Advanced Topics in Software Architecture Exercises

Torben Worm

September 2023

1 Lecture 3

1.1 Exercise 1: Programming Languages

Objective:

Building upon your work from Exercise 3 in Lecture 2, the primary aim of this exercise is to select appropriate programming languages for the different systems and subsystems in your project case. This will help you understand the suitability of different programming languages for various tasks within a large-scale, complex system.

Tasks:

- *Research Programming Languages:* Familiarize yourself with a variety of programming languages commonly used in industrial and software engineering. Consider languages like Python, C/C++, Java, C#, Rust, Go, and others that might be relevant to Industry 4.0.
- *Match Languages to Systems/Subsystems:* For each system and subsystem identified in Exercise 3:
  - Argue for the use of one or more programming languages that would be well-suited for the development and maintenance of that system or subsystem.
  - Discuss the trade-offs (e.g., performance, scalability, maintainability) involved in your choices.
- *Justify Your Choices:* Explain why the selected programming languages are the best fit for the specific needs and constraints of your systems and subsystems. You may refer to real-world examples, academic papers, or your own reasoning. Also, reflect on how your choices impact the Quality Attributes of your system.

Example: Quality Assurance System

- **Automated Inspection Subsystem:** C++
- **Defect Reporting Subsystem:** JavaScript

Justification:

- **C++:** is used for the Automated Inspection Subsystem because of its performance benefits and real-time capabilities, essential for image processing and real-time defect detection.
  Quality Attributes: The choice of C++ enhances the system's performance and real-time capabilities. Its low-level features also offer more control over hardware, which is critical for timely and accurate defect inspection.

- **JavaScript:** is chosen for the Defect Reporting Subsystem to allow for a web-based, real-time dashboard where defects are reported and can be viewed immediately from anywhere in the production line.
  Quality Attributes: Using JavaScript promotes accessibility and user experience, as it enables real-time updates and a platform-independent dashboard. However, it may require careful management to ensure scalability and security.

Deliverables:

- A document that outlines the chosen programming languages for each identified primary system and their constituent subsystems, including justifications for each choice.
- Optional: Code snippets or pseudo-code to illustrate how specific tasks could be implemented in the chosen languages.

Assessment Criteria:

The clarity, depth, and persuasiveness of your arguments for selecting specific programming languages. Consideration should also be given to the appropriateness of the language for the system and subsystem requirements.

1.2 Exercise 2: Selecting Databases for Systems and Subsystems

Objective:

Building upon your previous work in Exercise 3, Lecture 2, the primary objective of this exercise is to select appropriate databases for the different systems and subsystems in the Industry 4.0 production domain. This will enable you to comprehend the critical role databases play in ensuring data integrity, performance, and scalability in large-scale systems.

Tasks:

- *Research Database Options:* Familiarize yourself with a variety of databases commonly used in the industry. Consider databases like SQL databases (MySQL, PostgreSQL), NoSQL databases (MongoDB, Cassandra), and in-memory databases (Redis).
- *Match Databases to Systems/Subsystems:* For each system and subsystem identified in Exercise 3:
  - Select one or more databases that would be well-suited for the data storage and retrieval needs of that system or subsystem.
  - Discuss the trade-offs (e.g., performance, scalability, complexity) involved in your choices.

- *Justify Your Choices:* Explain why the selected databases are a good fit for the specific needs and constraints of your systems and subsystems. Refer to real-world examples, academic papers, or your own reasoning. Again, reflect on your choices' effect on Quality Attributes.

Example Database Selection:

Primary System: Supply Chain Management System
Chosen Database(s):

- Inventory Tracking Subsystem: PostgreSQL
- Order Processing Subsystem: MongoDB

Justification:

- **PostgreSQL:** is chosen for the Inventory Tracking Subsystem primarily for its ACID (Atomicity, Consistency, Isolation, Durability) compliance.
  Quality Attributes: ACID compliance ensures data integrity and reliability, which are critical in an inventory tracking system. A failure in tracking could lead to inaccurate stock counts, affecting the entire supply chain.
- **MongoDB:** is selected for the Order Processing Subsystem for its flexibility in schema design and scalability.
  Quality Attributes: MongoDB's flexible nature allows for quick adaptation to changing business requirements in order processing. Its ability to scale horizontally ensures that the system can handle growing data volumes and transaction rates.

Deliverables:

- A document that outlines the chosen databases for each identified primary system and their constituent subsystems, including justifications for each choice.
- Optional: Data models or ER diagrams that illustrate how data would be structured in the chosen databases.

Assessment Criteria:

The clarity, depth, and persuasiveness of your arguments for selecting specific databases. Consideration should also be given to the appropriateness of the database for the system and subsystem requirements.

1.3 Exercise 3: Peer Review of Systems and Subsystems Design

Objective:

The aim of this exercise is to engage in a peer review process where each student group will review the work of another group. This will facilitate learning through exposure to different perspectives and provide constructive feedback for improvements.

Tasks:

- *Download Review Material:* Navigate to Resources –> Review Material and download the material you have been assigned to review.
- *Perform the Review:* Using the provided rubric (Table 1), evaluate the work of the other group.
- *Document Your Review:* Prepare a written peer review that clearly outlines your evaluation based on the rubric. Make sure to provide constructive feedback.

Deliverables:

- A written peer review document, following the specified rubric.

Assessment Criteria:

- Thoroughness and relevance of the peer review.
- Quality of the constructive feedback provided.

Review Timeline:

The peer review can be performed either during the exercise session or as homework, depending on the course schedule.

Review Material:

The artifacts to review include the text and diagrams produced during the exercises in Lecture 2.

Criteria	Poor	Sufficient	Good	Excellent
Description of the domain	Incomplete or inaccurate	Somewhat accurate but lacks depth	Accurate and reasonably thorough	Comprehensive and highly representative
Quality of use cases	Vague and uninformative	Somewhat informative but lacks detail	Clear, but could be more detailed	Clear, thorough, and very well-defined
System structure supports use cases	Does not support use cases	Partially supports use cases	Supports most use cases	Supports all use cases excellently
Quality of arguments	Unconvincing or lacking	Somewhat convincing, but not strong	Convincing for the most part	Highly convincing and well-supported

Table 1: Peer Review Rubric for Systems and Subsystems Design