Coding Book

Adarsh Choudhary, Mohit Jain, Surabhi Lone

October 2024

1 Preface: Definitions & Terminology

1.1 Basic Definitions

The foundational terms used to categorize Existence, Executive, and Property issues in this book draw from Kruchten's work:

> Kruchten, Philippe. *An Ontology of Architectural Design Decisions in Software-Intensive Systems.* Presented at the 2nd Groningen Workshop on Software Variability, 2004.

These definitions are essential for labeling issues correctly. The guidelines provided here expand on Kruchten's definitions, clarifying our interpretation and approach to classifying various types of issues.

1.2 Determining Architectural Significance

The distinction between architectural and non-architectural issues often hinges on the complexity or difficulty involved in making a change. For instance, introducing a new component becomes an architectural decision if it would significantly impact the system and require substantial future modifications. A guiding question we ask is, *"Would changing this later be challenging?"*

2 Guidelines for Specific Types of Issues

- **User Requests**: Analyze requests based on the type of decisions they encompass.

3 Architectural Issues

Architectural significance indicates whether an issue influences the system's core qualities, such as its structure, performance, and maintainability. Architectural issues typically:

- Affect system-wide quality attributes, such as performance, scalability, and reliability.
- Entail substantial changes or improvements to primary components or modules.
- Influence how components interact or are structured within the system.

3.1 Issues Considered Architectural, but Not Existence

The following issues were considered architectural but do not fall under existence decisions:

- AXIS2-5993 : Upgrading to log4j v2.x is a big change that affects logging.

4 Classifying by Decision Type

4.1 Existence Decisions

Existence decisions include introducing or removing core components, as well as any substantial structural adjustments. These decisions shape the architectural framework by adding or changing foundational elements.

Key questions:

- Does this involve adding, removing, or replacing a fundamental component?
- Does it alter the system's underlying structure?

Examples of Issues:

- AXIS2-3269: Introduce a ServiceInstanceFactory mechanism in the JAX-WS layer.
- AXIS2-3000: Add a phase to incorporate all addressing handlers.
- AXIS2-2456: Enable direct streaming of attachments.
- AXIS2-2449: Stream content for requests and responses in SimpleHttpServer.
- AXIS2-2996: Develop a CORBA module for Axis2.
- AXIS2-5322: Adds a new piece called ServiceBuilderExtension that changes how services are put together.
- AXIS2-5659: Adds new settings for address and identity handling, a big change for security.
- AXIS2-5362: Allows the system to handle JSON data in addition to other data types.

4.2 Property Decisions

Property decisions aim to enhance existing components to improve quality attributes, such as performance, reliability, maintainability, or usability. These adjustments refine the behavior of current elements without altering the core structure.

Key questions:

- Does this change improve attributes like performance, reliability, or maintainability?
- Does it modify the behavior of an existing component without structural alteration?

Examples of Issues:

- AXIS2-3137: Address unclosed XMLStreamReader issues in XMLUtils.
- AXIS2-3298: Correct JAX-WS Async Endpoint thread-switching code.
- AXIS2-3202: Resolve client halt issues caused by server-side connection closure.
- AXIS2-4906: Makes the system faster by cutting out slow steps.
- AXIS2-4509: Improves general speed and performance.

4.3 Executive Decisions

Executive decisions involve choosing or updating major technologies, tools, or overarching policies that guide the development environment. These decisions shape the strategic direction of development without directly modifying components or their interactions.

Key questions:

- Does this involve selecting or substituting a significant technology or tool?
- Will this impact the development or deployment process?

Examples of Issues:

- AXIS2-5993: Upgrades to a newer version of the log4j library, improving security and performance.
- AXIS2-5230: Adds a tool called Maven archetype to make project setup easier.
- AXIS2-5229: Another Maven archetype tool that simplifies creating new services.

4.4 Non-Architectural Decisions

Non-architectural decisions do not impact the overall architecture of the system, but they contribute to operational efficiency or support.

Examples of Issues:

- AXIS2-5231: Fixes a bug in service deployment, making it work correctly without changing the architecture.
- AXIS2-5230: Adds a Maven archetype tool to help create new services, useful for setup but not a core change.
- AXIS2-5229: Another Maven archetype addition, making project setup easier but not changing the system itself.