KU EECS 348: Software Engineering 1

Code Commanders
Software Development Plan
Version 1.0

Revision History

Date	Version	Description	Author
02/25/24	1.0	Initial Project Plan	Code Commanders

Table of Contents

Software Development Plan

1. Introduction

1.1 Purpose

The purpose of the *Software Development Plan* is to condense all the information needed to develop and deploy the Boolean Logic Simulator in C++. This top-level plan will be used by the Team Manager to direct the development effort and by other Team Members to refer to so they can follow the requirements defined here.

The following people use 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 plan our team will use to develop our software-based implementation of a Boolean Logic Simulator using C++. The plans as outlined in this document are based upon the product requirements as defined in the Project Outline document provided by Professor Saiedian.

1.3 Definitions, Acronyms, and Abbreviations

See Glossary.

1.4 References

The Documents listed below will be included in the Appendix. The following list is subject to change and may be updated with further revisions.

- **Project Outline**
- **Phase Schedule**
- **Risk List**
- **Configuration Management Outline**

1.5 Overview

This *Software Development Plan* contains the following information:

Project Overview — provides a description of the project's purpose, any constraints applicable to the project, a list of project deliverables and their due dates, and how the Project Manager will be handling the evolution of the *Software Development Plan*.

Project Organization — describes the roles and responsibilities taken on by each team member.

Management Process — explains the schedule of the project, including timelines and workflows.

Applicable Plans and Guidelines — provide an overview of the software development process, including methods, tools and techniques to be followed.

2. Project Overview

2.1 Project Purpose, Scope, and Objectives

The purpose for this project is to develop a usable Boolean calculator for the user to input common logic functions. The objective we want to achieve is that the application is usable for anyone who needs to calculate Boolean expressions in any computer related field. We want the program to run seamlessly by being able to use symbols such as AND, OR, NOT, NAND and XOR. We also want to ensure that it can handle multiple operations and correctly identify and execute parentheses in the correct sequence inside of a complex logic statement.

2.2 Assumptions and Constraints

All members of the team will commit to their primary duties during the project.

This project will be handled in the C++ programming language and will be free to use by anyone. It must be fully operational by the end of Spring 2024, with both the running code and user interface for the program fully functional. All source code and documents will be stored in GitHub at:

https://github.com/TimoHolmes/EECS348-Project

2.3 Project Deliverables

The project will include the documents outlined in the **Project Outline** document, appropriate C++ practices, clear and error-free code, a list of test cases and a user-interface that is easy to maneuver.

2.4 Evolution of the Software Development Plan

The *Software Development Plan* will be revised based on the judgement of the Project Manager and published after agreed upon by the team. Each revision will be logged with the version number and the date it was revised below.

Plan Version:	**Date Revised**
Software Development Plan v1.0	2/25/24
Updated Software Plan	TBD

3. Project Organization

3.1 Organizational Structure

This project team is supervised by Professor Saiedian, and the team is led by Tanner Gurley. The development for this project will be handled among all the members, who have roles outlined in section **3.3 ROLES AND RESPONSIBILITIES**. The team members will work together to carry out the project, including designing and testing the code and ensuring the program is error-free from any defects. All members will collaborate with one another to ensure each phase is set and completed by the expected deadline.

3.2 External Interfaces

N/A

3.3 Roles and Responsibilities

Person	Unified Process for EDUcation Role
Tanner Gurley	Project Manager / Change Control Manager
Timothy Holmes	Analyst / Configuration Manager
James Rossillon	Designer / Integrator
Connor Williamson	Implementor / Tester
Dustin Le	Reviewer / Stakeholder

4. Management Process

4.1 Project Estimates

N/A

4.2 Project Plan

4.2.1 Phase Plan

The Phase Plan and Schedule has been condensed into one document. Please see the **Phase Calendar** in the Annex.

4.2.2 Iteration Objectives

Due to time constraints, the bulk of the team's iteration process will be isolated to their respective phases. In the transition phase after receiving feedback the team will revisit and reevaluate requirements.

4.2.3 Releases

No initial release has been completed yet. Section will be updated as releases start to roll out.

4.2.4 Project Schedule

See **Phase Calendar** located in the Glossary.

4.2.5 Project Resourcing

N/A

4.3 Project Monitoring and Control

The following sections outline the procedures for the team's project monitoring and control.

4.4 Requirements Management

N/A

4.5 Quality Control

Defects will be recorded and tracked as Change Requests. The process is outlined with more info in the **Configuration Management Outline.**

All deliverables are required to go through the appropriate review process, as described in the **Configuration Management Outline**. A review is required to ensure that each delivery is of acceptable quality.

Any defects found during review which are not corrected prior to releasing for integration must be captured as Change Requests so that they are not forgotten.

4.6 Reporting and Measurement

N/A

4.7 Risk Management

Risks will be identified in Inception Phase using the steps identified in the RUP for Small Projects activity "Identify and Assess Risks". Project risk is evaluated at least once per iteration. You can find a current list of assessed risks in the **Risk List** table located in the Annex.

4.8 Configuration Management

Appropriate tools will be selected which provide a database of Change Requests and a controlled versioned repository of project artifacts.

All source code, test scripts, and data files will be included in the team's final project delivery and all Configuration Management materials and guidelines are outlined in the **Configuration Management Outline** document in the Annex.

The Change Requests are reviewed and approved by one member of the project, the Change Control Manager role.

5. Annexes

The following pages of the *Software Development Plan* contain the previously referenced documents and can be identified by their **headings**. They will be included in the order described below.

Project Outline

Phase Schedule

Risk List

Configuration Management Outline

Project Outline

Project Title: Boolean Logic Simulator in C++ (Simplified as Boolean Expression Evaluator)

Spring 2024

Project Objective

This project delves into the world of **digital logic**. You will develop a C++ program acting as a simplified **Boolean logic simulator**. The aim of this project is to develop a program that simulates the behavior of logic circuits, including operations such as AND, OR, NOT, NAND, and XOR. The program should be able to handle complex logic circuits with multiple gates and input/output signals. The project will provide students with comprehensive hands-on experience in software engineering, emphasizing the development process from project planning to fully realized product. While aiming for the functionality of a full-fledged circuit simulator, we'll focus on evaluating **Boolean expressions** for an introductory learning experience.

This project provides an opportunity to explore the fascinating concepts of **logic gates**, **truth tables**, and **expression evaluation**. You'll gain valuable skills in parsing, data structures, algorithm design, and software engineering principles.

Key Features

1. **Operator Support:** Implement logical operations for:

    - AND (&): Returns `True` if both operands are `True`
    - OR (|): Returns `True` if at least one operand is `True`
    - NOT (!): Inverts the truth value of its operand
    - NAND (@): Returns `True` only if both operands are `False` (opposite of AND)
    - XOR ($): Returns `True` if exactly one operand is `True`

2. **Expression Parsing:** Develop a mechanism to parse user-provided Boolean expressions in infix notation, respecting operator precedence and parentheses.
3. **Truth Value Input:** Allow users to define truth values (True/False) for each variable represented by T and F.
4. **Evaluation and Output:** Calculate the final truth value of the entire expression and present it clearly (True or False).
5. **Error Handling:** Implement robust error handling for invalid expressions, missing parentheses, unknown operators, or other potential issues, and provide informative error messages.
6. Parenthesis Handling: Ensure that your program can handle expressions enclosed within parentheses (including seemingly excessive but correctly included pairs of parentheses) to determine the order of evaluation.

Project Guidelines

- Use object-oriented programming principles to structure your code using C++.
- Include comments and documentation to explain the logic and functionality of your program.
- Develop unit tests to verify the correctness of different operator evaluations and complex expressions.
- Ensure that your program provides clear and informative error messages for invalid input or situations.
- Design a user-friendly interface (text-based or graphical) for interacting with the Boolean expression evaluator.

Project Outline

Deliverables

1. The common software engineering artifacts, such as a project management plan, a requirements document, a design document, and test cases. These will be described in separate announcements.
2. A well-documented C++ program that functions as a Boolean expression evaluator with the specified features.
3. A user manual or README file explaining how to use your program, including examples of valid expressions and their expected outputs.

Grading Criteria

Your final product will be evaluated based on the following criteria (a total of 120 points):

- Correctness of expression evaluation (handling of operators, precedence, parentheses) [60 points]
- Robustness and error handling (handling invalid input, syntax errors, etc.) [20 points]
- Code quality, including structure, readability, and comments [20 points]
- Documentation and user manual quality (clarity, comprehensiveness, ease of use) [20 points]

Note: Feel free to explore additional features or optimizations beyond the specified requirements to enhance your project and demonstrate your creativity and engineering skills.

Examples of Valid Expressions

Expression: (T | F) $ F **Evaluation:** True

Expression: ! (T & T) **Evaluation:** False

Expression: (F @ T) | (T @ F) **Evaluation:** False

Expression: (T $ T) & F **Evaluation:** False

Expression: ! F | ! T **Evaluation:** True

Expression: (((((T | F) & F) | (T & (T | F))) @ (T @ T)) $ (! (T | F)))

Evaluation: True

Expression: ((F $ ((T | F) & (F @ (T | F)))) | (T $ (T & F)))
Evaluation: False

Expression: (((! (T $ F)) & (T @ T)) | ((F | T) & (T $ T)))
Evaluation: False

Expression: (((T @ T) $ (F @ T)) | ((!T) & (T | (!T)))) **Evaluation:** True

Expression: ((F @ T) $ (T | (F & F))) & (T & (T @ (!T))) **Evaluation:** False

Project Outline

Examples of Invalid Expressions

1. Missing operand: ! & T [Reason: Missing operand after NOT]
2. Unknown operator: T ? T [Reason: Unrecognized operator symbol]
3. Mismatched parentheses: (T |) False [Reason: Missing closing parenthesis]
4. Circular logic: T = !(T & T) [Reason: Variable defined in terms of itself]
5. Empty expression: [Reason: No operands or operators present]
6. Double operator: T && & F [Reason: Two consecutive AND operators]
7. Missing truth values: X | Y [Reason: Variables without assigned truth values]
8. Inconsistent characters: True | F [Reason: Using both "T" and "F" for variables]
9. Operator after operand: True! [Reason: NOT applied after a value, not before]
10. Invalid characters: a & b [Using lowercase letters instead of "T" and "F"]

UPEDU Phase Calander v1.0

Inception: February 6th

Defined Roles:

Project Manager / Change Control Manager
Analyst / Configuration Manager
Designer / Integrator
Implementor / Tester
Reviewer / Stakeholder

Assigned Members:
Tanner Gurley
Timothy Holmes
James Rossillon
Connor Williamson
Dustin Le

- *Describe initial requirements / stakeholder needs*
- *Determine the scope of our system*
  - *Product Scope(The WHAT)*
    - *Functionality and features that make up the product*
  - *Project Scope(The HOW)*
    - *Work that is required to deliver the results with features defined in the product scope*
- *Identify people, organizations, and external systems*
- *Develop and initial schedule*
- *Tailor Unified Process to meet our needs*

Elaboration: February 27th

- *Begin UML Diagram Models*
  - *Package Diagram*
  - *Use-Case Diagram*
  - *Class Diagram*
- *Review and provide/receive feedback on UML diagrams*
- *Assess any risk factors*
- *Implement an architectural baseline for the final software product*

Construction: March 19th

- *Two weeklong time-boxed iterations to flesh out foundation built in the elaboration phase*
- **First Iteration(Week One):**
  - *Flesh out UML Class Diagrams*
  - *Implement Classes*
  - *Refine User Interface*
  - *Link and execute first run and fix any bugs*
- **Second Iteration(Week Two):**
  - *Add reusability*
  - *Add any additional features agreed upon by team that can be completed within time constraints*

Transition: April 2nd

- *Deploy final project*
- *Test User Experience from with in the team and from outside the team*
- *Create a readMe file to be used as a user guide*
- *Revisit requirements / stakeholder needs and make sure all are met*

Risk List v1.0

ID	Date	Headline	Impact	Description	Mitigation Strategy
1	02/23/2024	Implementation Complexity	high	Technical complexity in Parsing Boolean expressions.	Assign tasks based on skills; conduct code reviews.
2	02/23/2024	Time Constraints	low	Time constraints impact feature completion and the development of new features.	Prioritize core features; agile development approach.
3	02/23/2024	Team Coordination	medium	Difficulties in team coordination may affect implementation.	Weekly team meetings. Utilizing Project management and communication tools like GitHub and discord.
4	02/23/2024	Encountering Bugs	high	Encountering bugs in critical functionality.	Implement an effective testing strategy.

Configuration Management Outline v1.0

Problem and change submissions:

We will use GitHub, which uses Git, a version control system.

- GitHub allows you to submit bug reports, enhancement requests, or general change requests. Users can create new issues, and assign labels: bug, enhancement, and feature requests. and provide detailed descriptions of the problem or proposed change.

- We can have multiple people, and stakeholders, such as developers, testers, and project managers, to collaborate. It allows us to communicate about issues and potential additions we want to make through comments.

- Allows each member to test through pull commands show changes with commits, and push them back to the team repository

Reviewing/Testing:

- GitHub allows you to pull request files of code and commit changes, which other team members can see comments added and which lines were changed or edited. Throughout each phase respective team roles will be making pull requests and testing the current code in place

- For group documents, we will be using Office 365, which everyone can edit and see when someone is on the document making changes.

Changes to Project Plan:

We will have a meeting on Tuesday of each week. All changes to the initial Project Plan will be agreed upon during a meeting or over a Discord server when all members are in attendance.

Naming, Marking, and Numbering Artifacts:

Naming

- The first version will be labeled 1.0, and any large changes will be denoted by .1 (Ex: Version 1.0 will then go to 1.1 after significant information is added to the document)

Marking and Numbering

- Every version or change made will get a comment on the first line of each file. Ex: Author Name, Date changed, time changed)
    - Status after this marking will be added after each team meeting, and we discuss each change and why it was made by the team member.
    - ‘APPROVED’ or ‘DENIED’ will be added on the top of each file after each one is reviewed.

Backup:

Backup

- Regular backups of project artifacts are performed at each Tuesday meeting and after any major code implementations or alterations to any code or document. These backups are stored securely and may include offsite or cloud storage for redundancy.