1 Lecture 4

1.1 Exercise 1: Pitching your Software Architecture

Objective:

The goal of this exercise is to pitch the designed software architecture. This will aid in comprehending how to convey complex technical details to a varied audience, focusing on the needs, approach, benefits, and possible alternatives.

Tasks:

- *Identify Needs:* Establish the business or technical necessities your architecture addresses. Is it solving a prevalent problem or improving an existing solution? Clearly articulate this need in layman's terms.
- *Outline Approach:* Briefly describe the software architecture. Avoid unnecessary jargon and ensure clarity for a non-technical audience.
- *Highlight Benefits:* Explain the advantages and the value proposition of your architecture. How does it stand out in terms of performance, scalability, maintainability, and other quality attributes?
- *Acknowledge Competition/Alternatives:* Recognize existing solutions and justify why your architecture is preferable. What does it offer that alternatives do not, and why is it the best fit for the identified need?
- *Develop a Clear and Concise Pitch:* Synthesize the above elements into a coherent and persuasive 3-minute pitch, ensuring it is understandable to a broad audience. Utilize visual aids, if possible, to enhance comprehension.

Deliverables:

- You will present your pitch in the second half of the lecture.

Assessment Criteria:

The success of the pitch will be evaluated based on the clarity and persuasiveness of your arguments, the relevance of the identified needs, the plausibility of the proposed architecture meeting the needs, and the acknowledgment of alternatives.

1.2 Exercise 2: Integrating Message Busses in Software Architecture

Objective:

This exercise aims to integrate message busses into your software architecture, promoting enhanced inter-process communication and system scalability. A focus will be placed on selecting appropriate message bus technologies and designing the communication protocols to optimize data flow and system integration.

Tasks:

- *Understand Message Busses:* Study the fundamentals of message busses and their role in facilitating communication between different systems and subsystems. Investigate different technologies like MQTT, Apache Kafka, and others, to understand their features, advantages, and limitations.
- *Identify Integration Points:* For each system and subsystem identified in previous exercises:
  - Recognize where the integration of message busses can improve communication and data flow.
  - Design the communication flows defining how systems and subsystems will interact through the message bus.
- *Select Appropriate Message Bus:* Based on your research, choose the most suitable message bus technology for your architecture, considering factors like scalability, reliability, and performance.
- *Justify Your Choices:* Explain why the chosen message bus technology and designed communication protocols are the best fit for your architecture's needs and constraints. Support your justifications with real-world examples, academic papers, or logical reasoning.
- *Assess Impact on Architecture:* Reflect on how the integration of message busses will affect the quality attributes of your system, such as modifiability, scalability, and performance.

Example: E-Commerce System

- **Message Bus:** Apache Kafka
- **Integration Points:** Between Inventory Management Subsystem and Order Processing Subsystem for real-time inventory updates; Between User Interface Subsystem and Recommendation Engine Subsystem for personalized user recommendations.

Justification:

- **Apache Kafka:** is chosen for its high throughput and scalability, crucial for managing the high volume of messages generated by the active user interactions and system updates in an E-Commerce platform.
- *Quality Attributes:* The choice of Apache Kafka promotes system scalability and reliability, ensuring seamless user experience even during peak usage times.

Deliverables:

- A document outlining the integration points, selected message bus technology, designed communication protocols, and their justifications, along with reflections on the impact on the architecture's quality attributes.
- Diagrams illustrating the integration of the message bus within the architecture and the flow of messages between different systems and subsystems.

Assessment Criteria:

Assessment will be based on the appropriateness of the selected message bus technology, the coherence of the designed communication protocols, the depth, and clarity of the justifications provided, and the understanding of the impact of integrating message busses on the architecture's quality attributes.

1.3 Exercise 3: Integrating Containerization using Docker

Objective:

This exercise focuses on integrating containerization, specifically using Docker, into your software architecture. By containerizing different components of your system, you aim to enhance the portability, scalability, and maintainability of your software, allowing for more seamless deployment, scaling, and development processes.

Tasks:

- *Study Docker and Containerization:* Start with understanding the basics of containerization and Docker. Investigate how containerization can aid in isolating and deploying applications in varied environments and understand the advantages it brings in terms of scalability and manageability.
- *Identify Components for Containerization:* From the systems and subsystems identified in previous exercises, determine which components would benefit most from containerization.
  - Consider components that require isolation, have specific environment requirements, or need to be scalable.
- *Design Dockerfiles and Compose Files:* For the identified components:
  - Create Dockerfiles to define the environment, dependencies, and configurations needed.
  - If multiple containers need to work together, design Docker Compose files to manage the interactions between them.
- *Justify Your Choices:* Clearly explain the reasoning behind containerizing the selected components and the configurations chosen in the Dockerfiles and Compose files. Relate your decisions back to how they improve the quality attributes of your system.
- *Assess Impact on Architecture:* Reflect on the integration of Docker and its impact on the software architecture's quality attributes, such as deployability, modifiability, and scalability.

Example: Financial Analysis System

- **Components for Containerization:** Data Processing Subsystem, User Interface Subsystem.

Justification:

- **Data Processing Subsystem:** Requires a specific set of dependencies and environment configurations to process financial data, making it an ideal candidate for containerization to ensure consistent behavior across different environments.

- **User Interface Subsystem:** Containerizing allows for easier scaling to handle varying loads and isolates the user interface to prevent conflicts with other subsystems.
- *Quality Attributes:* The use of Docker enhances the deployability and scalability of the system while ensuring consistent behavior across different deployment environments.

Deliverables:

- A document detailing the components selected for containerization, the designed Dockerfiles and Docker Compose files, and their justifications, along with insights on the impact on the architecture's quality attributes.

Assessment Criteria:

The evaluation will focus on the appropriateness of the chosen components for containerization, the effectiveness of the designed Dockerfiles and Compose files in addressing the components' needs, the depth and clarity of the justifications provided, and the understanding of the impact of integrating Docker on the architecture's quality attributes.