Re-Architecting

Software Re-engineering

Usman Ghani

Lecturer

Computer Science

Agenda

- Re architecting
- Breaking an application into modules
- Distributing web application into services
- Choosing an architecture
- Technical and Organizational Benefits
- Challenges
- Summary

Re architecting

- Re-architecting in the context of software reengineering involves making significant changes to the existing architecture of a software system.
- This is typically done to address issues such as performance bottlenecks, scalability concerns, or to adapt the system to new requirements and technologies.

Re architecting

- Re-architecting may include restructuring the overall design, modifying the interaction between components, and sometimes even changing the underlying technologies.
- It is a strategic effort aimed at improving the long-term viability and maintainability of the software.

Breaking an Application into Modules

- Breaking an application into modules is a modularization strategy that involves dividing a monolithic application into smaller, independent units or modules.
- Each module encapsulates a specific set of functionalities and has well-defined interfaces for interaction with other modules.

Breaking an Application into Modules

- This modular approach enhances maintainability, scalability, and reusability of code.
- It allows developers to work on and update individual modules without affecting the entire application.
- Breaking an application into modules is often a key step in reengineering to improve code organization, collaboration, and ease of maintenance.

Distributing Web Application into Services

- Distributing a web application into services refers to the architectural pattern known as microservices.
- In this approach, a monolithic web application is decomposed into a collection of small, independent services that communicate with each other through well-defined APIs.

Distributing Web Application into Services

- Each service is responsible for a specific business capability and can be developed, deployed, and scaled independently.
- This distribution of services provides benefits such as improved scalability, fault isolation, and the ability to use different technologies for different services.
- However, it also introduces challenges related to service coordination, data consistency, and network communication.

Choosing an Architecture

- Selecting an appropriate architecture is a crucial decision in software reengineering.
- The architecture of a software system defines its high-level structure, components, and the way they interact.
- The choice of architecture depends on various factors, including the system's requirements, scalability needs, performance considerations, and the development team's expertise.

Choosing an Architecture

- Common architectural styles include monolithic, microservices, client-server, and event-driven architectures.
- The selected architecture should align with the goals of the reengineering effort and support the long-term evolution of the software.

Technical and Organizational Benefits

- Software reengineering can bring about both technical and organizational benefits. Technical benefits may include improved performance, scalability, maintainability, and adaptability to new technologies. The reengineering process may lead to a more modular and flexible codebase, making it easier to enhance and extend the software.

Technical and Organizational Benefits

- Organizational benefits may include increased collaboration among development teams, better alignment with business goals, and a more efficient development process.
- Reengineering can also result in cost savings over time by reducing technical debt and minimizing the need for extensive rework in the future.

Challenges

- Despite the potential benefits, software reengineering comes with its set of challenges. Some common challenges are here listed.

1. Risk of Disruption

- Introducing major changes to a software system can be disruptive, potentially impacting existing functionality and user experience.

2. Cost and Resource Allocation

- Reengineering efforts require resources in terms of time, money, and skilled personnel.
- Balancing these resources while ensuring the continuity of regular operations can be challenging.

3. Legacy Integration

- Systems often need to interact with existing legacy components or data, posing challenges in integrating new architecture with the old.

4. Testing and Validation

- Validating the correctness and reliability of the reengineered system is crucial.
- Comprehensive testing strategies are needed to ensure that the changes do not introduce new defects.

5. Change Management

- Managing resistance to change within the organization is important.
- Stakeholders, including developers and end-users, may be accustomed to the existing system and resistant to modifications.

Summary

- These approaches in software reengineering aim to enhance the architecture, organization, and scalability of a software system.
- Re-architecting focuses on redesigning the overall structure, breaking an application into modules emphasizes modularization for maintainability, and distributing a web application into services adopts a microservices architecture for improved scalability and flexibility.
- Each approach addresses specific challenges and contributes to making the software system more adaptable and resilient.

Summary

- Successfully navigating these challenges requires careful planning, stakeholder communication, and a well-defined strategy for mitigating risks.
- Software reengineering should be approached methodically to maximize benefits while minimizing potential disruptions and drawbacks.