EECS 348 Group 16

Arithmetic Expression Calculator
Software Development Plan
Version 1.1

Revision History

Date	Version	Description	Author
24/09/2023	1.0	Document creation.	Josh Welicky, Sam Muehlebach, Jennifer Aber, Basim Arshad, Jawad, Ahsan, Mark Kitchin
28/11/2023	1.1	Updated sections 4.2.3 and 4.2.4 to better reflect the updated schedule of the project. Updated section 4.2.2 and Iteration Plans to reflect the updated implementation scheme of the project. Updated 2.3 to better reflect the actual project deliverables.	Josh Welicky

Table of Contents

1. Introduction 4

1.1 Purpose 4

1.2 Scope 4

1.3 Definitions, Acronyms, and Abbreviations..4

1.4 References 4

1.5 Overview 4

2. Project Overview 5

2.1 Project Purpose, Scope, and Objectives 5

2.2 Assumptions and Constraints..5

2.3 Project Deliverables..5

2.4 Evolution of the Software Development Plan 5

3. Project Organization..5

3.1 Organizational Structure 5

3.2 External Interfaces..5

3.3 Roles and Responsibilities..5

4. Management Process..7

4.1 Project Estimates 7

4.2 Project Plan 8

4.3 Project Monitoring and Control 8

4.4 Requirements Management..8

4.5 Quality Control 8

4.6 Reporting and Measurement..9

4.7 Risk Management..9

4.8 Configuration Management 9

5. Annexes..9

Software Development Plan

1. Introduction

1.1 Purpose

The purpose of the *Software Development Plan* is to keep all relevant information concerning the project in one place to make the development process cohesive and successful. This document presents our approach to the project and our strategy for completion, as well as considering all other factors in the project that could either accelerate or slow down the completion of the project.

The following people use the *Software Development Plan*:

- The **Project Leader** uses it to make and complete tasks for project completion, and clarification for project requirements and team roles.
- **Project team members** use it as a resource to complete their various tasks, as well as find other relevant information about the project.

1.2 Scope

This software development plan presents our approach and strategy for the project, particularly its development, testing, deployment and team member roles. Various iterations and deliverables for the project are described within this document. The plan presented in this document is a rough blueprint for our completion of the project in order to meet the project objectives.

1.3 Definitions, Acronyms, and Abbreviations

See the Project Glossary.

1.4 References

Project Description — can be found in the EECS 348 Canvas page under the Project file.

Software Architecture — can be found in the GitHub repository for the project.

Glossary — can be found in Annexes.

Iteration Plans — can be found in Annexes.

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 and composition of the project team.

Management Process — defines the major phases and milestones for the project and describes how the project will be monitored.

Annexes — provides the definitions of various terms used in this document and an in-depth description of each iteration objective.

2. Project Overview

2.1 Project Purpose, Scope, and Objectives

This purpose and objective of this project is to create a C++ program that can perform various arithmetic expressions with operators (+,-,*,/,%, ^), numeric operands, and parentheses, similar to a scientific calculator. The project deliverables consist of a variety of documents containing requirements and design plans, testing cases, a user manual explaining how to use the program, and the final program. Refer to the Project Description document for more information.

2.2 Assumptions and Constraints

Assumptions: Input that represents a valid arithmetic expression, Platform (MacOS, Windows or Linux), Basic arithmetic rules (such as GEMDAS), included C++ libraries, Performance.
Constraints: Timeframe, amount of team members (capped at six), C++, development environment, documentation, proper testing, error handling.

2.3 Project Deliverables

Deliverable 1: Project Management Plan (Due Date: September 24th at 11:59pm)
Deliverable 2: Requirements document
Deliverable 3: Design document
Deliverable 4: C++ ImplementationDecemb
Deliverable 5: Test Case Document
Deliverable 6: User Manual

2.4 Evolution of the Software Development Plan

The *Software Development Plan* will be revised prior to the start of each iteration phase as well as at the beginning of each major phase of project development to reflect any changes made to the plan during project development.

3. Project Organization

3.1 Organizational Structure

The project team consists of six members. Overall completion of the project is overseen and managed by a Project Leader. Different individual team members take leadership roles during each of the main phases of the development project, but all ultimately answer to the Project Leader.

3.2 External Interfaces

Not applicable.

3.3 Roles and Responsibilities

Project Leader: Sam Muehlebach
Contact Information: sam.muehlebach@ku.edu
Availability:
Monday: 12:00 PM - 2:00 PM
Tuesday: 12:30 PM -2:30 PM, 4:00-7:00 PM
Wednesday: 4:00 PM - 7:00 PM
Thursday: 12:30 PM - 2:30 PM, 4:00-5:00 PM

Friday: 12:00 PM – 2:00 PM
Sunday: Anytime
Coursework: EECS 168, EECS 268, EECS 140. EECS 210. Enrolled in EECS 510, EECS 348, EECS 388, EECS 330, EECS 468

Responsibilities:
Managing team meetings:
Crafting objectives for each meeting.
Communicating meeting times and locations.
Managing all official communications with:
Professor Hussein Saiedian
Teaching Assistant Toye Oloko
Settling disputes among team members.
Managing submissions of all project deliverables.
Contributing to the completion of all project deliverables.

Configuration Management Engineer: Jennifer Aber
Contact Information: j670j007@ku.edu
Availability:
Monday: All day via Teams
Tuesday: 11 AM – 1:30 PM, on campus
Wednesday: 12:00 PM – 1:00 PM, on campus, 4:00 – 8:00 PM via Teams
Thursday: 11 AM – 1:30 PM, on campus
Friday: all day, via Teams
Saturday/Sunday: varies
Coursework: Programming I, Programming II, Introduction to Digital Logic and Design (EECS 140). Embedded Systems, Theory of Computing, Data Structures, Programming Language Paradigms, Discrete Structures, Object-Oriented Programming in C++ (JCCC), UNIX Scripting and Utilities (JCCC)
Additional Experience: 2 years (past) work experience as a Technical Writer for Cerner.

Responsibilities:
Ensuring proper formatting and uniformity of project deliverables, including code-base.
Ensuring proper placement and organization of deliverables in the project GitHub repository.
Annotating weekly meetings in the project GitHub Repository.
Contributing to the completion of all project deliverables.

Quality Assurance Engineer: Mark Kitchin
Contact Information: mkitchin@ku.edu
Availability:
Monday: 10:00 AM - 2:00 PM
Wednesday: 3:00 PM - 7:00 PM
Friday: 10:00 AM - 7:00 PM
Coursework: EECS 168, EECS 268, EECS 140. Enrolled in EECS 210, EECS 348, EECS 388

Responsibilities:
Routinely testing project code during the implementation phase.
Managing the division of labor during the testing phase.
Ensuring that all test cases are successfully executed during the testing phase.
Contributing to the completion of all project deliverables.

Requirements Engineer: Jawad Ahsan
Contact Information: j.ahsan02@ku.edu
Availability:

Wednesday 6:00AM-08:00PM
Thursday 6:00PM-08:00PM
Friday 12:45PM – 01:50PM

Coursework: EECS 140, EECS 168, EECS 268, EECS 210. Enrolled in EECS 510, EECS 348, EECS 388, EECS 330.

Responsibilities:

Leading discussions during and related to the requirements engineering phase.
Managing the gathering and documentation of all project requirements.
Coordinating team efforts during the creation of any UML use case diagrams.
Contributing to the completion of all project deliverables.

Design Director: Josh Welicky

Contact Information: joshuawelicky@ku.edu
Availability:

Monday: 1:00 PM – 9:00 PM
Wednesday: 3:00 PM – 9:00 PM
Friday: 10:30 AM – 9:00 PM
Saturday: Anytime

Coursework: Programming I (EECS 169), Programming II (EECS 268), Introduction to Digital Logic and Design (EECS 140). Enrolled in Software Engineering I (EECS 348) and Discrete Structures (EECS 210).

Responsibilities:

Leading discussions during and related to the software design phase.
Coordinating team efforts during the creation of any UML class diagrams.
Manage the creation of the design documentation.
Contributing to the completion of all project deliverables.

Implementation Director: Basim Arshad

Contact Information: baswaqee@ku.edu
Availability:

Wednesday 11:00AM-10:00PM
Thursday 6:00PM-10:00PM
Friday 2:30PM-10:00PM
Sunday 6:00PM-10:00PM

Coursework: EECS 140, EECS 168, EECS 268. Enrolled in EECS 210, EECS 348, EECS 388

Responsibilities:

Coordinating the division of labor for the implementation of each software component.
Working with Project Leader to ensure on-schedule completion of each iteration.
Working with Configuration Management Engineer to:

  Ensure cohesion between software components.
  Ensure adequate commenting of code made during the implementation process.

Contributing to the completion of all project deliverables.

4. Management Process

4.1 Project Estimates

Not applicable.

4.2 Project Plan

4.2.1 Phase Plan

Not applicable.

4.2.2 Iteration Objectives

Iteration 1: Implement the Vector data structure.
Iteration 2: Implement the Stack data structure.
Iteration 3: Implement the Arithmetic data structure, and the basic arithmetic operations.
Iteration 4: Implement the tokenizing functionality in the Driver object.
Iteration 5: Implement the evaluation, conversion, and solving functionality in the Driver object.
Refer to the Iteration Plans and Software Architecture for more details.

4.2.3 Releases

The project management plan and other artifacts may be released prior to the final submission of the project. These artifacts may be revised to reflect changes in the completion of the project. However, no project deliverables will be revised after their final submission on December 3rd, 2023.

4.2.4 Project Schedule

September 24th, 2023: Initial Project Management Plan will be completed.
September 25th, 2023: Requirements Engineering will begin.
October 8th, 2023: Requirements Engineering will be completed.
October 9th, 2023: Software design will begin.
November 10th, 2023: The design phase will end. Project implementation will begin.
November 17th, 2023: The implementation phase will be complete. Testing will begin.
November 27th, 2023: All test cases will be completed. Creation of the user manual will begin.
December 3rd, 2023: All final versions of the project deliverables will be submitted.

4.2.5 Project Resourcing

Not applicable.

4.3 Project Monitoring and Control

4.4 Requirements Management

Not applicable.

4.5 Quality Control

The identification and resolving of defects and errors in the code base of this project will be managed by the Quality Assurance Engineer, both during the implementation and testing phases of this project. The formatting and overall quality of the text-based deliverables, such as the Software Development Plan, will be overseen by the Configuration Management Engineer. These individuals may either resolve errors themselves or organize the joint resolving of larger defects.

4.6 Reporting and Measurement

Not applicable.

4.7 Risk Management

The largest risk to the completion of the project is the possibility of a team member departing prematurely. In the event of a team member's premature departure, all the responsibilities allocated to the departed member will be reallocated equally to the remaining team members.

4.8 Configuration Management

All deliverables, including source code, test cases, and other artifacts, will be stored in a GitHub repository. GitHub is a website that stores version-controlled repositories. The GitHub repository will also store additional artifacts such as weekly meeting logs.

5. Annexes

Glossary:

Class Diagram: A UML diagram that is commonly used to map the entities that will eventually be implemented as classes in a software solution. A class diagram shows the relationships between these entities.

^: A symbol representing the exponent operation.

GEMDAS: An abbreviation of the standard order of arithmetic operations. Short for Grouping, Exponents, Multiplication, Division, Addition, and Subtraction.

%: A symbol representing the modulo operator, which returns the remainder of a division of two numbers.

UML: An abbreviation for Unified Modeling Language. A visual language used for modeling, specification and documentation. Used commonly in software engineering.

Use Case Diagram: A UML diagram that is commonly used to discover and document the functional requirements of a software solution. A use case diagram maps the interactions between external entities and the system, as well as the purpose and results of those interactions.

Iteration Plans:

Iteration 1: Implement the Vector data structure and its methods. Define the functionality necessary for creation, deletion, addition to, and removal from a Vector. Tentatively, these objectives should be ready for testing by November 10th, 2023.

Iteration 2: Implement the Stack data structure and its methods. Define the functionality necessary for creating, adding to, and removal from a Stack using the pre-existing functionality of the Vector object. Tentatively, this objective should be ready for testing by November 11th, 2023.

Iteration 3: Implement the functionality for the +, -, *, /, %, and ^ operators. These methods should be contained in an Arithmetic object, stored in a separate file from the rest of the codebase. Tentatively, these objectives should be ready for testing by November 12th, 2023.

Iteration 4: Implement the functionality required to obtain input from the user and parse it into individual tokens. This code should be contained in a Driver object. Tentatively, these objectives should be ready for testing by November 13th, 2023.

Iteration 5: Implement the functionality required to evaluate user input for errors, convert the parsed input to postfix notation, and solve and display the final result. These should be added to the already existing code base in the Driver object. Tentatively, these objectives should be ready for testing by November 15th, 2023.

The Implementation Director and Project Leader will be responsible for allocating labor to each of these iterations. Individuals will likely be assigned to contribute to specific iterations and/or functionality. The Quality Assurance Engineer will routinely test the code base for errors and defects during implementation, and the Configuration Management Engineer will ensure compatibility of all software components.

If novel requirements are introduced during the implementation of this project, the Implementation Director will be responsible for adjusting the iteration plans to reflect these changes. Consequently, the Software Development Plan will also be revised to reflect these changes.

Refer to the Software Architecture for further explanation of the referenced data objects and methods.