LogicSim Labs

EECS 348 Logic Gate Simulator Project

Software Development Plan
Version <1.0>

Revision History

Date	Version	Description	Author
2/25/24	1.0	Documented preliminary information for the Software Development Plan.	Lily Gray, David Sutherland, Daniel Bobadilla, Fatima Avila Cobian, Gaby Kill

Table of Contents

Software Development Plan

1. Introduction

In today's world of circuitry and learning we often have students and employees that often want to simply speed up their workflow, efficiency, and time management in software development. We planned our development strategy for a comprehensive logic gate system around the idea of iterative development. With a sturdy outlined foundation in our understanding of logic gates and programming we plan to create a code outline and add features frequently throughout the process. We want to prioritize our performance, reliability, and responsiveness of the project to ultimately deliver a solution that will assist logic gate users and creators in their process.

Purpose

The purpose of this Software Development Plan is to document the process of creating a logic gate expression calculator/table generator in a group/teamwork-oriented environment. It is meant to describe how we are planning to approach requirements and provides guidelines for members to refer to.

The following people use the *Software Development Plan*:

- Notetakers/documenters will use this plan to keep track of meetings and make sure that each meeting follows our development plan and stays on schedule
- Consultant will use this plan to understand what the task of the overall C++ project should be accomplishing and to keep code within guidelines for minimal errors and refactoring
- Lead code developer will be using this plan to produce as efficient and effective code as possible with assistance from the C++ consultant to follow project purpose and solutions
- Team leader will use this plan to understand what everyone’s role is in the team development process, manage GitHub artifacts, and operate project management

Scope

This Software Development Plan covers the Logic Gate project in C++ that will parse through an expression and evaluate logical operators and their subsequent iterations. The Iteration plans and schedule can be found in section 4.2.4. The Software Development Plan will affect later bodies of work such as the Software Architecture Document, Test Cases, User’s Manual, and Software Requirements document. This document will be pushed to our local GitHub repository.

Definitions, Acronyms, and Abbreviations

1. **AND Operator (&)**: Logic operator that outputs true when both conditions are met

2. **OR Operator (|)**: Logic operator that outputs true when one of two conditions are met

3. **NOT Operator (Inverter) (!)**: Logic operator that inverts a condition

4. **NAND Operator (Not AND) (@):** Logic operator that output true when neither of the two conditions are met

5. **XOR Operator (Exclusive OR) ($)**: Logic operator that outputs true when one and only one condition is met

See the Project Glossary.

References

Future references will be listed here.

Overview

This *Software Development Plan* contains the following information:

Project Overview	—	provides brief overview of the project concerning the scope, purpose, and intentions. Will also contain descriptions of intended project deliverables.
Project Organization	—	describes the project structure and roles of the project team.
Management Process	—	outlines the project's major phases and milestones. Includes estimates of projected costs and timelines.
Applicable Plans and Guidelines	—	provides a summary of the steps involved in the software development process, along with the tools, methods, and strategies used.

2. Project Overview

Project Purpose, Scope, and Objectives

The purpose of the Logic Gate project is to evaluate a series of logical expressions. The program must be able to parse through extraneous parentheses, only evaluating T,F,&,|,@,$ characters in proper order of operations.

At this stage, we plan on implementing a stack-based program that will store each character in a stack and isolate the key components. The project deliverable will be a C++ program that accepts user input of logic expressions and returns the correct evaluation.

Assumptions and Constraints

Assumptions:

- The functional requirements for the software will not change

- The program will be developed in C++
- Secondary to completing the functional requirements, the code should also be efficient in time and memory

Constraints:

- Limited to 5 staff members (Lily, Daniel, David, Fatima, Gaby)
- Equipment is limited to personal computers and desktop computers on the University of Kansas campus
- The project must be completed within one college semester (Spring 2024)
- Artifacts and deliverables must be pushed to local GitHub repository
- Local GitHub repositories should be created and maintained for each team member with backups of all deliverables and artifacts

Project Deliverables

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*.

2.3 Project Deliverables
[A list of the artifacts to be created during the project, including target delivery dates. The text below is provided as an example.] Requirements, design specs, test cases, code

- Required behavior will be its ability to parse through user input and turn “true”, “t”, or “1” into Boolean True
- Make a user-friendly interface (either text or graphical) for the user to see input and output.
- The ability to evaluate based on operator precedence in order of NOT (!), AND (&), NAND (@), OR (|), XOR ($).
- The ability to recognize parenthesis precedence.
- The project must be done in C++ and must be object oriented.
- Comments and notes both in the code and out.

Evolution of the Software Development Plan

The Software Development Plan will be revised before each iteration.

Date	Version	Description	Author
2/25/24	1.0	Initial information about inception stage of Logic Gate project.	Lily Gray, Daniel Bobadilla, Fatima Avila Cobian, David Sutherland, Gaby Kill
3/1/24	1.1	Update structure and project organization if necessary.	Lily Gray, Daniel Bobadilla, Fatima Avila Cobian, David Sutherland, Gaby Kill

3/8/24	1.2	Update structure and project organization if necessary.	Lily Gray, Daniel Bobadilla, Fatima Avila Cobian, David Sutherland, Gaby Kill
3/22/14	1.3	Update with information about Software Requirements from Software Requirements Specifications Document.	Lily Gray, Daniel Bobadilla, Fatima Avila Cobian, David Sutherland, Gaby Kill
3/29/24	1.4	Update structure and project organization if necessary.	Lily Gray, Daniel Bobadilla, Fatima Avila Cobian, David Sutherland, Gaby Kill
4/5/24	1.5	Update with information about Software Architecture, future demos, releases, and risk management.	Lily Gray, Daniel Bobadilla, Fatima Avila Cobian, David Sutherland, Gaby Kill
4/12/24	1.6	Update with Test Cases.	Lily Gray, Daniel Bobadilla, Fatima Avila Cobian, David Sutherland, Gaby Kill
4/24/24	1.7	Update with information on User's Manual, final modifications.	Lily Gray, Daniel Bobadilla, Fatima Avila Cobian, David Sutherland, Gaby Kill

Our team plans to update the Software Development Plan before starting each new iteration phase. The criteria for unscheduled revisions to the Software Development Plan are to document unforeseen changes in team roles and changes in the direction of software development.

3. Project Organization

3.1 Organizational Structure

The project team has been split into 6 roles:

- Team Leader- Responsible for communicating with the TA
- Project Monitor- Makes sure submissions follow correct guidelines
- Code Developers- Primarily responsible for implementing code in C++
- Notetaker- Details meeting agendas and progress, as well as team roles at that time
- Primary Writer- Does most of the writing on specified artifact
- GitHub Manager- Ensures that commits and pushes to GitHub are accurate and do not overwrite the work of other contributing members

Additional project review will be provided by the TA and the professor.

3.2 External Interfaces

[Describe how the project interfaces with external groups. For each external group, identify the internal and external contact names. This should include responsibilities related to deployment and acceptance of the product.]

3.3 Roles and Responsibilities

Person	Unified Process for EDUcation Role
Daniel Bobadilla	Notetaker, Primary Writer, Phone: 9134338960, Email: debobadilla2002@ku.edu
Lily Gray	Notetaker, Primary Writer, Project Monitor, Phone: 9135538867, Email: lbgray@ku.edu)
Fatima Avila Cobian	Code Developer, Contributing Writer, Email: fatima.avila@ku.edu, Phone: (785) 424-5427
David Sutherland	Code Developer, Contributing Writer
Gaby Kill	Team Leader, GitHub Manager cat@ku.edu, 9134981949

The roles of notetaker and project monitor will change frequently, likely after the submission of each artifact.

4. Management Process

Project Estimates

[Provide the estimated cost and schedule for the project, as well as the basis for those estimates, and the points and circumstances in the project when re-estimation will occur.]

Project Plan

Section 4.2 outlines the anticipated project artifacts and iterations, as well as the iteration schedule.

4.1.1 Phase Plan

[Include the following:

- *a Gantt chart showing the allocation of time to the project phases (Not necessarily detailed to the activity level; this type of Gantt Chart is providing along with the Iteration Plans themselves; Provide an Overview of the project Timeline with the major miles stones]*
- *identify **major milestones** with their achievement criteria*

Define any important release points and demos.]

*[If available, refer to the related **Iteration Plan Documents** for more details]*

4.1.2 Iteration Objectives

As stated in section 4.2, 8 total iterations are expected for this project. The first iteration will outline the plan for stack-implementation and necessary methods in Python as a backbone for development in C++. The second iteration will include full functionality of the stack. The third iteration will include limited capabilities to parse expressions. Lastly, the fourth iteration in Python will add error handling for incorrect

inputs. The same iterations process will be done in C++ with the inclusion of full parsing capabilities for a total of 8 total iterations. The complete iteration schedule is shown below:

Iteration 1: Python stack skeleton code completed by 2/25/24

Iteration 2: Python stack implemented by 3/1/24

Iteration 3: Parsing in Python with stack functional by 3/8/24

Software Requirements Specs: Anticipated 3/10/24

Iteration 4: Error handling with test cases completed by 3/15/24

Iteration 5: C++ stack skeleton code completed by 3/22/24

Software Architecture Document: Anticipated 3/24/24

Iteration 6: C++ stack functional by 3/29/24

Test Cases: Anticipated 4/7/24

Iteration 7: C++ parsing with stack and test cases functional by 4/12/24

User's Manual: Anticipated 4/21/24

Iteration 8: C++ error handling completed by 4/26/24

Artifacts created for this project include a Software Requirements Specification Document, Software Architecture Document, documentation of test cases, and a User's Manual. The timeline for these artifacts is shown in section 4.2.4.

4.1.3 Releases

2/25/24 - No current releases, in initial stages for software planning.

4.1.4 Project Schedule

Project Deliverable/Artifact	**Completed By:**
Project Requirements Specifications	3/10/24
Python Implementation	3/15/24
Software Architecture Document	3/24/24
C++ Stack Implementation	3/29/24
C++ Parsing	4/12/24
Test Cases	4/21/24
Software Implementation	4/26/24
User's Manual	5/6/24

Information about potential release points and demos is currently unknown.

4.1.5 Project Resourcing

[Identify the numbers and type of staff required here, including any special skills or experience, scheduled by project phase or iteration.

List any special training project team members will require, with target dates for when this training should be completed.]

Project Monitoring and Control

- Requirements Management: We will use google docs and email to share changes to the product requirements. GitHub will be used to maintain a log of comments and changes throughout the project. A log of each meeting will be recorded and uploaded to the GitHub repository.
- Quality Control: The software will be developed with an iterative approach to maintain functionality at each step. The code will be thoroughly commented on to ensure that the program is meeting its functional standards. To ensure the product is working well, we will test run it with a list of various logic statements (test cases) to make sure it can meet all its functional requirements, this should be done every time a change is made to make sure the changes work.
- Reporting and Measurement: To be added

- *Reporting and Measurement: Describe reports to be generated. Specify which metrics should be collected and why. **OR** if available, refer to the **Project Measurements and Project Measurements** document*

- Risk Management: The major risk is losing code via system crash or accident, or continuing and iterating on a version that is flawed. To prevent the former, we will keep a folder of all the copies on our personal devices, and the latter will be subject to testing (requirements testing, error handling, mashing the keyboard and pressing enter to see what it will do) every single time a change is made to it. Another large risk is making an error in GitHub (forgetting to commit changes, forgetting to push changes, overwriting another person's work). We have a GitHub Manager to ensure that errors like this will be mitigated.
- Configuration management: Product artifacts will be packaged and labelled by turning them into text files with a name and date. All artifacts will be pushed to the remote GitHub repository. The naming conventions will be EECS-348-Project_Document_Name_Version_#. The plan for recovery if a disaster happens is one or every member of the group should keep a personal repository for all the code, and if something gets deleted, we will have copies.

Requirements Management

The requirements for this system are captured in the Vision document. Requested changes to requirements are captured in Change Requests and are approved as part of the Configuration Management process.

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.

Reporting and Measurement

Updated schedule estimates, and metrics summary reports, will be generated at the end of each iteration.

The Minimal Set of Metrics, as described in the RUP Guidelines: Metrics will be gathered on a weekly basis. These include:

Earned value for completed tasks. This is used to re-estimate the schedule and budget for the remainder of the project, and/or to identify need for scope changes.

Total defects open and closed – shown as a trend graph. This is used to help estimate the effort remaining to correct defects.

Acceptance test cases passing – shown as a trend graph. This is used to demonstrate progress to stakeholders.

Refer to the Project Measurements Document (AAA-BBB-X.Y.doc) for detailed information.

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.

Refer to the Risk List Document (CCC-DDD-X.Y.doc) for detailed information.

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.

Refer to the Configuration Management Plan (EEE-FFF-X.Y.doc) for detailed information.

5. Annexes

Here will be additional information and materials for readers of the software development plan. Included should be any technical standards like programming guidelines, design guidelines, etc. As well as any off-topic plans.

The project will follow the UPEDU process.

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

1. Programming Guidelines:
    i. Use descriptive and meaningful names for classes, functions, and variables.
    ii. Follow snake_case naming convention

iii. No single letter variable names or abbreviations for clarity and understanding

iv. Keep indentation consistent for code readability

v. Always refer to peer code for common code writing habits to keep project consistent for readers

vi. **COMMENTS:**

1. Clear concise comments, do not excessively comment code
2. Try using block comments for functions or methods to explain chunks of code if the code seems self-explanatory

vii. Lastly, make sure each function or method has a single task to make debugging simple. Keeping function and method tasks simple makes finding errors simple

2. Design Guideline:

i. Use common data structures such as heap, stack, linked lists, etc. for common understanding among peers

ii. When designing outputs make sure new lines are used to space out responses for easy reading. This can mean designing tables to output clear readable content to descriptive title leading to output response (e.g., “Title: output”)

3. Communication Guidelines:

i. For standard group messaging to communicate include GroupMe and KU emailing.

ii. Communication will be conducted in timely manner to not have last minute changes affect workflow and workload of team members

iii. Communication must be clear and concise to avoid any confusions and misunderstandings

iv. Any communication errors will be clarified and solved in group meetings that are hosted every week