Group Without a Name

Arithmetic Expression Evaluator
Software Development Plan
Version <1.0>

Revision History

Date	Version	Description	Author
<09/17/2023>	<1.0>	Base iteration of document	Jake Bernard

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....5

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....5

3. Project Organization....5

3.1 Organizational Structure....5

3.2 External Interfaces....6

3.3 Roles and Responsibilities....6

4. Management Process....6

4.1 Project Estimates....6

4.2 Project Plan....6

4.3 Project Monitoring and Control....7

4.4 Requirements Management....7

4.5 Quality Control....7

4.6 Reporting and Measurement....7

4.7 Risk Management....8

4.8 Configuration Management....8

5. Annexes....8

Software Development Plan

1. Introduction

1.1 Purpose

This *Software Development Plan* exists to serve as a repository for all relevant information used to guide the management of the project and how the software should be developed.

The *Software Development Plan* is used by the p**roject manager** to plan, guide, and assess the project according to a schedule developed from the contents herein, and is used by the **project team members** to understand their tasks, deadlines, and dependencies.

1.2 Scope

The *Software Development Plan* is a broad overview of the plan for the development process of the Arithmetic Expression Evaluator. The planned development cycle is written in the "Iteration Objectives" section.

The requirements this plan is made to address come from the provided *Project Description* document and the *Project Vision* document.

1.3 Definitions, Acronyms, and Abbreviations

See the *Project Glossary*.

1.4 References

The *Software Development Plan* references the following documents:

- *Project Vision,* artifacts/project_vision.odt, Group Without a Name, 2023
- *Project Description,* artifacts/project_description.pdf, University of Kansas, 2023
- *Project Roles*, team_resources/roles.md, Group Without a Name, 2023
- *Project Glossary,* artifacts/project_glossary.odt, Group Without a Name, 2023
- *Project Tasks*, team_resources/tasks/*, Group Without a Name, 2023
- *Google C++ Style Guide*, https://google.github.io/styleguide/cppguide.html, Alphabet Inc., 2023

1.5 Overview

The following sections constitute the outline of the *Softwared Development Plan*:

Project Overview — Defines the project's scope, purpose, and objectives. The deliverables for the project are listed within.

Project Organization — Defines the organization and structure of the project team, along with listing assigned roles.

Management Process — Details the project's milestones and major phases, the estimated schedule, and the plan for monitoring progress on the project.

Applicable Plans and Guidelines — Contains the guidelines which the software development process will adhere to along with the methods, tools, and techniques which will be used.

2. Project Overview

2.1 Project Purpose, Scope, and Objectives

The purpose of this project is to create a program in C++ which acts as an arithmetic expression parser and evaluator. The program should be capable of parsing and evaluating user input containing multiple operands, numerical constants, parenthetical grouping, and the operators +, -, *, /, %, and ^ (or **) while adhering to the PEMDAS order of operations, with the possiblility of extra funcitonality being added depending upon time constraints.

In addition to this, it is necessary for the project to come with several required deliverables which inform, guide, and integrate the development and use of the program. These deliverables are a project plan, requirements document, design document, test cases, a user manual, and the source code, which should be well documented.

2.2 Assumptions and Constraints

This plan assumes all members will be available at some point outside of their class schedules to contribute ot the project, are capable of following through with their roles, and are willing and able to contribute. The plan will be constrained by team member's availability outside of class, deadlines for each deliverable, and the relative abilities and competencies of the team members to accomplish their tasks on time and with satisfactory quality.

2.3 Project Deliverables

The project deliverables are as follows:

- Project Management Plan, due 9/24/2023
- Requirements Document, due TBD
- Design Document, due TBD
- Test Cases, due TBD
- Source Code and Compiled Program, due TBD
- User Manual, due TBD
- Activity Logs and Misc Artifacts, due TBD

2.4 Evolution of the Software Development Plan

The *Software Development Plan* will be updated and revised as necessary throughout iterations, as requirements for deliverables are released, and as the project comes closer to realization.

3. Project Organization

3.1 Organizational Structure

See the *Project Roles* file for detailed roles and descriptions.

The following activities will be expected of team members and should be carried out on a weekly basis:

Weekly meetings -- To review and organize what has and needs to be done.

Activity-logging -- Each team member should keep a log file for activities undertaken and approximate schedules of when work was done.

Progress Reports -- Each week members should log progress for specific modules, which will be reviewed and presented at the weekly meeting by the Team Resources Manager.

Code Review	--	When not covered by the Software Quality Analyst, members should review code written by other members to make sure it adheres to style guidelines (from the Google C++ Style Guide), is properly documented, and performant. This is especially important for members working on the same module.

In addition to these responsibilities, each member may be appointed responsibility for specific modules of code that they are assigned to write or different artifacts which they should create, which will be determined as the project develops.

3.2 External Interfaces

N/A.

3.3 Roles and Responsibilities

Person	Role
Jake Bernard	Project manager
Nick Reinig	Software Quality Analyst
Drew Meyer	Team Resources Manager
Vidur Pandiripally	Scrum Master
Charlie Gillund	Configuration Manager

For a description of all roles, see the *Project Roles* file.

Tasks which are not role-specific are marked as **Any Member** tasks and may be performed by any member.

4. Management Process

4.1 Project Estimates

N/A.

4.2 Project Plan

4.2.1 Phase Plan

N/A.

4.2.2 Iteration Objectives

The current iteration objectives are as follows:

- **Iteration 0**
  - The requirements, goals, and vision for the project is finalized.
- **Iteration 1**
  - Base functionality of data structures. Basic bug-testing systems implemented.

- **Iteration 2**
  - Parsing functionality is added. Numerical constants and ** synonym for ^ is added. Strings are parsed into underlying data structures. Bug testing methods for parsing implemented.
- **Iteration 3**
  - Required operations are added along with evaluation of expressions.
- **Iteration 4**
  - GUI finalized.
  - More robust error handling, along with positional recognition of syntax errors in input.
- **Iteration 5 (Optional)**
  - Functionality to change answers to be fractional/involve symbolic expressions. Support for arbitrary precision (bignum) arithmetic.
- **Iteration 6 (Optional)**
  - GUI enhancements.
- **Iteration 7 (Optional)**
  - Support for additional operations such as logarithms, roots, ceiling, floor, etc.
- **Iteration 8 (Optional)**
  - Realtime, progressive display of results as enters input
- **Iteration 9 (Optional)**
  - Support for calculation history, UX flourishes, easter eggs.

4.2.3 Releases

The planned software releases are as follows:

- Closed releases (internal milestones)
  - Pre-Alpha
    - Iteration 0
  - Alpha
    - Iteration 1-3
  - Beta
    - Iteration 3+
  - Release Candidate
    - Testing phase -- repeated if necessary for Iterations above 3
- Open releases
  - Stable Release
    - After testing phase/end of project

4.2.4 Project Schedule

- Project Management Plan

- 9/24/2023
- Complete idea, agreement upon project plan
  - 9/27/2024
- Requirements Document
  - TBD
- Design Document
  - TBD
- Iteration 0
  - TBD
- Iteration 1
  - TBD
- Iteration 2
  - TBD
- Iteration 3
  - TBD
- Testing Phase
  - TBD
- User Manual
  - TBD
- Final Submission
  - TBD
- Activity Logs and Misc Artifacts
  - Updated Weekly

4.2.5 Project Resourcing

N/A.

4.3 Project Monitoring and Control

Most monitoring of the project will happen during weekly meetings or through online channels by which members may communicate to each other outside of meetings, e.g. a Discord server. Some log files will be made for tracking specific progress/project needs.

Some notable places where monitoring will be necessary are:

- Requirements Management: Changes to product requirements will be tracked in the version histories of artifacts uploaded to the project GitHub repo. Various task files for specific tasks/modules will be created in the *Project Tasks* to track requirements and progress in those areas, with an overview of all tasks also available. The project Jira will be used to assign specific tasks to members.
- Quality Control: Weekly code and artifact reviews along with meetings will be held to ensure that all submitted artifacts and code is up to project standards and without defects. The *Software Quality Analyst* will be primarily responsible for reviewing the codebase, with the other members playing lesser roles when necessary.
- Risk Management: Potential risks and outstanding problems will be discussed during weekly meetings.
- Configuration Management: Changes and problems will be submitted to the project Jira as tickets which are to be reviewed by the Congifuration Manager and brought up in meetings.

4.4 Requirements Management

N/A.

4.5 Quality Control

The Software Quality Analyst will review all new code prior to each weekly meeting and will devise test cases for modules to ensure stability and performance. Defects/problems will be submitted to the project Jira as tickets and will be addressed in meetings.

All deliverables will be reviewed by the Project Manager using quality guidelines and checklists to maintain appropriate standards for the project.

4.6 Reporting and Measurement

N/A.

4.7 Risk Management

Weekly meetings will assess possible risks to the project and attempts will be made to mitigate risks based upon planning ahead for worst case scenarios and consistently making alternative plans of action in case members may not be able to complete assignments. Appropriate version control systems will be used to make sure no files are permanently lost or made unusable. Risks will be prioritized by likelihood and severity, and regular check-ins should ensure that regular communication regarding risks is established.

4.8 Configuration Management

Changes and problems will be reviewed by the Configuration Manager. Weekly meetings will mention these changes to all member to ensure common knowledge and approval is received.

Artifacts are to be given simple filenames with no capital letters and underscores between words. Version numbers increase by 0.1 unless a new iteration is reached or a change is deemed signficant enough as to merit a full version change by the team.

A general "dev" branch for development of features/modules will be used in common among members, with potential experimental branches being made when necesary. Dev branches will be pushed to "test" branches once they are capable of running, and test branches will mainly exist for debugging. Test branches will only be pushed to main once a feature complete release is ready.

All artifacts and activity logs will be pushed to main. Logs will be put in appropriate folders, along with artifacts. All code will adhere to the Google C++ Style Guide. Version control will be used to ensure losses of data are minimal.

5. Annexes

The project will follow a modified version of the UPEDU process, and most team-related documents will use Markdown files which will be within the GitHub repository, with templates being provided to members. Certain elements of Scrum will be incorporated into the development process, including sprints. The project will use Jira to organize work and tasks for implementing different modules. UML diagrams will be used for several artifacts. All code will follow the Google C++ Style Guide.