Foobar Inc

C++ Arithmetic Evaluator
Software Architecture Document

Version 0.1

Revision History

Date	Version	Description	Author
010/Nov/2023	0.1	Initial Project Architecture	Tuan Vu

Table of Contents

Software Architecture Document

1. Introduction

1.1 Purpose

The Software Architecture Document (**SAD**) is a guide for **the project**. It breaks down **the project**'s structure, design principles, and important considerations.

1.1.1 Structure:

It covers different aspects of the system, from the overview to the smaller details.

1.1.2 Audience and Usage:

Developers: Use it to understand how to implement specific features and grasp the overall system design.

Architects: It's your tool to communicate design decisions, envision system interactions, and ensure the project stays on track.
Stakeholders: Help understand why certain decisions were made, how they impact the project, and what to expect as things progress.

1.2 Scope

This **SAD** describes all systems and subsystems of **the project**. It defines the boundaries of what's covered, touching everything from the fundamental building blocks to the overarching design principles.

1.3 Definitions, Acronyms, and Abbreviations

The project/this project/the program: "C++ Arithmetic Evaluator", described by the Software Development Plan (**SDP**). The repository for all code and documentation lives on GitHub, at https://github.com/codyduong/EECS-328-Project/.
SDP: Software Development Plan. See *Section 1.4 References — SDP*
SAD: Software Architecture Document. See *Section 1.4 References — SAD*
SRS: Software Requirements Specification. See *Section 1.4 References — SRS*

1.4 References

SDP: Available at https://github.com/codyduong/EECS-328-Project/
SAD: Available at https://github.com/codyduong/EECS-328-Project/
SRS: Available at https://github.com/codyduong/EECS-328-Project/

1.5 Overview

The document is organized into sections, starting with a brief introduction and reference information. The subsequent sections cover Architectural Representation, Architectural Goals and Constraints, Use-Case View, Use-Case Realizations, Logical View, Interface Description, Size and Performance, and Quality. Each section contributes to a holistic understanding of the project's architecture and design decisions, catering to different audience's needs.

2. Architectural Representation

The software architecture of the C++ Arithmetic Evaluator is represented through multiple architectural views, each focusing on specific aspects of the system. These include:

- Logical View: Describing the decomposition into subsystems and packages with a focus on significant classes, relationships, operations, and attributes.

- Process View: Illustrating the dynamic aspects of the system, including processes, tasks, and their interactions.
- Physical View: Defining the deployment and distribution of components across hardware resources.

3. Architectural Goals and Constraints

The architectural goals for the C++ Arithmetic Evaluator include:

- Safety: Ensuring that the evaluator handles inputs and expressions safely, preventing runtime errors or crashes.
- Security: Implementing secure coding practices to protect against potential vulnerabilities and attacks.
- Portability: Designing the system to be easily portable across different platforms and environments.
- Development Tools: Utilizing [specific tools and environments] for code development and testing.
- Team Structure: Coordinating development efforts among [team roles] to ensure collaboration and efficiency.

4. Logical View

4.1 Overview

This subsection describes the overall decomposition of the design model in terms of its package hierarchy and layers.

4.2 Architecturally Significant Design Modules or Packages

4.2.1 Presentation Layer

User Interface Package:

- Description: Manages user input and output.
- Classes: n/a

4.2.2 Business Logic Layer:

Expression Handling Package:

- Description: Deals with parsing and evaluating arithmetic expressions.
- Classes: Number node, Binary operator node, Expression parser.

5. Interface Description

This section provides a description of major entity interfaces, including screen formats, valid inputs, and resulting outputs. The major entity interfaces for the C++ Arithmetic Evaluator include the User Interface, specifying console-based interactions.

6. Quality

This section describes how the software architecture contributes to all capabilities of the system, beyond functionality. It emphasizes extensibility, reliability, security, and other quality attributes, ensuring a robust and effective C++ Arithmetic Evaluator.