EECS 348

Arithmetic Expression Evaluator
Software Development Plan
Version 1.0

Revision History

Date	Version	Description	Author
9/21/2023	1.0	Update project management plan	Marcos Lepage

Table of Contents

Software Development Plan

1. Introduction

1.1 Purpose

The purpose of the *Software Development Plan* is to outline the goals, scope, management structure, development process, and development iterations of the project as well as identify/describe any artifacts pertinent to the success of the project. It details the overarching project plan in both broad and specific terms with the goal of directing management and development efforts throughout the lifetime of the project. Additionally, this plan aims to facilitate the project by acting as a single source of truth: eliminating ambiguity at all levels of management. It should be noted that this plan is subject to change. The plan is based upon the project team's best estimates, yet we understand that unforeseen circumstances may require flexibility and are willing to make such adaptations as needed.

The following project members use the *Software Development Plan*:

- Project manager(s) (both at the administration and development level) use this plan to estimate the project timeline with the goal of staying on schedule. Additionally, they use the plan to guide development manpower and recourse allocation decisions throughout the project.
- Quality assurance and requirements engineers use this plan to ensure that the deliverables generated throughout the project's lifetime match the functionality and quality promised within the plan. Moreover, they use the plan to avoid quality mistakes altogether and to guide in their resolution if they do occur.
- Project developers and technicians use this plan to understand the broader scope/vision of their work and make design decisions accordingly. Additionally, they use this plan to understand their personal responsibilities and the schedule they expect to uphold.

1.2 Scope

The *Software Development Plan* outlines the plan of action for the EECS348 Arithmetic Expression Evaluator project. The plan includes steps for the inception, development, deployment, and management of the project. Project organization (specifically in terms of management) is described in section three, and planned project iterations are outlined in section four. The requirements and goals of this project are heavily derived from those described in the Project Description file.

1.3 Definitions, Acronyms, and Abbreviations

See the Project Glossary.

1.4 References

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

- Iteration Plans
- Development Case – N/A
- Vision (Scope)
- Glossary
- Risk List Document
- Configuration Management Plan

1.5 Overview

This *Software Development Plan* includes the following subsections:

- Project Overview: Includes the purpose, scope, and objectives of the project as well as an outline of the project's deliverables (i.e. the artifacts to be produced). Additionally, this section explains the project's assumptions and constraints.
- Project Organization: Outlines the organization's management structure, including each role present within the organization and the responsibilities thereof.
- Management Process: Describes overall project plan, iteration by iteration, including scheduled dates for project milestones. Includes the quality control and risk mitigation strategies that will be employed to ensure goals are met in both a timely and satisfactory manner.

2. Project Overview

2.1 Project Purpose, Scope, and Objectives

Our vision is to create a C++ program for arithmetic expression evaluation. Our goal is to provide users with a tool capable of parsing and evaluating expressions, including common operators and parentheses for grouping. Alongside developing the program, we prioritize practical project management, thorough documentation, and robust testing to ensure a seamless and effective development process. Our project aims to enhance our expertise in parsing techniques, data structures, and algorithm design, all while keeping user needs at the forefront, refining the tool based on their feedback and evolving requirements. This journey is about delivering a practical and valuable solution, as well as personal and professional growth.

The project is expected to deliver a comprehensive set of key deliverables, including a project management plan, requirements document, design document, test cases, a fully implemented C++ program for arithmetic expression evaluation, a user manual or README file, error handling documentation, and ongoing code documentation. These deliverables collectively ensure a well-structured, well-documented, and effective project outcome.

2.2 Assumptions and Constraints

Assumptions:

- **Stable Development Environment**: We assume that the development environment, including the C++ development tools and libraries, will remain stable throughout the project. Any significant changes to the development environment may impact project timelines and deliverables.
- **PEMDAS Rules**: The program will adhere to the PEMDAS (Parentheses, Exponents, Multiplication and Division, Addition and Subtraction) rules for operator precedence.
- **Valid Input**: Users are expected to provide valid arithmetic expressions as input. Robust error handling will be implemented to manage invalid expressions, but certain types of input errors (e.g., syntactically incorrect expressions) may not be fully recoverable.
- **Sufficient Testing Resources**: We assume that the testing team will have access to the necessary testing resources and environments to conduct thorough testing, including unit tests, integration tests, and user acceptance testing.

Constraints:

- **Staffing**: The project is constrained by the availability and expertise of the project team members. Any changes in team composition or availability may impact project progress.

- **Development Schedule**: The project schedule is constrained by external factors, such as academic timelines, deadlines, and other academic commitments of the project team members.
- **Equipment and Software**: The project relies on access to the required hardware and software resources, including computers with C++ development environments. Any disruptions in access to these resources may affect project progress.
- **Documentation Standards**: The project adheres to documentation standards and guidelines, and any deviations or delays in documentation may impact the overall project quality and delivery.

2.3 Project Deliverables

Project Management Plan (Due Date: 09/24/2023)
- Description: A comprehensive project management plan outlining project objectives, timelines, roles and responsibilities, risk management strategies, and communication plans.

Requirements Document (Due Date: 10/22/2023)
- Description: A detailed document specifying the functional and non-functional requirements of the C++ arithmetic expression evaluator, including user stories and use cases.

Design Document (Due Date: 10/29/2023)
- Description: A well-documented design document that describes the architecture, data structures, and algorithms used in the program. It should also explain how the design aligns with the specified requirements.

Test Cases and Test Plan (Due Date: 11/26/2023)
- Description: A set of rigorous test cases derived from the requirements and design, along with a test plan outlining the testing strategy, including unit tests, integration tests, and user acceptance testing.

README File (Due Date: 12/03/2023)
- Description: A user-friendly manual or README file that provides clear instructions on how to use the program, including examples of input and expected output.

C++ Program (Due Date: 12/07/2023)
- Description: A fully implemented and well-documented C++ program capable of parsing and evaluating arithmetic expressions with support for the specified operators and features.

Code Documentation (Ongoing)
- Description: Ongoing code documentation, including inline comments and explanations, to ensure that the logic and functionality of the program are well-documented.

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.

Version	Date	Description of Changes	Criteria for Unscheduled Revision/Reissue
1.0	09/24/2023	Initial release of the Software Development Plan.	N/A

3. Project Organization

3.1 Organizational Structure

The structure of this organization is as follows. The team administrator is the primary project manager and has authority over both the direction of the project and the members of the team. As a result, the team administrator has the ability to schedule and organize meetings. The development manager is the primary software management and review authority. The development manager has the ability to allocate developer time towards different aspects of the project. Moreover, the development manager is in charge of reviewing commits and merging pull requests. Next, the quality assurance engineer acts as the secondary review authority: concerned primarily with code and document styling as well as project functionality. Unlike the development manager, the quality assurance engineer's tasks include managing written artifacts. The secretary acts as an assistant team administrator, aiding the team administrator with written tasks. Finally, the developers perform routine software design and creation tasks, as specified by the development manager. Note that all roles participate in creating project deliverables.

3.2 External Interfaces

N/A

3.3 Roles and Responsibilities

Team Administrator: Marcos Lepage

Responsible For:

- Organizing team meetings.
- Taking notes during team meetings.
- Ensuring all roles are equipped to handle their tasks and ensuring all work is running smoothly.
- Working on product deliverables

Development Manager: Ted Athon

Responsible For:

- Reviewing code that is committed to the project.
- Delegating which developers will program what.
- Working on product deliverables.

Quality Assurance Engineer: Andrew Ward

Responsible For:

- Performing routine software functionality checks.

- Ensuring consistent styling between written artifacts.
- Enforcing code style guidelines.
- Implementing software test cases.
- Working on product deliverables.

Secretary: Jaiden Green

Responsible For:

- Documenting software changes and updating relevant artifacts and documents.
- Proofreading all documents associated with the project.
- Working on product deliverables

Developer #1: Amrit Sian

Responsible For:

- Working on product deliverables
- Working with other developers
- Application of mathematical logic and algorithms to ensure the practicality of the calculator.

Developer #2: Dylan O'Brien

Responsible For:

- Working on product deliverables
- Working with other developers
- Application of mathematical logic and algorithms to ensure the practicality of the calculator.

4. Management Process

4.1 Project Estimates

N/A

4.2 Project Plan

Project Part 1: Project Management Plan (September 5 – September 24)

Project Part 2: Project Requirements (September 25 – October 22)

Project Part 3: Project Architecture and Design (October 24 – October 29)

Project Part 4: Project Implementation (October 30 – November 12)

Project Part 5: Project Test Cases (November 14 – November 26)

Project Part 6: Project User Manual (November 28 – December 3)

Project Implementation Due (December 5 – December 7)

4.2.1 Phase Plan

N/A

4.2.2 Iteration Objectives

Iteration 1: Planning and Setup

- **Objective**: Define project scope, requirements, and initial design.
- **Deliverables**:
  - Project Management Plan
  - Requirements Document (including acceptance criteria)
  - Initial Design Document

Iteration 2: Expression Tokenization and Basic Parsing

- **Objective**: Implement tokenization and basic parsing of arithmetic expressions.
- **Deliverables**:
  - C++ code for expression tokenization
  - Basic parsing logic for arithmetic

Iteration 3: Operator Precedence and Parenthesis Handling

- **Objective**: Define operator precedence, handle parentheses, and implement evaluation logic.
- **Deliverables**:
  - Operator precedence rules in C++ code (PEMDAS)
  - Parenthesis handling mechanism

Iteration 4: Numeric Constants and Error Handling

- **Objective**: Recognize numeric constants, error handling for invalid input, and initial user interface.
- **Deliverables**:
  - Numeric constants recognition
  - Robust error handling code
  - Basic command-line interface

Iteration 5: User Interface and Final Testing

- **Objective**: Develop a user-friendly CLI, integrate all components, and perform comprehensive testing.
- **Deliverables**:
  - User-friendly CLI
  - Integration of all components
  - Complete Test Plan (covering all iterations)
  - Comprehensive testing results report

Iteration 6: Documentation and Finalization

- **Objective**: Prepare project documentation, user manual, and finalize the code.
- **Deliverables**:
  - Final C++ code
  - Project Documentation (including code comments)
  - User Manual or README
  - Final Design Document
  - Final Test Plan (with results)

4.2.3 Releases

The software will follow five major releases, from versions 1-5. There may be intermediate versions as bug fixes and feature requests arise, for example, there could be a bug fix version 3.1.0 or 3.0.1 depending on how major the change was, and how it affected functionality in the software.
Version 1.0.0 (Alpha): Basic Arithmetic

- A basic CLI tool that can parse addition, subtraction, multiplication, and division statements.

Version 2.0.0 (Alpha): PEMDAS Handling

- The tool should now be able to handle parenthesis and operator precedence (PEMDAS).

Version 3.0.0 (Beta): Error Handling

- The software should be able to catch and handle exceptions relating to bad input or any other sources of possible error.

Version 4.0.0 (Beta): User Experience and Final Testing

- The tool should have a user-friendly interface with a detailed README file documenting how it works and how the user can use it.

Version 5.0.0 (Release): Final Version

- The software should be fully complete. It should be fully useable, and it should be stable enough that users can use it without issue.
- There should be detailed documentation showing how the software was made and how to use it.

4.2.4 Project Schedule

Management Plan	September 5 – September 24
Requirements	September 25 – October 22
Architecture and Design	October 24 – October 29
Implementation	October 30 – November 12
Test Cases	November 14 – November 26
User Manual	November 28 – December 3
Project Implementation Due	December 5 – December 7

4.2.5 Project Resourcing

N/A

4.3 Project Monitoring and Control

- Requirements Management:

-Requirement Documentation: Detailing, structuring, and listing of requirements within the new software.

-Communication Plan: A clear and concise plan that specifies the ways in which stakeholders are informed of changes and implications.

-Change Management: Person responsible for approving or denying change requests among stakeholders.

-Version Control System: Git will be utilized for managing any adjustments to source code and documentation

- Timing and Methods for Quality Control:

Beginning Development Phase

a. Review of Stakeholder Requirements: Review all stakeholder requirements to ensure all requirements are complete and concise. They must also be documented.

b. Review of Design: Hold times to review the design to check that the proposed design matches with all project goals.

4.4 Requirements Management

N/A

4.5 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.

4.6 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 and documented in this table.

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

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 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

The project will primarily follow the unified software development process.

The software developed in this project will largely follow the Google C++ style guidelines.