

**Arithmetic Expression Evaluator in C++
Software Development Plan**
Version 1.2

Arithmetic Expression Evaluator in C++	Version: 1.2
Software Development Plan	Date: 29/September/24
01-Project-Plan.doc	

Revision History

Date	Version	Description	Author
09/20/2024	1.0	All team members added member profiles, Arpa added team roles, Peter added communication plan, ____ added weekly meeting details section Location: LEEP	Addie, Peter, Arpa, Emily, Hart
09/25/2024	1.1	All team members discussed and were assigned sections Peter - 2.2 and risk management (4.7 and part of 4.3) Hart - 2.3 - 2.4 and quality control (4.5 and part of 4.3) Emily - 1.3-1.5 and requirements management (part of 4.3) Arpa - 1.1-1.2 and 2.1 and configuration management (4.8 and part of 4.3) Addison - 3.1-3.3 and 4.2 (3.3 should be done but add your info; 3.2 Location: LEEP	Addie, Peter, Arpa, Emily, Hart
09/29/2024	1.2	Completed Project Plan	Addie, Peter, Arpa, Emily, Hart

Arithmetic Expression Evaluator in C++	Version: 1.2
Software Development Plan	Date: 29/September/24
01-Project-Plan.doc	

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 6

3. Project Organization..... 7

3.1 Organizational Structure 7

3.2 External Interfaces..... 7

3.3 Roles and Responsibilities..... 7

4. Management Process 8

4.1 Project Estimates 8

4.2 Project Plan 8

4.3 Project Monitoring and Control 7

4.4 Requirements Management..... 9

4.5 Quality Control 9

4.6 Reporting and Measurement..... 10

4.7 Risk Management..... 10

4.8 Configuration Management 10

5. Annexes 10

5.1 Glossary

Arithmetic Expression Evaluator in C++	Version: 1.2
Software Development Plan	Date: 29/September/24
01-Project-Plan.doc	

Software Development Plan

1. Introduction

1.1 Purpose

The *Software Development Plan* is a key document that brings together all the important information that is needed to create a software project. This outlines the approach to development and provides a guide for the project manager to lead the team. The team will be able to meet quality standards and keep organized by having this document that sets clear goals, timelines, and responsibilities

- **The project manager** uses it to plan the project schedule and resource needs and 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.
- **Stakeholders/Investors** use it to understand the project scope, timeline, and deliverables from a high-level perspective.
- **Quality Assurance** uses the plan to set up testing protocols and ensure the software meets functional and non-functional requirements.

1.2 Scope

This Software Development Plan outlines the comprehensive strategy for the Arithmetic Expression Evaluator in C++ project, including the deployment of the final product. Specific details for each iteration will be addressed in the respective Iteration Plans. The plans presented in this document are based on the product requirements outlined in the Vision Document.

This scope of the *Software Development Plan* describes the 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*.

1.3 Definitions, Acronyms, and Abbreviations

See the Project Glossary.

1.4 References

Vision: Our vision is to collaboratively develop a robust, efficient, and reliable arithmetic expression parser for language L, where teamwork ensures the precise evaluation of complex expressions with accurate operator precedence and grouping, forming the foundation of a powerful compiler

Supporting plans or documentation: Project Measurements Document, Risk List Document, Configuration Management Plan, EECS 348 Lecture Slides

1.5 Overview

This *Software Development Plan* contains the following information:

Arithmetic Expression Evaluator in C++	Version: 1.2
Software Development Plan	Date: 29/September/24
01-Project-Plan.doc	

Project Overview	—	Provides a detailed description of the project's purpose, objectives, and scope. It also lists the key deliverables expected, such as requirements, design specifications, and a software design document. Assumptions and constraints that may impact the project are also outlined.
Project Organization	—	Describes the structure of the project team, including roles and responsibilities. It explains how the team will interact with external groups and identifies the key contacts and their responsibilities.
Management Process	—	Outlines the project schedule, cost estimates, and resource requirements. It details how the project will be monitored and controlled, including risk management, quality control, reporting, and configuration management.
Annexes — Contains supplementary materials relevant to the project, such as technical standards, process guidelines, and references to other documentation, ensuring a structured approach to software development.		

2. Project Overview

2.1 Project Purpose, Scope, and Objectives

Purpose: Develop a C++ program that is capable of parsing and evaluating arithmetic expressions that include the operators +, -, *, /, % and **, as well as numeric constants. The program should also be able to handle parentheses so there is proper precedence and grouping. This program should have an arithmetic expression with all the requirements listed above and be able to work with a larger compiler.

Deliverables: The deliverables will include a detailed project plan, a requirements document, a design document compatible with the requirements, test cases based on the requirements and design documents which lead to the final parser.

2.2 Assumptions and Constraints

1. Assumptions

- All deadlines given by the instructor will remain as scheduled.
- Team members can code in C++ or will learn to before writing code
- Team members understand all math operators and PEMDAS order used in the arithmetic parser
- Team members have access to all technology necessary to complete project
- Team will communicate and meet via outlined communication channels to work on iterations

2. Constraints

- Due dates for project iterations
- Required deliverables: project plan, requirements and design documents, test plan, program, and user manual
- UPEDU formatting for project plan; certain documentation formats for requirements and design
- C++ for project code

2.3 Project Deliverables

1. Project Plan

Arithmetic Expression Evaluator in C++	Version: 1.2
Software Development Plan	Date: 29/September/24
01-Project-Plan.doc	

- A comprehensive outline of the project’s purpose, goals, schedule, timeline, and communication. It gives a clear guide of the project’s phases, deliverables, and process to achieve implementation. It also gives the reader a guide to navigate the project artifacts and follow the project’s connected parts.
2. Requirements document
 - A document consisting of the project's software requirements. It specifies whether requirements are functional or non-functional. This document clearly defines each need so the team doesn't add unnecessary features, leave out important features, or make a project outside the constraints.
 3. Software design document
 - A document illustrating the design of the calculator. It shows several angles of the project: a use case diagram, a class diagram, a text description of the design, and more. It gives team members and readers a full view of the project’s relationships, use cases, and the design decisions of the project.
 4. Test cases
 - The test cases validate that the product effectively implements each of its requirements. They test every aspect of the calculator to ensure it correctly parses syntax, handles errors, calculates, gives output, and more. This ensures the stakeholder’s needs are fulfilled, and the team is confident in the project handoff.
 5. User manual
 - The user manual gives users a guide in how to set up, provide correct syntax, give input, troubleshoot, and overall, how to use the calculator. This is the documentation for this part of the L language.
 6. C++ program
 - The C++ program is the part of the project that stores the code for the calculator. This program contains libraries, a make file, the calculator code, and more.

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

Version	Phase	Description	Criteria for Unscheduled Revision
1.2	Completion of Project Plan	Worked through sections within the Software development plan to define expectations and goals	N/A

Arithmetic Expression Evaluator in C++	Version: 1.2
Software Development Plan	Date: 29/September/24
01-Project-Plan.doc	

3. Project Organization



3.1 Organizational Structure

Macromanagement of the project will be overseen by the project manager. The Scrum Master will oversee individual sprints and set goals for each sprint the development team works on. The Technical Lead will make decisions concerning the implementation of requirements into code; they will also decide on software tools to be used by the team with their input. The Quality Assurance Lead will oversee testing on finished iterations and is responsible for creating and running test suites. The configuration lead is responsible for managing the Git repository and ensuring all team members are working with the same version of the project; if a version is to be reverted, that is their responsibility. The UI/UX lead will make decisions about the User Interface of the program, whether it be text-based or graphical.




3.2 External Interfaces

3.3 Roles and Responsibilities

Person	Unified Process for EDUcation Role
Addison Bartelli	Technical Lead: Make decisions concerning the implementation of requirements into code; will also decide on software tools to be used by the team with their input.
Peter Barybin	Project Manager: In charge of overseeing the project
Arpa Das	Configuration Lead: Responsible for managing the Git repository and ensuring all team members are working with the same version of the project; if a version is to be reverted, that is their responsibility
Emily Farley	Scrum Master: Oversee individual sprints and set goals for each sprint the development team works on. Will discuss the weaknesses of the project.
Hart Nurnberg	Quality Assurance Lead: Oversee testing on finished iterations and is responsible for creating and running test suites.
All	UI/UX Lead

Person	Discord	Email	Availability	Computing Platform Experience	Programming Knowledge	Photo
Addison Bartelli	addiebart	bartelli@ku.edu	M 11-7 T 11-2, 6-8 W 11-2, 7-8 Th 6-8pm F 1-8	Linux, Windows	Python, JS, Java	
Peter Barybin	phormeddolphin	pbarybin@ku.edu	M: 2-7pm T: 4-7pm W: 3-7pm Th: 6-10pm F: 10am-5pm	Linux, Mac	Python Java ML basics	

Arithmetic Expression Evaluator in C++	Version: 1.2
Software Development Plan	Date: 29/September/24
01-Project-Plan.doc	

Arpa Das	arpa9520	a511d599@ku.edu	M: 3pm -7pm T: 4pm-7pm W: 3pm-7pm Th: 10am-12pm F:10am - 1pm	Linux, Windows	Python OOP	
Emily Farley	emilyfarley	e920f430@ku.edu	M: 10 am-1pm T: 4pm-8pm W: 3pm-7pm Th: 6pm-9pm F: 10 am-2pm	Windows, WSL, Linux	Python OOP	
Hart Nurnberg	hartisthefart	hmurnberg@ku.edu	M: 12-5pm T: 8am-1pm W: 12-5pm Th: 3-7pm F: 2-5pm	Windows, Linux machines, WSL/Ubuntu	Python OOP C++ basics JS basics HTML basics	
Baker Anderson						

Anyone on the project can perform [Any Role](#) activities.

Communication Channels: Team will communicate via a discord server with specific channels for the project. Secondary communication will be done through the KU outlook email. File sharing will be done through a group GitHub repository.

4. Management Process

4.1 Project Estimates

4.2 Project Plan

4.2.1 Phase Plan

4.2.2 Iteration Objectives

Re-implement a working, but decreasingly flawed product based upon the notes from the previous sprint. Run test cases every sprint and adapt test cases where needed. In completing a sprint, team members are encouraged to find weaknesses in the deliverables to create goals for the next sprint. These weaknesses will be discussed in meetings with the Scrum Master. For the first sprint, merely making a fuctional product should be the goal, to be built upon in future sprints.

4.2.3 Releases

N/A as of 9/29/2024

4.2.4 Project Schedule

Limits for milestones will be in line with associated due dates as imposed by the EECS 348 lecture.

Milestone	Limit
Project Plan	09/29/2024, 11:59 PM CST

Arithmetic Expression Evaluator in C++	Version: 1.2
Software Development Plan	Date: 29/September/24
01-Project-Plan.doc	

Requirements	10/20/2024, 11:59 PM CST
Design	To be determined by lecture
Coding / Implementation	To be determined by lecture
Testing	To be determined by lecture
User Manual	To be determined by lecture

4.2.5 Project Resourcing

4.3 Project Monitoring and Control

Requirements Management

To manage changes to product requirements, we will use a version-controlled repository (e.g., Git) for tracking any modifications. All changes will be discussed in team meetings, documented in a change log, and approved by the team lead before being implemented. Requirement changes will be prioritized based on project impact and time constraints

Quality Control

We will implement regular code reviews and walkthroughs during the development process. Testing will occur at key milestones (after each major feature is implemented), focusing on both functionality and edge cases. We will use unit tests to verify the accuracy of calculator functions and peer code inspections to ensure code quality. If issues arise, corrective actions will involve code refactoring or adjustments based on feedback

Risk Management

We will maintain a risk list that will be reviewed weekly, identifying any potential risks such as delays in functionality implementation or integration issues. Risks will be evaluated based on their likelihood and potential impact, with mitigation plans established. For example, to mitigate the risk of missing deadlines, we will allocate buffer time at the end of the project for any unforeseen delays.

Configuration Management

All source code, test scripts, and documentation will be stored in a Git repository, with regular commits made after each development session. Naming conventions for files and functions will follow standard C++ practices, and a backup of the repository will be maintained regularly to safeguard against data loss. We will follow a simple branch management strategy, with a primary branch for stable code and separate feature branches for development. Product artifacts naming convention:

<ProjectName>_<ArtifactType>_<Description>_v<VersionNumber>.<FileExtension>

4.4 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.

4.5 Quality Control

Defects will be recorded and tracked as Change Requests, and defect metrics will be gathered (see Reporting and Measurement below).

Arithmetic Expression Evaluator in C++	Version: 1.2
Software Development Plan	Date: 29/September/24
01-Project-Plan.doc	

All deliverables must 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

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 weekly. 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.

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 follow the UPEDU process.

Arithmetic Expression Evaluator in C++	Version: 1.2
Software Development Plan	Date: 29/September/24
01-Project-Plan.doc	

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

5.1 Project Glossary

Arithmetic Expression Parser: a component in a compiler or interpreter responsible for analyzing and processing arithmetic expressions, typically written in programming languages

Compiler: a software tool that translates code written in a high-level programming language (such as C++, Java, or Python) into machine code (binary instructions) that a computer's processor can understand and execute.

Constraints: limitations, restrictions, or conditions that must be considered during the design, development, and deployment of a software system

Deliverables: refers to the tangible or intangible outputs or products produced and delivered during a project. These deliverables are typically specified as part of the project plan and are used to measure progress and success at various stages of development

Functional Requirements: These are specific behaviors or functions that the software must exhibit. Functional requirements detail what the system should accomplish, outlining the tasks the software will perform for its users

Git: a version control system (VCS), used for tracking changes in files and coordinating work among multiple people or teams. It allows developers to maintain a complete history of changes to their codebase, collaborate on projects, and manage versions of their work efficiently

GitHub: GitHub.com is a website server that hosts git repositories

Implementation: the process of converting a design or plan into a functioning software system or feature. It involves writing and developing the actual code that fulfills the requirements, specifications, and design laid out during earlier stages of the software development lifecycle (SDLC)

Iteration: the process of repeatedly executing a set of instructions or refining a process until a desired result is achieved

Language L: a placeholder name being used for the specific programming language we are working with

Makefile: a special file used by the make build automation tool to manage the compilation and linking of programs in software engineering. It defines a set of rules and dependencies for building a project, allowing developers to automate the build process, streamline compilation, and manage project files efficiently.

Arithmetic Expression Evaluator in C++	Version: 1.2
Software Development Plan	Date: 29/September/24
01-Project-Plan.doc	

Nonfunctional Requirements (NFRs): refers to the criteria that specify how a system should behave and impose constraints on its functionality. Unlike functional requirements, which define what a system should do (i.e., the specific behaviors and features), non-functional requirements focus on the quality attributes of the system, including its performance, usability, reliability, and security.

PEMDAS: an acronym that stands for parenthesis, exponents, multiplication, division, addition, and subtraction. Used to remember the order of operations.

Repository: a central location where data, files, or source code for a software project are stored and managed. In software engineering, repositories are often associated with version control systems (VCS) like Git, where developers can commit changes, track versions, and collaborate on the same codebase

RUP Guidelines: Also known as Rational Unified Process Guidelines, refer to a structured framework for software development that emphasizes iterative and incremental development. RUP is characterized by its focus on best practices, risk management, and a well-defined process that encompasses various phases, including inception, elaboration, construction, and transition. The guidelines provide a roadmap for project teams to effectively plan, execute, and monitor software projects while ensuring quality and stakeholder satisfaction.

Scope: defines the boundaries and extent of a software project, detailing what will be included and excluded from the project. It outlines the functionalities, features, and requirements that the software must fulfill, serving as a guideline for project planning and execution.

Software Development Plan: a comprehensive document that outlines the strategy, processes, resources, timelines, and milestones for a software project. It serves as a roadmap for the development team and stakeholders, detailing how the project will be executed, monitored, and controlled.

Sprints: time-boxed iterations in Agile software development, typically lasting between one to four weeks, during which a specific set of features or user stories is developed and delivered. Each sprint involves planning, execution, testing, and review, allowing teams to incrementally build and refine the product. Sprints promote collaboration, flexibility, and rapid feedback, enabling teams to respond quickly to changing requirements and priorities.

Stakeholder: any individual or group that has an interest in or is affected by a software project. Stakeholders can include project sponsors, users, customers, developers, testers, and other parties involved in the project.

UI/UX: UI (User Interface) and UX (User Experience) refer to the design and overall experience of users interacting with a software application.

Arithmetic Expression Evaluator in C++	Version: 1.2
Software Development Plan	Date: 29/September/24
01-Project-Plan.doc	

UPEDU: an acronym for Unified Process for Education, refers to an adaptation of the Unified Process (UP) framework specifically tailored for educational environments. UPEDU emphasizes the iterative and incremental development of educational software and resources, focusing on improving learning outcomes through technology. It integrates best practices from software engineering with pedagogical principles, providing a structured approach to the design, development, and deployment of educational software.

Use Case: a detailed description of a specific interaction between a user (or actor) and a software system that outlines the steps necessary to achieve a particular goal. Use cases are a critical part of requirements gathering and system design, helping to clarify user needs and system functionalities.