Arithmetic Expression Evaluator in C++

Software Development Plan

Version <1.0>

*[Note: The following template is provided for use with the Unified Process for EDUcation. Text enclosed in square brackets and displayed in blue italics (style=InfoBlue) is included to provide guidance to the author and should be deleted before publishing the document. A paragraph entered following this style will automatically be set to normal (style=Body Text).]*

*[To customize automatic fields in Microsoft Word (which display a gray background when selected), select File>Properties and replace the Title, Subject and Company fields with the appropriate information for this document. After closing the dialog, automatic fields may be updated throughout the document by selecting Edit>Select All (or Ctrl-A) and pressing F9, or simply click on the field and press F9. This must be done separately for Headers and Footers. Alt-F9 will toggle between displaying the field names and the field contents. See Word help for more information on working with fields.] Marked (shaded) areas: items that are OK to leave out.*

Revision History

| **Date** | **Version** | **Description** | **Author** |
| --- | --- | --- | --- |
| 19/09/2024 | 1.0 | Initial Meeting | Evan Rogerson, Mason West, Nick Heyer, Rahul nesan, Ben Haney, Samantha Adorno |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents *[keep this; say N/A when inapplicable]*

[**1.**](#_heading=h.30j0zll) **Introduction 4**

[*1.1*](#_heading=h.1fob9te) *Purpose 4*

[*1.2*](#_heading=h.3znysh7) *Scope 4*

[*1.3*](#_heading=h.2et92p0) *Definitions, Acronyms, and Abbreviations 4*

[*1.4*](#_heading=h.tyjcwt) *References 4*

[*1.5*](#_heading=h.3dy6vkm) *Overview 5*

[**2.**](#_heading=h.1t3h5sf) **Project Overview 5**

[*2.1*](#_heading=h.4d34og8) *Project Purpose, Scope, and Objectives 5*

[*2.2*](#_heading=h.2s8eyo1) *Assumptions and Constraints 5*

[*2.3*](#_heading=h.17dp8vu) *Project Deliverables 5*

[*2.4*](#_heading=h.3rdcrjn) *Evolution of the Software Development Plan 5*

[**3.**](#_heading=h.26in1rg) **Project Organization 5**

[*3.1*](#_heading=h.lnxbz9) *Organizational Structure 5*

[*3.2*](#_heading=h.35nkun2) *External Interfaces 6*

[*3.3*](#_heading=h.1ksv4uv) *Roles and Responsibilities 6*

[**4.**](#_heading=h.44sinio) **Management Process 6**

[*4.1*](#_heading=h.2jxsxqh) *Project Estimates 6*

[*4.2*](#_heading=h.z337ya) *Project Plan 6*

[*4.3*](#_heading=h.3whwml4) *Project Monitoring and Control 7*

[*4.4*](#_heading=h.qsh70q) *Requirements Management 7*

[*4.5*](#_heading=h.49x2ik5) *Quality Control 7*

[*4.6*](#_heading=h.147n2zr) *Reporting and Measurement 7*

[*4.7*](#_heading=h.ihv636) *Risk Management 8*

[*4.8*](#_heading=h.1hmsyys) *Configuration Management 8*

[**5.**](#_heading=h.2grqrue) **Annexes 8**

Software Development Plan

# 

# Introduction

*[The introduction of the* ***Software Development Plan*** *provides an overview of the entire document. It includes the purpose, scope, definitions, acronyms, abbreviations, references, and overview of this* ***Software Development Plan****.]*

## Purpose

*[Specify the purpose of this* ***Software Development Plan****. The text below is provided as an example****.*** *]*

The purpose of this project is to create a calculator to take an input from the user as a string, turn said string into an equation, and return the result. The calculator must be capable of doing addition subtraction multiplication division floor division and exponentials. The calculator should be able to handle parenthesis and order of operations as well as dealing with constants defined earlier.

The purpose of the *Software Development Plan* is to gather all information necessary to control 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 use the *Software Development Plan*:

* The **project manager** uses it to plan the project schedule and resource needs, and to track progress against the schedule.
* **Project team members** use it to understand what they need to do, when they need to do it, and what other activities they are dependent upon.

## Scope

*[A brief description of the scope of this* ***Software Development Plan****; what Project(s) it is associated with and anything else that is affected or influenced by this document. The text below is provided as an example.]*

This *Software Development Plan* describes the overall plan to be used by the <project name> project, including deployment of the product. The details of the individual iterations will be described in the Iteration Plans.  
The plans as outlined in this document are based upon the product requirements as defined in the *Vision Document*.

The Project must be completed by the end of the semester. Additionally we must progressively work on the project in order to not fall behind and meet smaller deadlines (such as our initial plan on Sept 29). We must have a fully completed project submitted by the deadline in such a way that it doesn’t have errors.

This Software Development Plan applies to the development of the Arithmetic Expression Parser project. It covers all aspects of the project, including design, implementation, testing, and error handling. Each phase of the project will be influenced by this plan, ensuring that all requirements are met, and the final product is delivered on time and operates as expected.

## Definitions, Acronyms, and Abbreviations

*[This subsection provides the definitions of all terms, acronyms, and abbreviations required to properly interpret the* ***Software Development Plan****. This information may be provided by reference to the project’s Glossary.]*

See the Project Glossary

.References

*[This subsection provides a complete list of all documents referenced elsewhere in the* ***Software Development Plan****. Identify each document by title, report number if applicable, date, and publishing organization. Specify the sources from which the references can be obtained. This information may be provided by reference to an appendix or to another document.*

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

* *Iteration Plans*
* *Development Case*
* *Vision [you may prepare a vision statement of your own: what your vision for the project is]*
* *Glossary*
* *Any other supporting plans or documentation.*

Our vision for the Arithmetic Expression Parser project is to create a robust, efficient, and user-friendly C++ program capable of parsing and evaluating complex arithmetic expressions with accuracy and precision. The parser will not only handle standard arithmetic operators but will also correctly interpret expressions with parentheses and maintain operator precedence (PEMDAS), ensuring reliability in various scenarios.

This vision includes:

* Developing a C++ program capable of parsing and evaluating arithmetic expressions involving \*+, -, , /, %, and \*\* operators.
* Ensuring the program handles parentheses correctly and follows the PEMDAS order of operations.
* Accommodating numeric constantOur vision for this project is to have a fully functional calculator. This calculator must follow the order of operations and be resistant to crashing.
* s (initially integers, with future flexibility for floating-point numbers).
* Implementing robust error handling to manage invalid expressions or operations such as division by zero.

## Overview

*[This subsection describes what the rest of the* ***Software Development Plan*** *contains and explains how the document is organized. The text below is provided as an example.]*

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  — explains the estimated cost and schedule, defines the major phases and milestones for the project, and describes how the project will be monitored.

Applicable Plans and Guidelines — provide an overview of the software development process, including methods, tools and techniques to be followed.

.

# Project Overview

## Project Purpose, Scope, and Objectives

*[A brief description of the purpose and objectives of this project and a brief description of what deliverables the project is expected to deliver.]*

The purpose of this project is to develop a versatile arithmetic expression evaluator that processes mathematical expressions, parses them, and computes their results following the correct order of operation (PEMDAS). The project scope includes Input handling, parsing, Order of operations, Error handling, Output. The Objectives are Expression parsing, Algorithm design, Handling parentheses and precedence, Robustness and accuracy, Interface.

## Assumptions and Constraints

*[A list of assumptions that this plan is based and any constraints, for example. staff, equipment, schedule, that apply to the project.]*

Some assumptions that this plan is based on are that all team members will contribute equally and to the best of their ability. We will meet in person biweekly and communicate over discord between meetings. Some constraints include varying schedules and levels of experience with C++ coding.

## 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*

The deliverables will be the project management document, the requirements document, the design specs, the implementation and the test cases. The implementation of the calculator must follow the order of operations for addition, subtraction, multiplication, division, modulus, and exponents. Our code must be formatted to have good code practices and be easy to read. The code will adequately comment and have efficient usage of memory allocation. Here are the test cases we will be testing. We know their prior results of these operations and thus can compare our calculator's results to the known values.

* 3 + 4
* 8 - (5 - 2)
* 10 \* 2 / 5
* 2 \*\* 3
* 4 \* (3 + 2) % 7 - 1
* (((2 + 3))) + (((1 + 2)))
* ((5 \* 2) - ((3 / 1) + ((4 % 3))))
* (((2 \*\* (1 + 1)) + ((3 - 1) \*\* 2)) / ((4 / 2) % 3))
* (((((5 - 3))) \* (((2 + 1))) + ((2 \* 3))))
* ((9 + 6)) / ((3 \* 1) / (((2 + 2))) - 1)
* +(-2) \* (-3) – ((-4) / (+5))
* -(+1) + (+2)
* -(-(-3)) + (-4) + (+5)
* +2 \*\* (-3)
* -(+2) \* (+3) - (-4) / (-5)
* 2 \* (4 + 3 - 1
* \* 5 + 2
* 4 / 0
* 5 (2 + 3)
* 7 & 3
* (((3 + 4) - 2) + (1)
* ((5 + 2) / (3 \* 0))
* ((2 -) 1 + 3)
* ((4 \* 2) + ( - ))
* ((7 \* 3) ^ 2)

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*.

## Evolution of the Software Development Plan

*[A table of proposed versions of the* ***Software Development Plan****, and the criteria for the unscheduled revision and reissue of this plan. The text below is provided as an example.]*

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

We will start the unscheduled revision and plans as time goes on in our meeting after this initial meeting ends.

# Project Organization

## Organizational Structure

*[Describe the organizational structure of the project team, including management and other review authorities.]*

We have divided ourselves into separate roles and determined our skills from there and we will all be reviewing and helping each other as time goes on.

## 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.]*

## Roles and Responsibilities *[the more details here, the easier your job; include contact info, availability info, expertise, …]*

*[Identify the project organizational units that will be responsible for each of the disciplines, workflow details, and supporting processes. The text below is provided as an example.]*

Project Manager: Mason West

Contact info: mason.west@ku.edu

* Keeps track of the project schedule, assigns tasks, and makes sure everyone meets deadlines. They also handle any issues that come up.

Scrum Master: Ben Haney

Contact info: benhaney05@ku.edu

* Helps the team follow Agile practices, organizes daily check-ins, and plans work sessions.

Technical Lead: Nick Heyer

Contact info: nickheyer@ku.edu

* Offers technical advice and ensures the team follows good coding practices. They also help solve any technical problems.

Quality Assurance Lead: Evan Rogerson

Contact info: [e890r383@ku.edu](mailto:e890r383@ku.edu)

* Makes sure the project artifacts meet quality standards. During coding, they plan and run tests to find and fix bugs.

Configuration Manager: Rahul Nesan

Contact info: [rahulnesan@ku.edu](mailto:rahulnesan@ku.edu)

* Ensures the integrity, traceability of project configurations and code, facilitating smooth collaboration and minimizing risk in the software development lifecycle.

UX/IU Designer: Samantha Adorno

Contact info: [samantha.adorno@ku.edu](mailto:samantha.adorno@ku.edu), sadorno1

Availability: Thursdays after 4pm, Mondays after 3pm.

* Designs the user interface if we decide to make a GUI. Creates sketches and prototypes.

| **Person** | **Unified Process for EDUcation Role** |
| --- | --- |
|  |  |
|  |  |

Anyone on the project can perform [Any Role](about:blank) activities.

# 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.]*

We plan to complete the project over the course of 6 meetings. We will also communicate over discord to ensure we are all up to date on our objects.

## Project Plan

*[This section contains the schedule and resources for the project.]Project artifact as well as iteration schedules]*

*We will be meeting bi-weekly on Thursday at 4PM and having small meetings in between via zoom and discord.*

### 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]*

### Iteration Objectives

*[Briefly list the objectives to be accomplished for each of the iterations and Refer to the related* ***Iteration Plan Documents*** *for more details.]*

*For iteration 1 the goal is to plan the project and list out requirements. For iteration 2 we will discuss the architecture and design of the parser. For iteration 3 we will go through initial development and prototyping. For iteration 4 we will test our designs/prototypes and figure out any needed modifications and feedback. For iteration 5 we will go through final testing and discussion relating to the project.*

### Releases

*[A brief description of each software release and whether it’s demo, beta, and so on.]*

* *Release 1: Initial Demo (October 22)*

*Type: Demo*

*Description: This release will showcase the basic functionality of the arithmetic expression parser, including tokenization of simple arithmetic expressions and basic evaluation of operations like addition and subtraction. It will focus on demonstrating the program’s core structure and tokenization logic, ensuring that the foundation is solid before further feature implementation.*

* *Release 2: Beta Version (November 12)*

*Type: Beta*

*Description: This release will include full operator support (+, -, \*, /, %, \*\*) and parentheses handling for correct operator precedence (PEMDAS). It will be a feature-complete version of the parser with initial error handling in place, but further testing and optimizations will still be needed. Feedback from testing will be integrated into the next iteration.*

* *Release 3: Final Version (December 1)*

*Type: Final Release*

*Description: This final release will be a fully functional version of the arithmetic expression parser, including comprehensive error handling (e.g., division by zero, invalid expressions) and performance optimizations. It will be thoroughly tested with all features working as expected. The release will be accompanied by a user manual/README, design documentation, and a final set of test cases*

### Project Schedule

*[Diagrams or tables showing target dates for completion of iterations and phases, release points, demos, and other milestones.] [Limit to major project milestone, e.g., requirements, design, implementaiotn, and testing]*

| **Milestone** | **Target Date** | **Description** |
| --- | --- | --- |
| Project Management Plan | September 20 | Complete the document with the roles, goals and ideas. |
| Requirements Document | September 30 | Complete the requirements document, clearly defining what the program needs. |
| Design Document | October 10 | Finalize the design of the system, including diagrams. |
| Prototype Implementation | October 20 | Implement basic functionality to ensure a working foundation. |
| Milestone 1 Demo | October 22 | Demo basic features.. |
| Full Implementation | November 10 | Complete the implementation with operator precedence, parentheses handling, and error checking. |
| Milestone 2 Demo | November 12 | Demo the completed parser with all features and error handling. |
| Testing and Bug Fixing | November 20 | Conduct thorough testing and debugging of the program. |
| Final Submission | December 1 | Submit the final C++ program, README file, and all required documentation. |
| Final Demo | December 3 | Presentation |

### 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.]*

We will have 6 people working on this project. All members will have prior coding experience.

## Project Monitoring and Control

*[The following is a checklist of items to consider:*

* *Requirements Management: Specify the information and control mechanisms which will be collected and used for measuring, reporting, and controlling changes to the product requirements.*
* *Quality Control: Describe the timing and methods to be used to control the quality of the project deliverables and how to take corrective action when required. Include techniques, metrics, criteria, and procedures used for evaluation— this will include walkthroughs, inspections, and reviews. Note that this is in addition to the Test Plan, which is not enclosed in the Software Development Plan.*

* *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: Describe the approach that will be used to identify, analyze, prioritize, monitor and mitigate risks. If available, refer to the* ***Risk List*** *document.*

We will be prioritizing saving frequently and making sure to not make any decisions on our own without talking to one of our team members first. We want to minimize major mistakes and identify and problems that can lead to the elimination of our code and read and rewrite as applicable.

* *Configuration Management: Describe the process by which problems and changes are submitted, reviewed, and dispositioned. Describe how project or product artifacts are to be named, marked, and numbered, including system software, plans, models, components, test software, results and data, executables, and so on. Describe retention policies, and the back-up, disaster, and recovery plans.* ***OR*** *if Available, Refer to the* ***Configuration Management Plan*** *document*

*The text that follows is provided as an example.]*

## **Requirements Management**

Any and all changes to requirements will be approved by all team members and documented.

## **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.

We will record each of the errors we encounter while writing the code and our solutions for how we fixed them. By keeping a detailed error log, we will ensure that if 1 person on our team encounters an error, others will know what to do if they find the same error. This will smoothline the debugging process.

## **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.

We will track the results of our test cases and include them in our final report.

*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.

We will run numerous test cases to ensure that the program does not crash on the user. We will also ensure the program does not get stuck in an infinite loop as this causes excessive cpu usage.

*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.*

*Change Process:*

* *Submission: Any issues or change requests are logged in Git.*
* *Review: Changes are reviewed during weekly team meetings to assess their impact and priority.*
* *Approval: Minor changes are approved quickly, while significant changes are discussed and voted on by the team.*

*Naming & Versioning:*

* *Code: module\_function\_vX.X.cpp (e.g., parser\_eval\_v1.0.cpp)*
* *Documentation: DocType\_Project\_vX.X.extension (e.g., Design\_ExprParser\_v1.0.pdf)*
* *Test Cases: Test\_Feature\_TestID.extension (e.g., Test\_Parentheses\_TC101.txt)*
* *Executables: Project\_vX.X\_OS.exe (e.g., ExprParser\_v1.0\_Win.exe)*

*Versioning will follow semantic versioning (Major.Minor.Patch) where major changes increment the first number, minor updates increment the second, and patches increment the third.*

*Retention:*

* *All project files will be retained in the GitHub repository for the duration of the project and archived for one year afterward.*

*Backup & Recovery:*

* *Backups: GitHub provides automatic version control and cloud storage for continuous backups.*
* *Recovery: In case of issues, the latest stable version will be restored from the repository within 24 hours.*

# 

# Annexes

*[Additional material of use to the reader of the* ***Software Development Plan****. Reference or include any project technical standards and plans which apply to this project. This typically includes the Programming Guidelines, Design Guidelines, and other process guidelines. The text that follows is provided as an example.]*

The project will follow the UPEDU process.

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

Programming Guidelines:

* Follow Google’s C++ Style Guide for consistent, readable code.
* Use proper error handling and add comments where needed.

Design Guidelines:

* Stick to object-oriented design and modular structure.
* Use UML diagrams as per standard practices.

Testing Guidelines:

* Use Google Test (GTest) for unit tests.
* Cover all cases, especially operator precedence and parentheses.