Bonsai Tree

M.A.R.C.O.
Software Development Plan
Version <1.0>

M.A.R.C.O.	Version: 1.0
Software Development Plan	Date: 24/09/23
PROJ2023-SDP-001	

Revision History

Date	Version	Description	Author
24/09/23	1.0	Initial plan set up.	Pashia Vang

M.A.R.C.O.	Version: 1.0
Software Development Plan	Date: 24/09/23
PROJ2023-SDP-001	

Table of Contents

M.A.R.C.O.	Version: 1.0
Software Development Plan	Date: 24/09/23
PROJ2023-SDP-001	

Software Development Plan

1. Introduction

The Software Development Plan is made to outline the development of project M.A.R.C.O. (Multi-Arithmetic Righteously Calculated Operations), a calculator program made using C++ for EECS 348. It will be adjusted as needed by the Project Administrator and used to communicate project guidelines and schedule to team members. This document covers the organization, management process, and guidelines for this project.

This document will be updated as needed throughout this program's development.

1.1 Purpose

The Software Development Plan gathers all the information necessary to supervise 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 will use this Software development plan:

- The **project manager** shall use this to create a project delivery schedule and describe the resources needed to accomplish the project, As well as Tracking the project process against the schedule Determined in this document\
- The **Project Team Members** shall use this to understand what needs to be done, When the need to do it, and what is their responsibility to fulfill.

1.2 Scope

This Software Development Plan describes the plan to be used by the M.A.R.C.O project,
Including The Project overview, Project Organization, and Management process. The details of the individual iterations will be described in the Management Process section of this document.

The plans outlined in this document are based on the product requirements defined in the Project Overview Section.

1.3 Definitions, Acronyms, and Abbreviations

- M.A.R.C.O: Multi-Arithmetic Righteously Calculated Operations

1.4 References

N/A

1.5 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	— Provides an overview of the software development process, including methods, iterations, and tasks due for each iteration.
Annexes	— Provides basic documentation of additional resources and processes used during the development process.
Applicable Plans and Guidelines	— Provide an overview of the software development process, including methods, tools, and techniques to be followed. (Not applicable to this Version of the document)

2. Project Overview

2.1 Project Purpose, Scope, and Objectives

Purpose: Develop a C++ arithmetic expression evaluator.

Scope: Create a software tool for parsing and evaluating mathematical expressions, supporting operators (+, -, *, /, %, and ^) and handling parentheses.

Objectives:

- Implement expression parsing.
- Provide support for specified operators.
- Handle parentheses in expressions.
- Recognize and compute numeric constants.
- Create a user-friendly command-line interface.
- Implement error handling.

2.2 Assumptions and Constraints

Assumptions:

- Assume non-complex arithmetic.

Constraints:

- Must use some sort of implementation of data structures.
- Must be able to handle parenthesis.
- Must be able to handle exponents via “^” and “**” (optional)
- Must handle all operators including modulo
- Must handle errors correctly.
- Must be programmed using C++.

2.3 Project Deliverables

1. The most common software engineering artifacts, e.g., 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 can evaluate arithmetic expressions with the specified operators and features.
3. A user manual or README file explaining how to use your program, including examples.

2.4 Evolution of the Software Development Plan

The *Software Development Plan* will be revised prior to the start of each Iteration phase. This will ensure a gradual buildup through each iteration. With a current total of 8 iterations, each described in the Management Process Section, there should be 8 revisions/updates to this document. Every revision can be of a different magnitude based on the changes needed at that time.

3. Project Organization

3.1 Organizational Structure

Our project team is structured as follows:

- Project Administrator: Responsible for overall project coordination, monitoring progress, and ensuring alignment with project goals and timelines.
- Deputy Administrator: Assists the administrator in project oversight and coordination, ensuring a smooth flow of tasks and alignment with project objectives.
- Quality Assurance Engineer: Focuses on testing and quality control to ensure that the software meets specified requirements and maintains quality standards.
- Software Engineer: Responsible for designing, coding, and testing the software solution, working closely with the development team.
- DevOps Engineer: Manages the development and deployment pipeline, ensuring the software development process is efficient and the deployment is smooth and reliable.
- Kanban with Jira: We're using Kanban and Jira to keep everything organized, prioritize tasks, and track their progress. This approach promotes transparency and adaptability in our software development process.

3.2 External Interfaces

N/A

3.3 Roles and Responsibilities

Pashia Vang	Project Administrator
Anya Combs	Deputy Administrator
Marco Martinez Reyez	Quality Assurance Engineer
Edgar Mendez Cano	Software Engineer
Blake Jesse	Software Engineer
Mikey Cauthon	DevOps Engineer

4. Management Process

4.1 Project Plan

4.1.1 Iteration Objectives

Iteration 1 (September 17th - September 23rd):

- Objective: Lay the foundation for the project and set up the initial project structure.
- Tasks:
  - Research how arithmetic expression evaluation works, focusing on parsing and precedence.
  - Create a GitHub repository for the project and establish the initial project files.
  - Start planning the project, including the project management plan, and requirements.
- Deliverable: GitHub repository with project structure and initial planning.

Iteration 2 (September 24th - September 30th):

- Objective: Implement the core components for expression parsing.
- Tasks:
  - Begin creating a data structure (e.g., stack or tree) to represent expression structure.
  - Continue refining project planning, including design documentation.
- Deliverable: Initial expression parsing functionality and improved project planning.

Iteration 3 (October 1st - October 7th):

- Objective: Extend parsing capabilities and handle operator precedence.
- Tasks:
  - Define the precedence of operators according to PEMDAS rules.
  - Implement logic to evaluate expressions considering operator precedence.
  - Refine design documentation and start outlining test cases.

- Deliverable: Enhanced expression parsing with operator precedence handling.

Iteration 4 (October 8th - October 14th):

- Objective: Focus on parentheses handling and numeric constants.
- Tasks:
  - Develop a mechanism to identify and evaluate expressions within parentheses.
  - Begin recognizing and calculating numeric constants in input expressions.
  - Continue refining design documentation and test case planning.
- Deliverable: Improved parentheses handling and initial numeric constant recognition.

Iteration 5 (October 15th - October 21st):

- Objective: Create a user-friendly command-line interface (UI).
- Tasks:
  - Design and implement a user-friendly CLI for entering expressions.
  - Integrate the UI with the existing parsing and evaluation components.
  - Review and refine test cases.
- Deliverable: Functional command-line interface integrated with expression evaluation.

Iteration 6 (October 22nd - October 28th):

- Objective: Focus on testing, error handling, and code quality.
- Tasks:
  - Conduct comprehensive testing of the entire application.
  - Implement robust error handling to manage scenarios like division by zero or invalid expressions.
  - Review and improve code structure and readability.
- Deliverable: Thoroughly tested, robust application with enhanced code quality.

Iteration 7 (October 29th - November 4th):

- Objective: Finalize documentation and prepare for submission.
- Tasks:
  - Create a user manual or README file explaining how to use the program, including examples.
  - Ensure all project artifacts, including the project management plan and requirements, are in order.
- Deliverable: Completed documentation and prepared project for final submission.

Iteration 8 (November 5th - December 11th):

- Objective: Finalize the project and submit.
- Tasks:

- Perform a final review of the entire project, addressing any remaining issues.
- Submit the fully realized project by the due date.

- Deliverable: Submitted and completed project.

Subject to change.

4.1.2 Releases

N/A

4.1.3 Project Schedule

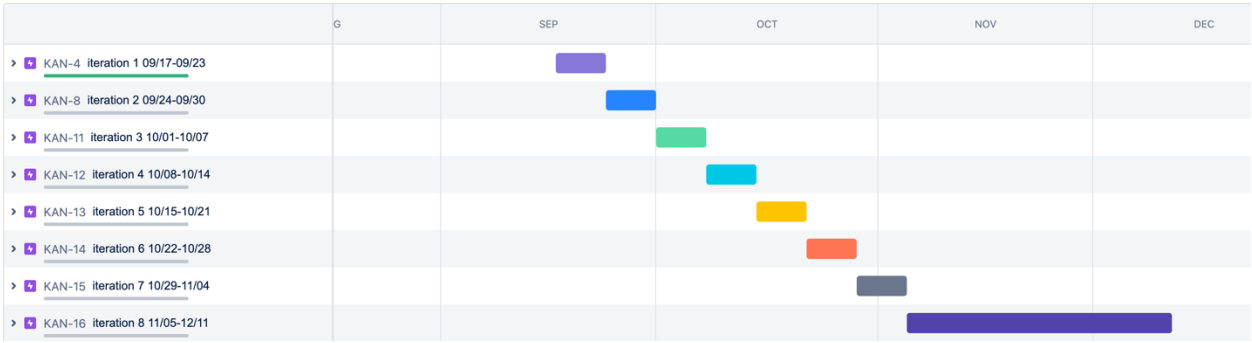

Photo of Project Timeline scheduling in Jira.

4.2 Project Monitoring and Control

4.2.1 Requirements Management

Objective: Follow provided project documentation and specifications.
Approach: Adhere to the plan, requirements, and design. Track changes, communicate deviations, and obtain approvals.
Roles: Project manager, development team, quality assurance team.
Change Control: Document, review, and get approvals for changes.
Version Control: Use version numbers for tracking.
Communication: Maintain clear communication channels.
Documentation: Keep records.
Quality Assurance: Conduct regular checks.

4.2.2 Quality Control

Throughout the project, we'll employ a rigorous quality control process. Each component undergoes:

1. Initial Review: The Quality Assurance Engineer ensures alignment with standards.
2. Team Review: Comprehensive evaluation by the project team.

3. Iterative Refinement: Addressing feedback for improvements.
4. Final Validation: Reassessment by the Quality Assurance Engineer.
5. Branch Management: No pushes to the main branch without successful reviews.

This approach ensures high-quality components and maintains a stable main branch.

4.2.3 Risk Management

Objective: Identify and mitigate project risks.
Approach: Regularly review project phases and designs for potential flaws.
Roles: Project Administrator, development team.
Mitigation: Early risk detection and mitigation measures.
Documentation: Maintain risk logs.

4.2.4 Configuration Management

Our goal is to maintain organization and control in our project. To initiate changes, the project administrator and deputy administrator must provide their approval. We also emphasize comprehensive documentation, consistent naming conventions, and meticulous version tracking. Before pushing updates to GitHub, we ensure that each submission contains adequate information and is presented in an orderly manner.

5. Annexes

The project will follow the UPEDU process.