
SuperCoding

**Boogle Project
Software Architecture Document**

Version 1.2

Boogle Project	Version: 1.2
Software Architecture Document	Date: 08/04/2024
SC-SA-Boogle	

Revision History

Date	Version	Description	Author
08/04/2024	1.0	First Revision and Completion	SuperCoding
09/04/2024	1.1	Second Revision	Brandon Dodge
12/04/2024	1.2	Final Revision	Brandon Dodge

Boogle Project	Version: 1.2
Software Architecture Document	Date: 08/04/2024
SC-SA-Boogle	

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.	Logical View	5
5.1	Overview	5
5.2	Architecturally Significant Design Packages	5
5..	Interface Description	5
6.	Quality	5

Boogle Project	Version: 1.2
Software Architecture Document	Date: 08/04/2024
SC-SA-Boogle	

Software Architecture Document

1. Introduction

1.1 Purpose

The Boogle Project is dedicated to creating a versatile platform accessible to users from all disciplines, enabling them to perform swift and efficient Boolean-based calculations and simulations. Through the development of an intuitive interface, the project strives to streamline the process, making it easy for individuals regardless of their expertise level to engage with Boolean logic effectively.

1.2 Scope

- Overall Objective: Develop a C++ program acting as a simplified Boolean logic simulator.
- Functionality:
 - Implement logical operations for AND, OR, NOT, NAND, and XOR.
 - Develop a mechanism to parse user-provided Boolean expressions in, respecting operator precedence and parentheses.
 - Allow users to define truth values for each variable.
 - Calculate the final truth value of the entire expression and present it clearly.
 - Implement robust error handling for invalid expressions, missing parentheses, unknown operators, or other potential issues.
 - Ensure proper handling of expressions enclosed within parentheses to determine the order of evaluation.
- User Interface:
 - Define user interface requirements for accepting Boolean expressions, truth values, and presenting the final truth value output.
- External Dependencies:
 - Identify any external libraries or frameworks required for implementing specific functionalities.
- Performance Considerations:
 - Consider performance requirements for handling complex logic circuits efficiently.
- Testing and Validation Strategy:
 - Outline the approach for testing and validating the software, including unit testing, integration testing, and user acceptance testing.
- Documentation:
 - Specify documentation requirements, developer guides, and code documents.

1.3 Definitions, Acronyms, and Abbreviations

- Boolean - The process of using only 0s and 1s
- Parsing - Analyzing string symbols in order to determine output data
- Module - a self-contained unit of software that encapsulates related functions, data, or procedures, promoting code organization, reusability, and maintainability.
- AND Operation:
 - Takes two input values.
 - Returns true if and only if both input values are true; otherwise, it returns false.
- OR Operation:
 - Takes two input values.
 - Returns true if at least one of the input values is true; otherwise, it returns false.
- NOT Operation:
 - Takes a single input value.
 - Returns the opposite of the input value; true becomes false, and false becomes true.
- NAND Operation:
 - Takes two input values.
 - Returns false if both input values are true; otherwise, it returns true.
- XOR Operation:

Boogle Project	Version: 1.2
Software Architecture Document	Date: 08/04/2024
SC-SA-Boogle	

- Takes two input values.
- Returns true if exactly one of the input values is true; otherwise, it returns false

1.4 References

EECS 348 Part 3: Software Architecture Announcement

EECS 348 Software Engineering I Project files

(<https://canvas.ku.edu/courses/119254/files/folder/Project?>)


PyQt Boolean GUI

(<https://www.riverbankcomputing.com/static/Docs/PyQt5/index.html>)

Stanford University Boolean Documentation

(<http://thue.stanford.edu/bool.html>)

02 Software Requirement Specifications Document

 02-Software-Requirements-Spec.docx

1.5 Overview

The Software Architecture document is organized into 8 sections, with possible subsections included. The sections include:

- Introduction
 - Purpose
 - Scope
 - Definitions
 - References
 - Overview
- Architectural Representation
- Architectural Goals and Constraints
- Logical View
 - Overview
 - Architecturally Significant Design Modules or Packages
- Interface Description
- Quality

2. Architectural Representation

For the Boogle Project, the architectural representation encompasses several key views to effectively capture the system's design and functionality:

- Logical View: This view focuses on the high-level organization of the software components and their interactions. It includes model elements such as classes, interfaces, modules, and their relationships, illustrating how different parts of the system collaborate to execute Boolean-based calculations and simulations.
- User Interface View: This view details the presentation layer of the system, illustrating how users interact with the interface to input Boolean expressions, configure simulations, and interpret results. Model elements include user interface components, screens, menus, and navigation flows, highlighting the ease of use and accessibility of the system.
- Data View: This view describes the data structures and storage mechanisms utilized by the system. It includes model elements such as databases, files, data schemas, and data flows, illustrating how Boolean expressions and simulation configurations are stored, retrieved, and manipulated within the system.

Boogle Project	Version: 1.2
Software Architecture Document	Date: 08/04/2024
SC-SA-Boogle	

- **Behavioral View:** This view outlines the dynamic behavior of the system, showcasing how it responds to user inputs, processes calculations, and executes simulations. Model elements include state diagrams, sequence diagrams, and activity diagrams, providing insight into the system's operational flow and logic execution.

By incorporating these views, the architectural representation of the Boogle Project comprehensively captures its design and functionality, facilitating understanding, communication, and further development of the system.

3. Architectural Goals and Constraints

The architecture will ensure the calculator to operate reliability and security, minimizing the risk of errors in calculations. It implements measures to protect user data and prevent unauthorized access or manipulation of calculations and history records. The system respects user privacy by safeguarding sensitive information, such as the history of calculation and providing control over data management. The system is designed to be platform-independent, allowing it to run in the different operating systems and environments. Boogle is designed with C++ in Linux for development. Some parts of the code will form existing code or components from related projects while minimizing the technical debt. The development of Boogle is delivered with incremental updates and improvements to meet the architecture goals and deadlines. The test strategy includes unit tests, reliability, verifying the correctness, case insensitivity test and performance of the calculation.

4. Logical View

4.1 Overview

The logical overview of the Boogle Project's architecture encompasses ten essential modules, each fulfilling distinct functions to ensure the system's robustness and usability. The User Interface Module facilitates seamless interaction between users and the system through graphical user interface (GUI) components, while the Boolean Calculation Module handles fundamental and complex Boolean operations. The Simulation Module enables the creation, execution, and visualization of simulations based on Boolean logic, while the Parser Module converts user input into a format processable by the system. The Data Management Module ensures efficient storage and retrieval of Boolean expressions and configurations, and the Error Handling Module detects and reports errors for enhanced system stability. Additionally, the Documentation Module generates comprehensive project documentation, and the Testing Module ensures software reliability through rigorous testing methodologies. The Logging Module facilitates system behavior tracking and issue diagnosis, while the Configuration Module allows users to customize software behavior according to their preferences, collectively providing a comprehensive framework for the Boogle Project's functionality.

4.2 Architecturally Significant Design Modules or Packages

- **User Interface Module:** This module handles the graphical user interface (GUI) components, allowing users to interact with the system easily.
- **Boolean Calculation Module:** This module contains functions and algorithms for performing Boolean-based calculations. It might include operations like AND, OR, NOT, XOR, as well as more complex operations for manipulating Boolean expressions.
- **Simulation Module:** This module facilitates the creation and execution of simulations based on Boolean logic. It could include features for setting up simulation parameters, running simulations, and visualizing the results.
- **Parser Module:** This module parses user input, converting Boolean expressions into a format that can be processed by the system.
- **Data Management Module:** This module handles the storage and retrieval of Boolean expressions, simulation configurations, and other data within the system.

Boogle Project	Version: 1.2
Software Architecture Document	Date: 08/04/2024
SC-SA-Boogle	

- **Error Handling Module:** This module deals with error detection and reporting within the software. It includes functions for validating user input, handling unexpected errors, and providing meaningful error messages to users.
- **Documentation Module:** This module generates documentation for the project, including user guides, API references, and developer documentation.
- **Testing Module:** This module includes functions and utilities for testing the software to ensure its reliability and correctness.
- **Logging Module:** This module manages logging and debugging information generated by the software during runtime. It helps developers track the system's behavior and diagnose issues efficiently.
- **Configuration Module:** This module handles system configuration settings, allowing users to customize the behavior of the software. It might include features for saving and loading user preferences and settings.

5. Interface Description

Boogle will be an application running in a Bash terminal via the C++ standard I/O library.

All input will be treated as case-insensitive, and whitespace-insignificant, and the application will prompt the user in three distinct cases:

1. The boolean expression to evaluate. For example:

- A&B
- A AND B
- (!TRUE) | SOME

Symbolic and word-form operators will be allowed:

- |, OR
- &, AND
- \$, XOR
- @, NAND
- !, NOT

Boolean literals TRUE and FALSE are also supported within expressions.

Variable names can be any string of letters (other than the above reserved keywords for operators and literals).

Parentheses are supported, as are standard operator precedences: NOT, AND, NAND, XOR, OR.

All operators are binary (e.g. A OR B) except for the NOT operator, which is a unary prefix (NOT A or !A).

If there is a syntax error in the user-provided expression (e.g. mismatched parentheses, not enough or too many arguments for an operator, use of a disallowed character, etc.) the user will be prompted again for an expression.

2. Values for variables. For each variable defined in the given expression, the user will be prompted to give a value (t/f). Any input strings beginning with "t" or "T" will be evaluated as true, and any input strings beginning with "f" or "F" will be evaluated as false.
3. The option for full truth-table output. The user will be prompted with a yes/no option to print a full truth table; if they select 'no', only the relevant row will be output.

The output of Boogle will be a table, constructed using the Unicode box-drawing characters and true/false values. The relevant row will be bolded using terminal control characters.

After the first evaluation, the user will be prompted either to provide another expression to evaluate, view their history, or quit the program.

Boogle Project	Version: 1.2
Software Architecture Document	Date: 08/04/2024
SC-SA-Boogle	

From the history interface, the user may choose to re-evaluate a previous expression.

A mock-up of the future interface can be found [here](#).

6. Quality

Boogle is designed with modularity in mind, employing a layered approach where components are loosely coupled and have a well-defined interface. The architecture will allow developers to extend the system capability by adding the new modules or functionalities without changing the existing mode too much. The architecture is designed to be platform-independent, avoiding platform specific dependencies. With the creation of the makefile, it ensures the system can be developed through different environments with the minimal effort. The architecture prioritizes user safety by implementing features that promote transparency and control over their data. Users can access a comprehensive history of the boolean calculations, empowering them to review past activities for accuracy and auditability.