, allowing for the quick detection and easier location of errors.

Key Benefits:

- **Quick Error Detection**: Immediate feedback helps address issues early.
- **Faster Development Cycles**: Frequent testing reduces integration risks.
- **Improved Code Quality**: Automated testing ensures stability with each code change.

Experience Sharing: Have you experienced improvements in error detection, faster releases, or better code quality using CI/CD?

Single Code Base

Challenges of Maintaining Multiple Software Versions:

- Multiple software versions can be difficult to manage.
- A core product should include common features needed by all target customers.
- Maintaining a **single code base** ensures that all customers use a consistent code version.
    - Branching strategies can be used for different customers, but all branches should merge into a common main trunk.

[Insert Single Code Base Diagram here]

Product Configuration Capability

Overview:

Products often require customization to cater to different customer needs.

Configuration Options:

- **Choice of Modules** (e.g., SAP)
- **Configurable Workflows** (e.g., Salesforce)
- **Configurable Fields** (e.g., SAP)
- **Configurable Rules** (e.g., Navitair Airline Reservation)
- **Configurable UI** (e.g., Yahoo! Mail)
- **Choice of Language**
- **Configurable Error Messages**

Additional Configurations: Can you think of any other areas for customization?

API for Integration

APIs Overview:

APIs (Application Programming Interfaces) allow external systems to interact with your product.

Examples of Integration APIs:

- **Facebook**
- **SAP**
- **Open API of Banks**
- **Google Maps**
- **GitHub**

Exercise: Identify other API examples you have used or encountered in projects.

Component-Based Design

Overview:

Component-based design divides a system into smaller, independent modules that can be developed and tested individually.

Benefits:

- **Ease of Understanding, Building, and Maintenance**: Simplified code structure.
- **Reuse**: Components can be reused across different projects.
- **Scaling & Fault Detection**: Easier to scale and troubleshoot issues.

Examples include **Web Services** and **Microservices**.

[Insert Component-Based Design Diagram here]

Prove Value, Scale Later

Strategy:

Rather than designing for scale from day one, focus on proving the product's value first. Once validated, optimize and scale the product.

Example: Zendrive

- **Zendrive** captures driving behavior data from drivers' phones for analysis.
- Initially, raw data was collected and sent to servers.
- After proving product value, optimization was introduced by summarizing data on the phone before sending it to servers, reducing data transfer and server load.

Exercise: Can you think of another example where a product was optimized after proving its value?

Re-architecting Products

As a business grows, products may require re-architecture to adapt to new needs and expectations.

Examples:

- **Amazon**: Shifted from monolithic software to a microservices architecture (using "2-pizza teams").
- **Adobe Creative Suite**: Transitioned from desktop-based to cloud-based software.
- **Oracle Apps**: Moved from on-premise to cloud solutions.

[Insert Re-architecting Diagram here]

Exercise: List other companies you know that have re-architected their products.

Platform as a Product

Overview:

A platform acts as a base for building new services or features. It supports diverse plug-in modules and capabilities.

Examples:

- **Apple iOS** & **Android**: Serve as mobile application platforms.
- **AWS & Azure**: Provide cloud services like databases, messaging, and serverless functions.
- **Uber** & **Airbnb**: Adapted to offer virtual experiences during the COVID-19 pandemic.

Benefits:

- Facilitates scalability, modularity, and flexibility.
- Supports a broad range of functionalities via plug-ins.

[Insert Platform as a Product Diagram here]

Exercise: Identify another platform-based product and describe its core features.

Case Study: Visio Graphics-Charting Software

Overview:

Visio is a graphics-charting software that allows users to create diagrams using smart shapes and custom scripting. It has plug-in modules tailored to various industries (e.g., biotechnology, engineering, insurance).

Key Components:

- **Core Graphics Engine**
- **SmartShape Management**: Allows for the incorporation and manipulation of graphic objects.
- **API for Scripting**: Enables developers to create and integrate plug-ins.

[Insert Case Study Diagram here]

Scenario: Discuss how you would use Visio's capabilities to solve a complex diagramming challenge in your organization.

Exercises & Solutions

Exercise 1: API Integration

- **Task**: Identify three APIs you could integrate into a hotel reservation system to enhance functionality.

Solution:

- **Google Maps API**: For location-based search and navigation.
- **Payment Gateway API**: For handling secure transactions.
- **SMS Gateway API**: For sending booking confirmations.

Exercise 2: Configuring a Product

- **Scenario**: You are tasked with customizing a CRM application for a client. Define configurable options needed to meet the client's requirements.

Solution:

- Configurable fields for customer data.
- Choice of modules based on specific business processes.
- Customizable UI to match the client's branding.