HL Group

Boolean Logic Simulator in C++
Software Development Plan
Version <1.0>

Revision History

Date	Version	Description	Author
02/20/24	1.0	Initial Template Filled out	Kevinh Nguyen

Table of Contents

Software Development Plan

1. Introduction

This Software Development Plan is for a C++ program that can parse and evaluate Boolean expressions. It includes the purpose, scope, definitions, acronyms, abbreviations, references, and overview of this Software Development Plan.

1.1 Purpose

The purpose of the *Software Development Plan* is to gather all information necessary to control the project. It describes the approach to the development of the software and is the top-level plan generated and used by managers to direct the development effort.

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 overall plan to be used by the Boolean Logic Simulator in C++ project, including deployment of the product. The details of the individual iterations will be described in the Iteration Plans.

The plans as outlined in this document are based upon the product requirements as defined in the *Vision Document*.

1.3 Definitions, Acronyms, and Abbreviations

Change Request: Means the same as Pull Request at https://github.com/KNEternity/348HL/pulls

Issue: https://github.com/KNEternity/348HL/issues

Ticket: Synonymous with Story. Tickets are to be tracked via Issues, which will automatically link to associated Project Board at https://github.com/KNEternity/348HL/projects

References

[This subsection provides a complete list of all documents referenced elsewhere in the **Software Development Plan**. 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.

For the **Software Development Plan**, the list of referenced artifacts includes:

- Iteration Plans

1.4 Overview

This *Software Development Plan* contains the following information:

Project Overview — provides a description of the project's purpose, scope, and objectives. It also defines the deliverables that the project is expected to deliver.

Project Organization — describes the organizational structure of the project team.

Management Process — explains the estimated cost and schedule, defines the major phases and milestones for the project, and describes how the project will be monitored.

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

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.

This is a software engineering project, and as such the emphasis extends beyond the final product; it encompasses the development process. Project deliverables will include a meticulously crafted project plan, a requirements document, a design document that seamlessly aligns with the specified requirements, a set of rigorous test cases derived from the requirements and the design, and ultimately, the fully realized product.

2.2 Assumptions and Constraints

- All team members are students (limited working and meeting times)
- Deadline of ~late April

2.3 Project Deliverables

- Engineering Artifacts/Documentation
    - Project Management Plan
    - Requirements Document
    - Design Specification
- C++ Program to simulate behavior of logical circuits
    - User Manual / README

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*.

2.4 Evolution of the Software Development Plan

The *Software Development Plan* will be revised prior to the start of each Iteration phase. Unplanned revisions to this document can be done at any time for any reason, but common reasons may include fixing typographical errors, improving clarity of document, et. cetera.

3. Project Organization

3.1 Organizational Structure

All members of the team have equal responsibility and there is no organizational hierarchy, except for the Project Leader who generally dictates Project Direction as well. Organizational roles are also flexible, there will be expected variance in role assignment through project iterations (except for Project Leader). All available roles are described now:

- Team Leader: Overall project leadership; Point of Contact with the instructor/client; handles meetings and coordination

- Technical Leader: Make decisions about how to implement requirements, primary approver of code changes

- Quality Assurance Engineer (QA): Automated Q/A Engineer or Manual Q/A Engineer; Write tests and suggests corrections to meet requirements; Secondary approver of code changes.

- Project Leader: Handles dividing up requirements into issues; tracks progress of work; Secondary Team Leaders

Person	**Unified Process for EDUcation Role**
Kevinh Nguyen	Team Leader
Changwen Gong	Technical Leader
Rajkunwar Kaura	Product Owner
Riley England	Quality Assurance Engineer
Kemar Wilson	Project Leader 1
Abdulahi Mohamed	Project Leader 2

4. Management Process

4.1 Project Plan

This section contains the schedule and resources for the project

4.1.1 Iteration Objectives

Document Iteration Objectives

- Software Development Plan
- Project Requirements Document
- Project Architecture and Design Document

Software Iteration Objectives

- Expression Parsing
- Operator Support
  - AND (&)
  - OR (|)
  - NOT (!)
  - NAND (@)
  - XOR ($)
- Truth Value Input
- Evaluation and Output
- Error Handling

- Parenthesis Handling
- Unit and Integration Testing
- User Manual

Each objective may be worked on the same time due to overlap and/or be further broken down

4.1.2 Releases

The latest releases are to be tracked on the online GitHub repository releases page:

https://github.com/KNEternity/348HL/releases

Timeline of releases:

- No Release as of V1.0

4.1.3 Project Schedule

- Software Development Plan Feb 15 – Feb 25
- Requirements Document Feb 26 – March 10
- Project Architecture and Design March 19 – March 31st
- Implement Project March 26 – April 9
  - Expression Parsing TO BE SET
  - Operator Support TO BE SET
  - Truth Value Input TO BE SET
  - Evaluation and Output TO BE SET
  - Error Handling TO BE SET
  - Parenthesis Handling TO BE SET

Due to iterative nature of this document, dates can fluctuate and change based on instructor's order or speed

4.2 Project Monitoring and Control

[The following is a checklist of items to consider:

- *Requirements Management: GitHub Projects will be used to manage tickets. Tickets will be tracked via GitHub Issues which will automatically link to the GitHub Project Board https://github.com/KNEternity/348HL/projects*
- *Quality Control: GitHub's CI/CD (GitHub Actions) will be used for automated quality control. Pull Requests are expected to pass the CI/CD pipeline before merging into the primary/master branch. Manual Q/A testing should be completed on each release.*
- *Risk Management: To mitigate risk, tickets/stories will be measured on their foreseeable risks, such as technical debt to implement, scope/size of ticket, and possible conflicts. This risk will be logged in the GitHub project ticket.*

- *Configuration Management: : Configuration is done through code-as-infrastructure. As such all code changes necessitate a similar code review. This will act as the necessary configuration management review change as well. Code is managed through GitHub Pull Requests.*

4.3 Quality Control

Defects will be recorded and tracked as Change Requests, and defect metrics will be gathered (see Reporting and Measurement below).

All deliverables are required to go through the appropriate review process, as described in the Development Case. The review is required to ensure that each deliverable is of acceptable quality, using guidelines and checklists.

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.

4.4 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 and documented in this table.

4.5 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 are included in baselines. Documentation related to the source code is also included in the baseline, such as design documentation. All customer deliverable artifacts are included in the final baseline of the iteration, including executables.

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

5. Annexes

The project will follow the UPEDU process.

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