NUMBERCRAFTERS

C++ Implemented Arithmetic Expression Evaluator
Software Development Plan
Version <1.4>

Revision History

Date	Version	Description	Author
21/09/2023	1.0	Created title, began describing the software development process using the provided blue text as a base.	C. Hale, I. Aidan, B. Landon, S. Kyle, S. Blake
21/09/2023	1.1	Completed Introduction section.	C. Hale
24/09/2023	1.2	Completed the Overview of the project.	S. Kyle
24/09/2023	1.3	Completed section 4.0 - 4.2.5 of the project. Added iteration and version release timetables.	I. Aidan
24/09/2023	1.3.1	Completed section 3 dealing with organizational structure and assigning roles to each member.	S. Blake
24/09/2023	1.4	Completed section 4.3-4.8 dealing with project monitoring and control.	B. Landon

Table of Contents

Software Development Plan

1. Introduction

This document provides a layout of the plan needed to develop an arithmetic expression evaluator using C++. This document will describe all information necessary to understand and carry out the project. Every detail related to the project will be listed and described in this document.

1.1 Purpose

The purpose of the *Software Development Plan* is to obtain and organize all needed information to describe and carry out the project. It will lay out a plan that describes each step needed to complete the project correctly and on time. It will be used by all team members to understand what is relevant and what is needed to complete the project. The project manager will utilize this document to list needs, dates, and track the progress of the project. All other project members will utilize this document as a tool to help understand what is needed of them and to obtain all other information that may be needed.

1.2 Scope

This *Software Development Plan* outlines the plan being used to deliver the final version of the project. All the iterations will be listed and planned out in the *Iteration Plans*. Also, all plans outlined by this document will be found in the *Vision Document*.

1.3 Definitions, Acronyms, and Abbreviations

See the *Project Glossary*.

1.4 References

Iteration Plans: Each iteration of the project will serve to further develop the functionality of the program. The specific iterations listed in detail can be found in *4.2.2*.

Vision: Our vision for the project to create an arithmetic expression evaluator using C++ that can parse and evaluate arithmetic expressions containing operators as well as numeric constants. Our goals expand beyond coding and expect to have comprehensive planning and documentation.

Glossary: The list of terms allows for complete understanding of the Software Development Plan and what it is describing.

1.5 Overview

The rest of the project development plan contains:

- Project Overview – Gives an in–depth overview of the entire project. Describes purpose, assumptions, complaints, deliverables, etc.
- Project Organization – Describes how the team and project will be organized.
- Management Process - Describes everything management related. Including iterations, monitoring, requirements, quality control, risk management, etc.

2. Project Overview

2.1 Project Purpose, Scope, and Objectives

The purpose of this project is to deliver a full-functional expression evaluator. The scope of this project is

to develop foundational skills and real-world application of software engineering through a software project. It is built to handle a singular user/actor. Our objective is to implement an efficient and organized project plan, execute the plan, and deliver a fully functional software application. There are items we must take into consideration such as the functional requirements, non-functional requirements, and the constraints of this software application. This software should be a fully functional logical expression calculator which means it should be able to calculate any logical expression that is sent to the calculator by an actor. The calculator itself should handle any errors if the actor's input is not valid.

2.2 Assumptions and Constraints

In this application, it is assumed that the user will **NOT** properly format the logical expressions in this calculator. If true, then the system must handle the improper format by either telling the user there was an error, or understanding the error in the logical expression i.e., capitalization errors. The constraints of this project are as follows: The project must be finished before the early December deadline. The logical expression calculator can only calculate specific types of logical expressions. The project must follow legal and ethical guidelines regarding data privacy. The software application may not be compatible with certain other software or systems.

Project Deliverables

The deliverables of this project include a fully functional logical expression calculator that can handle all user input by December 7th and an organizational plan with six iterations and five software release dates, each containing a new version of the software as specified in section 4.2.2-3. Each software release should be bi-weekly, or every other two weeks, from the date 09/28/23. This document will also be a deliverable for the system establishing the foundation of the project and a skeleton structure of our plan. This should be delivered on 9/24/23.

Evolution of the Software Development Plan

Release Information	Date
Create a document illustrating our project plan including an overview, schedule, deadline, and other concepts that are key to the foundation of the project.	09/28/23
Pre-Alpha will be released. A description of Pre-Alpha is found in section 4.2.2.	10/12/23
Alpha Version will be released. A description of the Alpha Version can be found in section 4.2.2.	10/26/23
Beta Version will be released. A description of the Beta Version can be found in section 4.2.2.	11/09/23
Release Candidate Version will be released. A description of the Release Candidate Version can be found in section 4.2.2.	11/23/23
The Final Release Version will be released. A description of the Final Release Version can be found in section 4.2.2.	12/07/23

The Software Development Plan will indeed be revised before each start of the Iteration phase to ensure that it will meet the scheduled due date.

The *Software Development Plan* will be revised prior to the start of each Iteration phase.

3. Project Organization

3.1 Organizational Structure

How the structure of the organization will look:

Project leader of the team is Hale, who will take overall project leadership.

Deputy Administrator is Kyle, who will assist Hale in administrative and management duties.

The configuration manager is Aidan, overseeing development of a project management plan.

Quality assurance engineer is Landon, ensuring functionality.

Blake will take care of scheduling, taking notes of the meetings, and recording a log of team meetings and team member contributions.

Review authorities, also known as Change Control Managers, will be Aidan, Landon, and Blake. They will be responsible for reviewing plans and approving key project decisions.

3.2 External Interfaces

N/A by Marked Plan Document

3.3 Roles and Responsibilities

Person	**Unified Process for EDUcation Role**
Hale	Project Leader
Kyle	Deputy Administrator
Aidan	Configuration Manager, Review Authority
Landon	Quality Assurance Engineer, Review Authority
Blake	Scheduler and Meeting Recorder, Review Authority

Anyone on the project can perform Any Role activities.

4. Management Process

4.1 Project Estimates

N/A by Marked Plan Document

4.2 Project Plan

4.2.1 Phase Plan

N/A by Marked Plan Document

4.2.2 Iteration Objectives

Each iteration of the project will serve to further develop the functionality of the program. These iterations and their accomplishments will include:

- **1st Iteration:** Adding expression parsing and information entrance, as well as a user-friendly graphical interface that allows for easy input (user input iteration)
- **2nd Iteration:** Adding '+' and '-' operator cohesion (addition and subtraction iteration)
- **3rd Iteration:** Adding '*' and '/' operator cohesion (multiplication and vision iteration)

- **4th Iteration:** Adding '%' and '^' operator cohesion (modulo and exponentiation iteration)
- **5th Iteration:** Adding parenthesis handling operator cohesion (PEMDAS iteration)
- **6th Iteration:** Adding numeric constraint cohesion, and error handling for static errors that could be caused by user input (constraint development and error iteration)

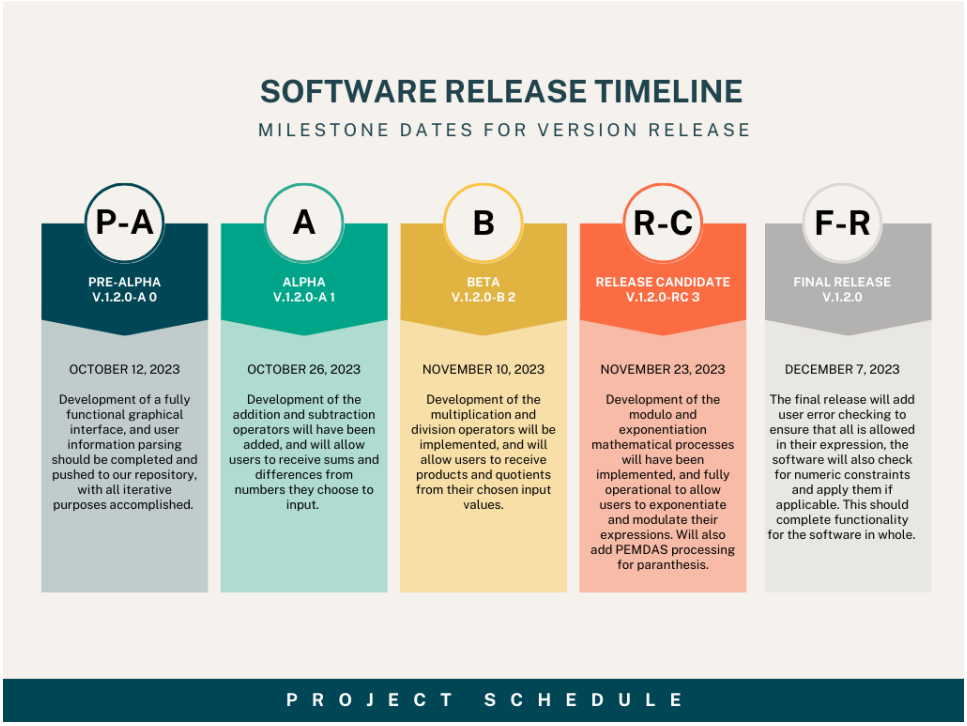
4.2.3 Releases

Each release will increase functionality of the program and will iteratively increase in usefulness to the user. The releases are as follows:

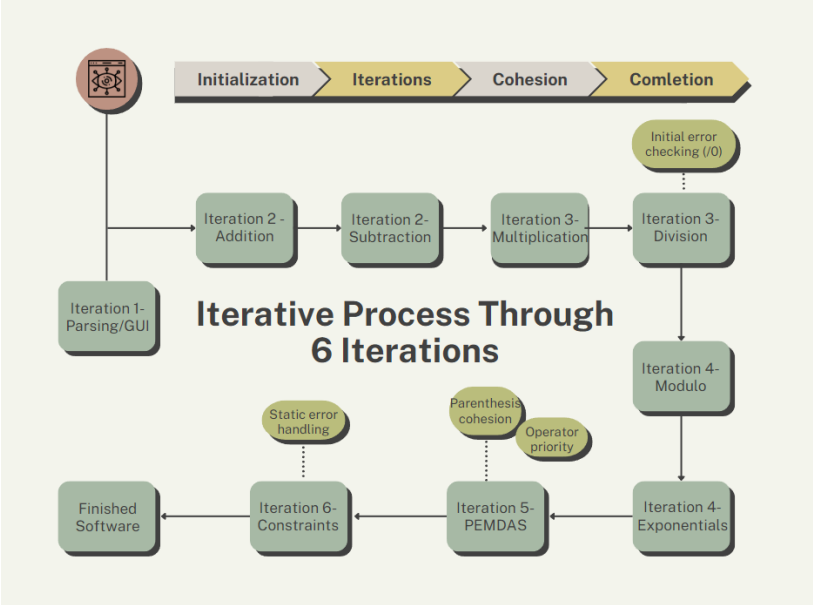
- **Pre-Alpha Version**
  - *1.2.0-a. 0:* The Pre-Alpha version of the software will show basic use cases, like user inputting and output of the input characters, as well as basic information parsing done by the system. It will also display a user-friendly graphical interface for inputting data.
- **Alpha Version**
  - *1.2.0-a 1:* The Alpha version of the software will add user functionality for the addition and subtraction operators, allowing users to input numbers and output the sum or difference of those numbers.
- **Beta Version**
  - *1.2.0-b 2*: The Beta version of the software will add user functionality with multiplication, and division operators. This means that a user will now be able to input numbers, have them parsed, and have a sum, difference, product, or quotient output to their screen.
- **Release Candidate Version**
  - *1.2.0-rc 3*: The Release Candidate version will add modulo, exponentiation, and PEMDAS reading to the software, which will read in for parenthesis added by the user. This will determine if the user wishes to modulate an expression or exponentiate, and then utilizes the order of operations for the expression input while considering any previously implemented operators. This release adds what is likely to be the most functionality.
- **Final Release Version**
  - *1.2.0*: The Final Release version of the software will add support for numeric constraints on the software, in addition to all previous implementations. This implementation will also add error handling for static scenarios in which the user may try to implement an expression that is impossible in the world of mathematics (divide by zero, improper operators etc.).
  -

4.2.4 Project Schedule

The schedule our project follows is intended to proceed along milestones in the semester, as this is a roughly 13 weeklong development process. This assumes the software implementation completion date is December 7th, 2023. The following is our version release timeline:



For iteration development in our process, we will want to follow a different sort of chart, as we have more iterations than we do release versions. The following image is one of how we imagine our iteration timeline to look:



Largely, our scheduling is based on version releases and thus each iteration has no time frame, but rather needs to be completed before the version release. Refer to the version release graph to check when that release date is.

4.2.5 Project Resourcing

N/A by Marked Plan Document

4.3 Project Monitoring and Control

- *Requirements Management:* For each iteration, a series of tests will be run on the current version of software and the results will be recorded. The data will be saved and analyzed across all versions.
- *Quality Control:* Defects are recorded and tracked for version history. Any changes must be reviewed before approval and are recorded. Each version must be inspected and tested before each release.
- *Risk Management:* Risks will be identified prior to each iteration and must be monitored during each version. Risks are reviewed and reprioritized every iteration.
- *Configuration Management:* GitHub is the platform used for version control and history records. Each change request submitted to GitHub will be reviewed and approved before merging to the base code.

4.4 Requirements Management

N/A by Marked Plan Document

4.5 Quality Control

Defects will be recorded and tracked as Change Requests to be reviewed and approved by other members of the team.

All deliverables are required to go through the appropriate review process. The review is required to ensure that each deliverable is of acceptable quality, using guidelines and checklists. Approval will be decided by the Change Control Manager.

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.

Any defects that are reviewed and corrected will be captured and recorded in each version.

Before each version is released, the team will review all corrections and additions made to the software. The team will also review the outline version as described in the *Releases* section to ensure all goals were met and discuss the details of the next version.

4.6 Reporting and Measurement

N/A by Marked Plan Document

4.7 Risk Management

Risks will be identified in Inception Phase. Project risk is evaluated at least once per iteration and documented in this table. Each iteration, previous risks will be reviewed, and new unforeseen risks may be added, meaning risks may need reprioritization. Risks will be monitored by attaching them to a task that is related to said risk.

4.8 Configuration Management

Change requests will be made through GitHub, which will house our software. Change requests will be submitted and reviewed by the Change Control Manager. If the request is approved, it will be merged onto the base code and recorded for version history. If the request is denied, it will need to be edited and submitted again.

Project artifacts are to be named in this format, “ProjectArtifactX” where X is the number of the artifact. Artifacts will be stored on the projects GitHub page and sorted by type, which includes plans, models, components, results, and data, etc.

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.

In the event of a backup restoration, the changes made after the backup will stay recorded in the history. All team members must agree that a backup is necessary before it is done. Any changes made after the backup event, even changes that were made before backing up but after the restoration point, will still need to be submitted, reviewed, and recorded as normal.

5. Annexes

The project will follow the UPEDU process. It will be single-source, and pull from no existing GitHub repositories, or use any code other than that which is created by the members of the group, in C++.

We will utilize the Agile software development model, and create our software fully at first, and add more features to each following iteration as we progress through the project, as can be seen in section 4.2.4.

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