Arithmetic Expression Evaluator in C++

Software Development Plan

Version 1.0

Revision History

| **Date** | **Version** | **Description** | **Author** |
| --- | --- | --- | --- |
| 21/Sep/23 | 1.0 | <details> | Taha Khalid |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[**1.**](#_30j0zll) **Introduction 4**

[*1.1*](#_1fob9te) *Purpose 4*

[*1.2*](#_3znysh7) *Scope 4*

[*1.3*](#_2et92p0) *Definitions, Acronyms, and Abbreviations 4*

[*1.4*](#_tyjcwt) *References 4*

[*1.5*](#_3dy6vkm) *Overview 5*

[**2.**](#_1t3h5sf) **Project Overview 5**

[*2.1*](#_4d34og8) *Project Purpose, Scope, and Objectives 5*

[*2.2*](#_2s8eyo1) *Assumptions and Constraints 5*

[*2.3*](#_17dp8vu) *Project Deliverables 5*

[*2.4*](#_3rdcrjn) *Evolution of the Software Development Plan 5*

[**3.**](#_26in1rg) **Project Organization 5**

[*3.1*](#_lnxbz9) *Organizational Structure 5*

[*3.2*](#_35nkun2) *External Interfaces 6*

[*3.3*](#_1ksv4uv) *Roles and Responsibilities 6*

[**4.**](#_44sinio) **Management Process 6**

[*4.1*](#_2jxsxqh) *Project Estimates 6*

[*4.2*](#_z337ya) *Project Plan 6*

[*4.3*](#_3whwml4) *Project Monitoring and Control 7*

[*4.4*](#_qsh70q) *Requirements Management 7*

[*4.5*](#_49x2ik5) *Quality Control 7*

[*4.6*](#_147n2zr) *Reporting and Measurement 7*

[*4.7*](#_ihv636) *Risk Management 8*

[*4.8*](#_1hmsyys) *Configuration Management 8*

[**5.**](#_2grqrue) **Annexes 8**

Software Development Plan

# 

# Introduction

The Software Development Plan (SDP) presented herein serves as a guiding blueprint for the meticulous execution of the "Arithmetic Expression Evaluator in C++" project. This document aims to articulate the purpose, scope, and framework of the project's software development process. It offers a structured approach to the design, implementation, and testing phases, ensuring that the team aligns its efforts with the specified project requirements and objectives. Within this SDP, key elements such as project objectives, features, tasks, guidelines, and grading criteria are delineated to facilitate a comprehensive understanding of the project's trajectory. Additionally, the definitions section provides clarity by defining key terms such as "PEMDAS" (Parentheses, Exponents, Multiplication and Division, Addition and Subtraction) and "Tokenization," ensuring a shared lexicon among project stakeholders. As a foundational document, the SDP not only offers a comprehensive view of the project but also serves as a reference point for all stakeholders involved, promoting a unified understanding of the project's goals and expectations.

## Purpose

The primary purpose of this Software Development Plan (SDP) is to serve as a comprehensive document that outline the strategy, methodology, and key details necessary to manage control the “Arithmetic Expression Evaluator in C++” project. It provides a structured approach to the software development process, ensuring clarity and alignment with project objectives. The SDP serves as the top-level plan that guides project managers and team members throughout the project's lifecycle.

The SDP play a pivotal role for various parties involved in the project:

* The **project manager** utilizes the SDP as a tool for project planning. It aids in the development of project schedules, resource allocation, and progress tracking against established timelines. It serves as a critical reference point to maintain project control and ensure adherence to project control and ensure adherence to the defined plan.
* The **project team members** rely on the SDP to gain a clear understanding of their individual roles and responsibilities within the project. It provides them with a comprehensive view of project activities, timelines, and dependencies, helping them effectively contribute to project success.

## Scope

This Software Development Plan outlines the overarching plan to be employed for the “Arithmetic Expression Evaluator in C++'' project, encompassing all aspects of the project’s Lifecyle, from conceptualization to deployment. The document also addresses the scope of influence and affected components associated with this project.

## Definitions, Acronyms, and Abbreviations

PEMDAS: An acronym representing “Parentheses, Exponents, Multiplication and Division, Addition and Subtraction” If defines the order of operations for evaluating arithmetic expressions.

Tokenization: The process of breaking down an arithmetic expression into individual tokens or elements for further analysis and evaluation.

## References

* Iteration Plan
* Development Case
* Vision Document
* Glossary

## Overview

This *Software Development Plan* contains the following information:

Project Overview  — The primary purpose of this project is to design and implement a robust C++ program capable of parsing and evaluating arithmetic expressions. This program will handle a wide range of arithmetic operators, including addition (+), subtraction (-), multiplication (\*), division (/), modulo (%), and exponentiation (^). Furthermore, it will accommodate numeric constants and parentheses for grouping and precedence. Beyond building a functional expression evaluator, this project serves as a software engineering exercise, providing an opportunity to reinforce knowledge in parsing techniques, data structures, and algorithm design. The project's scope encompasses the entire software development lifecycle, commencing with project planning and concluding with the deployment of the final product. It covers the following key aspects: Expression Parsing, Operator Support, Parenthesis Handling, and Numeric Constants.

The key objectives of this project are as follows.

* Develop a tokenization mechanism for input expressions.
* Design and implement an appropriate data structure (e.g., stack or tree) to represent the structure of expressions.
* Define operator precedence according to the PEMDAS rules.
* Implement the logic to evaluate expressions while respecting operator precedence.
* Create a mechanism to identify and evaluate expressions within parentheses.
* Recognize and calculate numeric constants in the input expressions.
* Establish a user-friendly and legible command-line interface to enable users to input expressions and view calculated results.
* Implement robust error handling to manage scenarios such as division by zero or invalid expressions.

The project is expected to deliver the following key outcomes:

* A meticulously documented and well-structured C++ program capable of parsing and evaluating arithmetic expressions, adhering to the specified operators and features.
* A user manual or README file, providing clear instructions on how to use the program and offering illustrative examples.
* A suite of rigorous unit tests designed to validate the correctness and reliability of the expression evaluator.

Project Organization  — N/A

Management Process  — N/A

Applicable Plans and Guidelines — N/A

# Project Overview

## Project Purpose, Scope, and Objectives

## Assumptions and Constraints

Purpose:

Develop a C++ program to parse and evaluate arithmetic expressions, reinforcing software engineering concepts.

Scope: Encompasses the full software development lifecycle, including a user-friendly command-line interface and error handling.

Objectives:

- Efficient expression parsing.

- Support for operators (+, -, \*, /, %, ^).

- Handling of parentheses.

- Numeric constant recognition.

- User-friendly interface.

- Robust error handling.

Expected Deliverables:

- Well-documented C++ program.

- User manual or README.

- Suite of rigorous unit tests.

## 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.]*

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*.

Our project involves a range of deliverables that mark significant milestones throughout the development process. These deliverables, outlined in our Development Case, include crucial artifacts such as code implementations, design documentation, test reports, and user guides. Each deliverable is associated with a specific project phase or iteration and is scheduled to be completed towards the end of that phase, as indicated in our Project Schedule. This structured approach ensures that we maintain clarity regarding what needs to be accomplished at each stage and enables us to track our progress effectively.

## Evolution of the Software Development Plan

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

Our Software Development Plan (SDP) is a dynamic document that evolves in tandem with our project's progress. We have established a practice of revising the SDP prior to the start of each Iteration phase. These revisions are essential to ensure that the SDP remains aligned with the project's current objectives and requirements. The criteria for unscheduled revision and reissue of the plan are primarily driven by significant changes in project scope or goals. Any deviations from the original plan, such as major shifts in objectives or resource allocation, would prompt an unscheduled revision to maintain consistency and transparency in our project management approach. This iterative approach to plan evolution is integral to our commitment to delivering a successful "Arithmetic Expression Evaluator in C++" project.

# Project Organization

## Organizational Structure

* Our project team for the "Arithmetic Expression Evaluator in C++" consists of all five team members who will actively collaborate on various aspects of the project. This inclusive structure promotes teamwork and collective responsibility.
* Responsibilities within our team are distributed based on expertise and interest, ensuring that each team member can contribute effectively to the project. The shared sense of ownership fosters collaboration and mutual support.

## External Interfaces

*N/A*

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

| **Person** |  | **Unified Process for EDUcation Role** |
| --- | --- | --- |
| Siddh Bharucha  Contact : [siddh@ku.edu](mailto:siddhb@ku.edu)  Phone # 913-228-4427  Availability: Monday through Sunday 6:30 and after.  Computer Science Major - Junior 2025  Relevant Course Work: EECS 168, 268, 210, 468 and 388. Currently enrolled in EECS 510, 330, 348  Programming languages : Python, Java, Javascript, Angular, Angular JS, React, C, CSS, HTML |  |  |
| Gregory Markose  Contact : [gregorymarkose@ku.edu](mailto:gregorymarkose@ku.edu)  Phone # 785-979-5411  Availability : Monday through Sunday 5pm and after.  Computer Science Major - Junior 2025  Relevant Coursework : EECS 168, EECS 140, EECS 268. Currently enrolled in EECS 210, EECS 388, EECS 348, EECS 468.  Programming Languages : Python, JavaScript, React, Node.js, C++, HTML & CSS. |  |  |
| Taha Khalid  Contact : taha@ku.edu  Phone # : 913-337-7389  Availability : Monday to Sunday After 5pm, Thursday after 6pm.  Computer Science Major - Sophomore 2026  Relevant Coursework : EECS 168, EECS 140, EECS 268. Currently enrolled in EECS 210, EECS 348.  Programming Languages : Python, JavaScript, C++, HTML & CSS. |  |  |
| Saurav Renju  Email: [saurav.renju@ku.edu](mailto:saurav.renju@ku.edu)  Availability: Monday to Friday, Sunday(after 5 pm) & Saturday (after 8 pm)  Computer Science Major - Sophomore 2026  Relevant Coursework: EECS 168, EECS 140, EECS 268.  Currently enrolled in EECS 210 & EECS 348  Programming Languages: Python,basic knowledge of C++ & CSS |  |  |
| Alec Slavik  Email: [alec.slavik@gmail.com](mailto:alec.slavik@gmail.com)  Availability: Monday to Friday, Sunday (after 5 pm) and Saturday (after 5 pm)  Computer Science Major -  Sophomore 2026  Relevant Coursework: EECS 168, EECS 140, EECS 268.  Currently enrolled in EECS 210 and EECS 348  Programming Languages:  Python, just a little knowledge on C++. |  |  |
| Divit Kannan  Email: [dkannan@kul.edu](mailto:dkannan@kul.edu)  Availavility: Monday to Friday (after 5 PM), Saturday, Sunday (after 5 pm)  Computer Science Major - Sophomore 2026  Relevant Coursework: EECS 168, EECS 140, EECS 268.  Currently enrolled in EECS 210 and EECS 348  Programming Languages: Python, React, HTML, HTML5, CSS, JavaScript, C++, Typescript |  |  |

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

# Management Process

## Project Estimates

*N/A*

## Project Plan

Our project plan is the guiding framework for developing the "Arithmetic Expression Evaluator in C++." It provides a well-defined timeline, assigns roles and responsibilities, and outlines crucial phases, beginning with project initiation and concluding with deployment. A visual Gantt chart offers a clear overview of the schedule, aiding in tracking progress and maintaining alignment among team members. Major milestones serve as key achievements, and we've identified release points, including demos and the final product release, each with specific criteria for advancement. This comprehensive plan ensures efficient project management, fosters effective teamwork, and keeps us firmly on the path toward achieving our project's objectives.

### 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 Objective

In our project, we embrace an iterative approach to develop the "Arithmetic Expression Evaluator in C++." Each iteration has specific objectives that steer our progress. We begin with project initiation, defining scope and requirements, followed by designing the software and creating a detailed project plan. Subsequent iterations focus on core functionality implementation, operator expansion, parenthesis handling, and recognition of numeric constants. We then turn our attention to user interface development, robust error handling, and thorough testing. With an emphasis on documentation and user guidance, we prepare for the final phase of testing and deployment. These objectives are collaboratively defined among our team members to ensure a shared vision and effective coordination. For detailed task breakdowns and timelines, please consult our Iteration Plan Documents.

### Releases

Our project will have multiple software releases, and all team members will actively contribute to these release phases. These releases serve different purposes, including demos, beta versions, and the final product release.

The collective effort of all team members ensures that each release meets our project's quality and functionality standards.

### Project Schedule

Our project schedule is the heartbeat of our "Arithmetic Expression Evaluator in C++" development. It offers a visual representation of our timeline, including iterations, phases, milestones, and release points. This schedule acts as our compass, helping us stay on course by providing a clear view of target dates for crucial project activities. It ensures that all team members are aware of deadlines, facilitating effective coordination and progress tracking. With its detailed insights into our project's temporal aspects, the schedule serves as a fundamental tool for ensuring that we adhere to our project's timelines and goals.

### 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.]*

## 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.*
* *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**

The requirements for this system are captured in the Vision document. Requested changes to requirements are captured in Change Requests, and are approved as part of the Configuration Management process.

## **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.

## **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.

*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.

*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.*

# 

# 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.