MacroHard

Boolean Expression Evaluator Software Development Plan

Version <1.0>

Revision History

Date	Version	Description	Author
<23/02/24>	<1.0>	We decided to work on separate parts of the document on our first meeting.	<name>
<25/02/24>	<1.1>	Added section 2 descriptions.	Garret

Table of Contents

Software Development Plan

1. Introduction

1.1 Purpose

The purpose of this Software Development Plan is to plan out and strategize for the project. It will describe the necessary process to develop said software. To do so we have the following people working on the Software Development Plan:

- The **project manager** uses it to plan the project schedule and resource needs, and to track progress against the schedule.
- **Project team members** use it to understand what they need to do, when they need to do it, and what other activities they are dependent upon.

1.2 Scope

This *Software Development Plan* describes the overall plan to be used by the Boolean Expression Evaluator project. The plans as outlined in this document are based upon the product requirements as defined in the Project Deliverables.

1.3 Definitions, Acronyms, and Abbreviations

KU - University of Kansas
EECS - Electrical Engineering and Computer Science
UML - Unified Modeling Language

See the Project Glossary.

1.4 References

References include:

- 2024-EECS348-project-description.pdf

-Development Case

-Vision

1.5 Overview

The Software Development Plan contains the following information:

Project Description - We are making a Boolean logic simulator in C++.

Team Roles -
Project Leader - Eli Pijanowski
Assistant Project Leader - Srihari Meyoor
Technical Leader - Atharva Patil
Quality Assurance Engineer - Burke Barnett
Team Administrator - Trent Weston
Assistant Team Administrator - Garret Whitson

Project Process - describes cost, phases of the project such as goals and milestones and how the project will carried out

2. Project Overview

2.1 Project Purpose, Scope, and Objectives

Boolean logic simulator in C++

This project will simulate digital logic circuits for Boolean expressions.

2.2 Assumptions and Constraints

The project team consists of 6 members that have the same schedule on Fridays. This assumes that the team will be able to meet at least one time a week. Each member has experience in programming object-oriented languages similar to C++. Each member has access to equipment to work on the project outside of their regular school schedule at home if needed.

2.3 Project Deliverables

The following deliverables will be due on the 16th week of class, sometime between April 30 and May 1:

- The common software engineering artifacts, such as a software development plan, a requirements document, a design document, and test cases.
- A well-documented C++ program that functions as a Boolean expression evaluator with the specified features.
- A user manual or README file explaining how to use your program, including examples of valid expressions and their expected outputs.

Deliverables for each project phase are identified in the Development Case. Deliverables are delivered towards the end of the iteration, as specified in section *4.2.4 Project Schedule*.

1. Operator logic coded to each symbol (gates)
    a. AND => &
    b. OR => |
    c. NOT => !
    d. NAND => @
    e. XOR => $
2. Reading input Boolean expressions
    a. Expressions should use letters and the operators above.
3. Reading input values (T/F) for Boolean expression variables
    a. Assign values of T and F to the variables of the expression.
4. Calculation of logic gates
5. Output final value of function with given variable values
6. Handling errors and showing user how to use correct syntax for input
7. Add functionality for parenthesis for grouping parts of the Boolean expression

2.4 Evolution of the Software Development Plan

The software will be developed in parts as described in section 2.3. See section 4.2 for the project plan as well. If issues or another development plan is needed, the team will meet and discuss revisions.

Software Version	Functionality
0.1	Logic gates
0.2	User input
1.0	Output values
1.1	Error handling
1.2	Grouping operations

3. Project Organization

3.1 Organizational Structure

Our team is composed of six Software Engineering students that share the same lab time. Each member is assigned a role from a list of roles used in an example meeting log. Some members will have specific requirements:

- Project Leader must compile and submit all deliverables on time
- Quality Assurance Engineer will be in charge of testing

However, the scope of this project is small enough that any team member can contribute to any part of the project. Also, each team member will have an equal say in important decisions. These roles and requirements are subject to change as the project progresses. Everyone is reachable through a Discord group.

3.2 Roles and Responsibilities

Person	**Unified Process for EDUcation Role**
Eli Pijanowski	Project Leader
Atharava Patil	Technical Leader
Srihari Meyoor	Assistant Project Leader
Burke Barnett	Quality Assurance Engineer
Garret Whitson	Assistant Team Administrator

4. Management Process

4.1 Project Estimates

This project is expected to cost $0, as we intend to use the materials already available to us as students of KU. Unless otherwise specified, meetings will occur every Friday at 1:00pm CST. Re-estimation of costs will occur if any money is spent during the development of this project. If participants of this meeting find it difficult to attend the weekly meetings, they will voice concerns and a discussion will take place regarding rescheduling the meeting time.

4.2 Project Plan

Scheduled Meeting: Every Friday at 1:00pm CST.

Resources: Personal laptops, KU lab computers, course materials

Project Artifacts: Meeting notes, a well-documented C++ program, a user manual, a project management plan, a requirements document, a design document, and test cases.

4.2.1 Iteration Objectives

For the software itself there will be a single iteration of the software with its full functionality.

4.2.2 Releases

There will be one software release at the end of the semester. This will be the program with all of its intended features and functionality.

4.2.3 Project Schedule

Note: The below table will later be updated to include due dates for future milestones. As of now, the target dates will be listed as TBD.

Boolean Expression Evaluator - Version <1.0>	
Milestone	**Target Date**
Project Plan	2/25/2024
Software Requirements Specs	TBD
Software Architecture	TBD
Test Cases	TBD
User Manual	TBD

4.3 Project Monitoring and Control

4.4 Requirements Management

N/A

4.5 Quality Control

Any errors will be documented and listed via test cases and thorough tests. All deliverables will be tested for quality assurance and ease of use. Any defects found will be documented to be fixed and not

4.6 Reporting and Measurement

N/A

4.7 Risk Management

Risks will be managed through documentation in the Github repository and discussed during team meetings

4.8 Configuration Management

Source Code, Test Cases, and documentation will be managed in a Github repository.

5. Annexes

The project will follow the UPEDU process.

Other applicable process plans are listed in the references section, including Programming Guidelines.

The project is required to be completed in C++.

The program will require operation support, meaning the ability to convert logical operators for AND, OR, NOT, NAND, XOR using $, @, &, and |.

The program is required to interpret logical expression in infix notation, and must respect precedence and parenthesis to understand the order of which logical operators are executed.

The program should also allow a user to create their own truth tables in which they can pass through True or False, by typing T and F.

The program should output a True or False value for the expression given, in a well formatted manner.

The program should be able to identify invalid expressions, missing parentheses, unidentified operators, and other errors, and swiftly prompt the user with an error message. Additionally, The program should be able to interpret an expression with excessive parentheses without returning an error message, while maintaining the order of operations within the expression.