Boolean Logic Simulator in C++

Version 1.0

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 04/08/2024 | 1.0 | Initial Template Filled Out | Kemar Wilson |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

1. Introduction 4

1.1 Purpose 4

1.2 Scope 4

1.3 Definitions, Acronyms, and Abbreviations 4

1.4 References 4

1.5 Overview 4

2. Architectural Representation 4

3. Architectural Goals and Constraints 4

4. Use-Case View 4

4.1 Use-Case Realizations 5

5. Logical View 5

5.1 Overview 5

5.2 Architecturally Significant Design Packages 5

6. Interface Description 5

7. Size and Performance 5

8. Quality 5

# Introduction

The Boolean Logic Calculator project aims to develop a C++ software application that enables users to perform logical operations based on Boolean algebra. This Software Architecture Document provides a comprehensive overview of the architectural design and structure of the Boolean Logic Calculator software. The document outlines the purpose, scope, definitions, acronyms, and abbreviations used throughout the software development process. By detailing the architecture and design principles, this document serves as a reference guide for developers, stakeholders, and users involved in the creation and utilization of the Boolean Logic Calculator software.

## Purpose

[This document provides a comprehensive architectural overview of the system, using a number of different architectural views to depict different aspects of the system. It is intended to capture and convey the significant architectural decisions which have been made on the system.]

[This section defines the role or purpose of the **Software Architecture Document**, in the overall project documentation, and briefly describes the structure of the document. The specific audiences for the document are identified, with an indication of how they are expected to use the document.]

## Scope

[A brief description of what the Software Architecture Document applies to; what is affected or influenced by this document.]

## Definitions, Acronyms, and Abbreviations

[This subsection provides the definitions of all terms, acronyms, and abbreviations required to properly interpret the **Software Architecture Document**.  This information may be provided by reference to the project’s Glossary.]

## References

[This subsection provides a complete list of all documents referenced elsewhere in the **Software Architecture Document**. Identify each document by title, report number (if applicable), date, and publishing organization. Specify the sources from which the references can be obtained. This information may be provided by reference to an appendix or to another document.]

## Overview

[This subsection describes what the rest of the **Software Architecture Document** contains and explains how the **Software Architecture Document** is organized.]

# Architectural Representation

[This section describes what software architecture is for the current system, and how it is represented. It enumerates the views that are necessary, and for each view, explains what types of model elements it contains.]

# Architectural Goals and Constraints

[This section describes the software requirements and objectives that have some significant impact on the architecture; for example, safety, security, privacy, use of an off-the-shelf product, portability, distribution, and reuse. It also captures the special constraints that may apply: design and implementation strategy, development tools, team structure, schedule, legacy code, and so on.]

# Use-Case View

[This section lists use cases or scenarios from the use-case model if they represent some significant, central functionality of the final system, or if they have a large architectural coverage—they exercise many architectural elements or if they stress or illustrate a specific, delicate point of the architecture.]

## Use-Case Realizations

[This section illustrates how the software actually works by giving a few selected use-case (or scenario) realizations, and explains how the various design model elements contribute to their functionality. If a Use-Case Realization Document is available, refer to it in this section.]

# Logical View

[This section describes the architecturally significant parts of the design model, such as its decomposition into subsystems and packages. And for each significant package, its decomposition into classes and class utilities. You should introduce architecturally significant classes and describe their responsibilities, as well as a few very important relationships, operations, and attributes.]

## Overview

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

## Architecturally Significant Design Modules or Packages

[For each significant package, include a subsection with its name, its brief description, and a diagram with all significant classes and packages contained within the package.

For each significant class in the package, include its name, brief description, and, optionally, a description of some of its major responsibilities, operations, and attributes.]

# Interface Description

[A description of the major entity interfaces, including screen formats, valid inputs, and resulting outputs. If a User-Interface Prototype Document is available, refer to it in this section]

# Size and Performance

[A description of the major dimensioning characteristics of the software that impact the architecture, as well as the target performance constraints.]

# Quality

[A description of how the software architecture contributes to all capabilities (other than functionality) of the system: extensibility, reliability, portability, and so on. If these characteristics have special significance, such as safety, security or privacy implications, they must be clearly delineated.]