Kung-Fu Programmers

Boolean Expression Evaluator
Software Requirements Specifications

Version 1.0

Revision History

Date	Version	Description	Author
24/Mar/24	1.0	Initial publishing of document.	Schmidt, S., Stonestreet, B., Rodenberg, S., Medallada, S., Whitmer, K.

Table of Contents

Software Requirements Specifications

1. Introduction

The Software Requirements Specifications (SRS) provides a comprehensive outline of the software requirements for the Boolean Expression Evaluator. Encompassed within this document is all pertinent information essential for the successful development and implementation of the evaluator.

1.1 Purpose

The purpose of the SRS document for the Boolean Expression Evaluator is to provide stakeholders with a detailed description of the software's purpose, functionality, and constraints. It aims for the establishment of a foundational outline of the project objectives and deliverables.

1.2 Scope

The scope of the SRS serves as a guide for the development group and stakeholders by providing a clear outline of the project boundaries regarding areas of concern for the facilitation of efficient and successful software development and implementation.

A list of project-related aspects outlined within this document:

1. **Purpose:** Clear description of the Boolean Expression Evaluator and its intended capabilities.
2. **Functionality:** Detailed explanation of the evaluator's range of logical operations and expression interpretation capacity.
3. **User Interface:** Outline of the software user's experience, including interface design and usability features.
4. **Error Handling:** Description of error detection and handling mechanisms to ensure accurate expression evaluation.
5. **Constraints:** Provision of project constraints, such as programming language limitations, and operating system compatibility.
6. **Assumptions:** Expectations of the design team for the successful end-user experience, including a common understanding of Boolean expressions, logic, and expression format.
7. **Dependencies:** Coverage of any software reliant artifacts and third-party libraries and frameworks necessary for the implementation of the Boolean Expression Evaluator.

1.3 Definitions, Acronyms, and Abbreviations

List of definitions, terms, acronyms, and abbreviations for comprehensive understanding of the SRS:

1. **Boolean Expression Evaluator (BXE):** The software system under development is to determine the result of user provided logic expressions.
2. **Software Requirements Specification (SRS):** The document outlining the project's software requirements.
3. **User Interface (UI):** Referencing the interactive interface of the software user, including its display and navigation controls.
4. **Operating System (OS):** Hardware managing software to serve the execution of the BXE.
5. **Stakeholders:** Individuals and groups with interest or involvement in the project's design, development and implementation including development team, management, end-users.
6. **Boolean Logic:** A branch of mathematics and logic dealing with variables and expressions with one of two values, such as true or false.
7. **Constraints:** Limitations that affect the design, development, and implementation of the project, such as programming language, software and hardware systems, project time allotment, and any budgetary concerns.

1.4 References

- *Use Case Document.pdf*, 24/Mar/24, KFP

2. Overall Description

2.1 Product perspective

2.1.1 System Interfaces

- NA

2.1.2 User Interfaces

- Console, which allows the user to input a Boolean expression and see the result of the expression.

2.1.3 Hardware Interfaces

- Computer screen
- Keyboard
- Mouse

2.1.4 Software Interfaces

- Windows
- Linux
- Other operating systems that can run compiled code.

2.1.5 Communication Interfaces

- NA

2.1.6 Memory Constraints

- The program does not have any specified memory constraints. Thought the program should still comply with basic safety practices and should try to limit the usage of memory as much as possible. This includes making sure there are no leaks, making sure we are destroying unused objects, and that we aren't changing any data on the computer that we shouldn't be changing.

2.1.7 Operations

- The program will be able to parse AND, OR, NOT, NAND, and XOR statements and output the result.

Commented [SS1]: Adding something here

2.2 Product functions

- Handle input while checking to see if it's valid.
- Parse to solve the Boolean expression.
- Must be able to handle parenthesis.
- Output the Boolean expression.

2.3 User characteristics

- Anyone who wishes to solve a Boolean expression.

2.4 Constraints

- Must be written in C++.
- Must be compliable to Linux.
- Must be object-oriented.

2.5 Assumptions and dependencies\

- NA

2.6 Requirements subsets

- NA

3. Specific Requirements

This section will outline the software requirements in detail for functional and non-functional requirements. It will help the designers to design the system of these requirements and the code-runners to verify the system will satisfy it.

3.1 Functional Requirements

3.1.1 Functionality

3.1.1.1 Operator Support

The BXE should be capable of implementing logical operations for:

- AND (&): calculates logical AND if both inputs are TRUE.
- OR (|): calculates logical OR if at least one input is TRUE.
- NOT (!): calculates logical NOT of one input and inverts the truth value.
- NAND (@): calculates logical NAND if both inputs are FALSE.
- XOR ($): calculates logical XOR if one input is TRUE.

3.1.1.2 Expression Parsing

The BXE should be capable of taking simple and complex Boolean expressions and parsing them into computer-readable statements for evaluation.

The precedence of logical operators is as follows from highest to lowest: NOT, AND/NAND/OR/XOR.

3.1.1.3 Expression Input/Truth Variable Assignment

The BXE should have a user-friendly interface for inputting the Boolean expressions and output of True or False.

The BXE should be capable of allowing users to create a variable and assign it a Boolean value. The user should be able to use that variable when passing expressions to the program for evaluation.

3.1.1.4 Evaluation and Output

The BXE should evaluate the output of simple and complex circuit expressions and display it on the user's screen in a human-readable format.

Output should be displayed clearly to the user, indicating whether the Boolean expression evaluates to True or False.

3.1.1.5 Error Handling

The BXE should be capable of detecting errors in Boolean expressions passed to the evaluator.

The BXE should return an error statement that defines the exact nature of the input error.

3.1.1.6 Parenthesis Handling

The BXE will use parenthesis to define precedence and separation of certain terms in the Boolean expression passed in for evaluation.

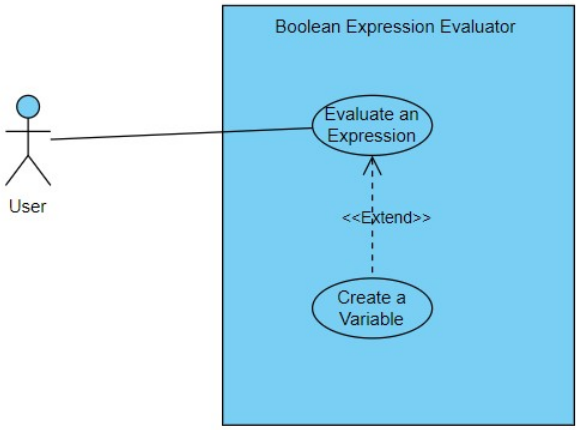
The BXE should be capable of detecting an error with the use of parenthesis in Boolean expressions passed for evaluation.

Commented [SS2]: I'm going to move this info around so we can get our numbering correct

Commented [SS3]: This will need to be more "human readable" like the examples in the slides and the questions in the labs/exams. I'm going to reword some stuff here with the information that you have already placed.

3.2 Use-Case Specifications

3.2.1 UML Use Case Diagram



Refer to the Use-Case Document (Use Cases.pdf) for detailed information.

3.3 Supplementary Requirements

- Performance

  Program should provide timely responses from the input and handle the logical expressions efficiently.

  Responsive times should be acceptable limits from the complex logical expressions.

- Usability

  The user interface should be easy to navigate.

  Tooltips should be provided to help the users understand the program and what to include in the input.

- Software testing

  When running code, it needs to make sure to go through testing to meet the response time of requirements in the load conditions.

  Testing should include load testing and scalability.

- Object-Oriented C++

  This program will be written in C++.

  This program will use Object-Oriented programming to structure the code.

4. Classification of Functional Requirements

Functionality	Type
Operator Support	Functional
Expression Parsing	Functional
Truth Value Input	Functional
Evaluation and Output	Functional
Error Handling	Functional
Parenthesis Handling	Functional
Performance	Non-Functional
Usability	Non-Functional
Software Testing	Non-Functional
Object-oriented C++	Constraint

5. Appendices

5.1 Requirements Appendices:

- **N/A**

5.2 Supplementary Appendices:

- **Project Management Plan:** Managerial documentation of the planned-project approach for the design, development, and implementation of the Boolean Expression Evaluator.
- **Glossary:** Contains definitions for terms, acronyms, and abbreviations referenced in the Project Management Plan.
- **Vision Document (Vision Statement):** Documentation of the overarching objectives of the Boolean Expression Evaluator project.
- **Change Control Procedures:** A document outlining the procedures for the project development group to manage changes to the project, its development, and facilitations.
- **Code Formatting:** A document guide for the format to be followed for the code of the Boolean Expression Evaluator.