08

**Fall**

**GP GENIE**

SEIS 610 Final Project Report: GP Genie

Ujin Han Ι Susan Mairs Ι Li Wang Ι Justin Florkiah Ι Roger Peterson Ι Felista Mpanga

Table of Contents

[Project Plan 4](#_Toc374770477)

[Problem Description 4](#_Toc374770478)

[Problem Title 4](#_Toc374770479)

[Problem Summary 4](#_Toc374770480)

[Detailed description 4](#_Toc374770481)

[Approach 5](#_Toc374770482)

[Requirements Analysis And Preliminary System Design 5](#_Toc374770483)

[Weekly Software Configuration Management Files And Traceability 5](#_Toc374770484)

[Work Plan 6](#_Toc374770485)

[Roles & Responsibilities 6](#_Toc374770486)

[Milestones & Schedule 7](#_Toc374770487)

[Final System Design 8](#_Toc374770488)

[System Testing Plan and Test Results 9](#_Toc374770489)

[System Testing Plan 9](#_Toc374770490)

[Test Results 10](#_Toc374770491)

[Lessons-Learned 10](#_Toc374770492)

[What Worked 10](#_Toc374770493)

[Strong Lead in Each Section 10](#_Toc374770494)

[Weekly Team Meetings 11](#_Toc374770495)

[Team Review 11](#_Toc374770496)

[Building Our Own Code 11](#_Toc374770497)

[What Didn’t Work 11](#_Toc374770498)

[Time for Tool and Test Environment Set Up 11](#_Toc374770499)

[Delay in Parallel Testing 11](#_Toc374770500)

[Short User Manual 11](#_Toc374770501)

[Description 11](#_Toc374770502)

[Instructions 11](#_Toc374770503)

[Project Summary 12](#_Toc374770504)

[Post-Project Analysis 13](#_Toc374770505)

[Review of the Project Objective 13](#_Toc374770506)

[Team-Analysis 13](#_Toc374770507)

[Revisiting the Original Project Plan 13](#_Toc374770508)

[Analyzing What Happened During The Project 13](#_Toc374770509)

[Review the Project Outcomes 13](#_Toc374770510)

[Appendix A: Project Related Materials 14](#_Toc374770511)

[A1. Weekly SCM files/folders (in plain text file) 14](#_Toc374770512)

[A2. Visual Materials (PPT and printed PPT) 14](#_Toc374770513)

[A3. Post-project Analyses (in plain text file) 14](#_Toc374770514)

# Project Plan

## Problem Description

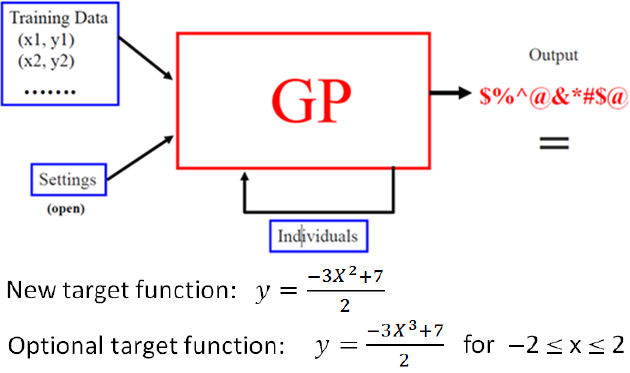
### Problem Title

SEIS 610 – Generic Programming (GP) Project

### Problem Summary

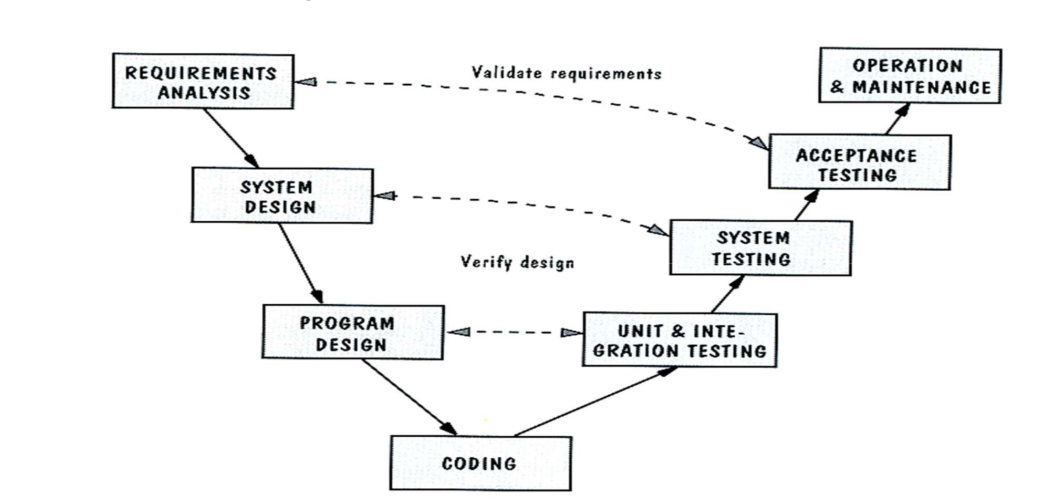
Generating a function that is equivalent to the target function

### Detailed description



* Generate a set of training set to evaluate the fitness. A set of random input (x) will be generated. Evaluate the input using the target function, to provide the training set.
* Generate a set of initial inputs of random compositions of the function of the problem.
* Execute each program in the population and assign it a fitness value according to how well it solves the problem. The fitness can be calculated as the delta (sum of error) of the function and the target function. Higher delta leads to a lower fitness.
* If there is a function that meets the termination condition (delta =0), program stops.
* If termination condition does not meet, the following steps will be executed
  + **Natural Selection**: The solutions with good fitness value survive to the next generation with high probability. Solutions will lower fitness value perish with high probability.
  + **Crossover**: Two solutions chosen with natural selection to generate new solutions
  + **Mutation**: replacing part of the solution with random generated new solution to reproduce new solutions
  + Repeat the fitness and selection step until termination condition meets, or time is over

### Approach

The project will follow the “V-model” software development process.

The client requirement should be documented and captured during the development

## Requirements Analysis And Preliminary System Design

The team created a Software Requirements Specification (SRS) document to capture requirements, requirements analysis in model diagrams and a data dictionary, and the preliminary design in a class diagram. A hard copy of this SRS document “SRS GP Genie.docx“ is attached and submitted with this final report.

## Weekly Software Configuration Management Files And Traceability

The team initially embraced a quasi-form of SCM by utilizing Google Docs as a central repository for the various team related project and design files. This worked well for the initial creation of documents such as the SRS, project plan and requirements traceability matrix. However, as the inertia of project documentation unfolded, it was apparently clear that the absence of version control would quickly become a handicap.

In earnest, the team began to explore various options for a reliable form of SCM. In the essence of time the team decided to quickly explore the basic merits between GitHub and WinCVS. Our selection criteria was based on overall usability in that the SCM tool with the shortest learning curve and accessibility would become the premier solution of choice. In the end GitHub was selected by the team not only for the GitHub elegance in version control within a central repository, but the seamless desktop cloning and change management synchronization ability as well. Overall, SCM integration was a very successful endeavor for the GP-Genie team and was decidedly so as the team was very fortunate to have a team member who took it upon herself to become a champion and super user of GitHub functionality so that the entire team gracefully embraced the version control tool.

## Work Plan

### Roles & Responsibilities

The following is a list of roles and who is responsible for each role.

|  |  |
| --- | --- |
| **Roles** | **Name** |
| Project Management & Communications Liaison | Ujin Han |
| Process Leaders | Ujin Han  Justin Florkiah |
| Development Leader | Li Wang |
| Technical Leaders | Roger Peterson  Susan Mairs |
| Measurement Leaders | Roger Peterson  Ujin Han |
| Capture Requirements | Susan Mairs |
| Coding | Li Wang |
| Code Review | Susan Mairs |
| Testing |  |
| User Testing | Felista Mpanga  Justin Florkiah |
| Functional Testing | Susan Mairs  Li Wang |
| Verification Leader | Roger Peterson |

For the roles with multiple team members listed, they will share the responsibilities of that particular section of the project. As the project goes on and more responsibilities come up, those roles will be added. During the project, if a certain team member is struggling with the given task, others with knowledge and availability will assist in its completion.

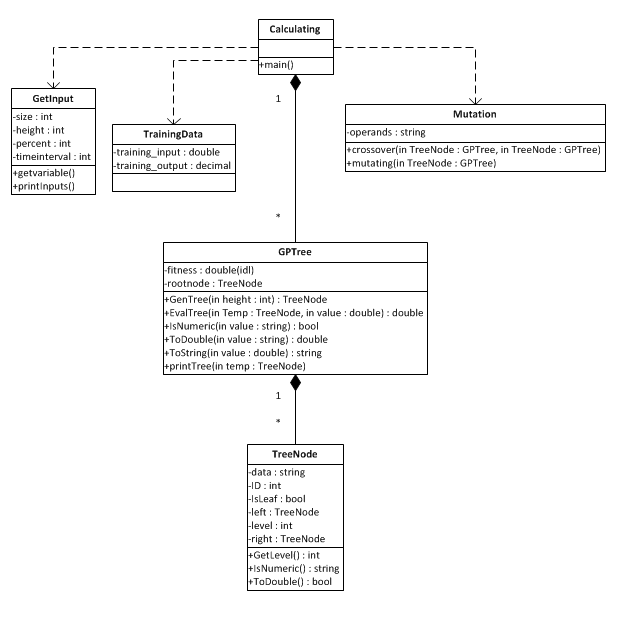
### Milestones & Schedule

To complete the project with changes in requirements by December 14th, the team has following milestones and dates set.

|  |  |
| --- | --- |
| **Milestone** | **Due Date** |
| Requirements & preliminary system design finalized; Project Plan completed and submitted | October 26, 2013 |
| All complements needed for program design finalized | October 29, 2013 |
| Working software | November 19, 2013 |
| Testing software | November 26, 2013 |
| Revised working software with new requirements | December 7, 2013 |
| PowerPoint presentation completed | December 12, 2013 |
| Project completion and deliverables submitted | December 14, 2013 |

The team has a weekly meeting on Tuesday evenings at 6 pm until the end of the semester to check-in and discusses the progress of the project. These meetings will last from an hour to two hours depending on the subject of the meeting at the time. Additional meetings will be scheduled as the team sees fit.

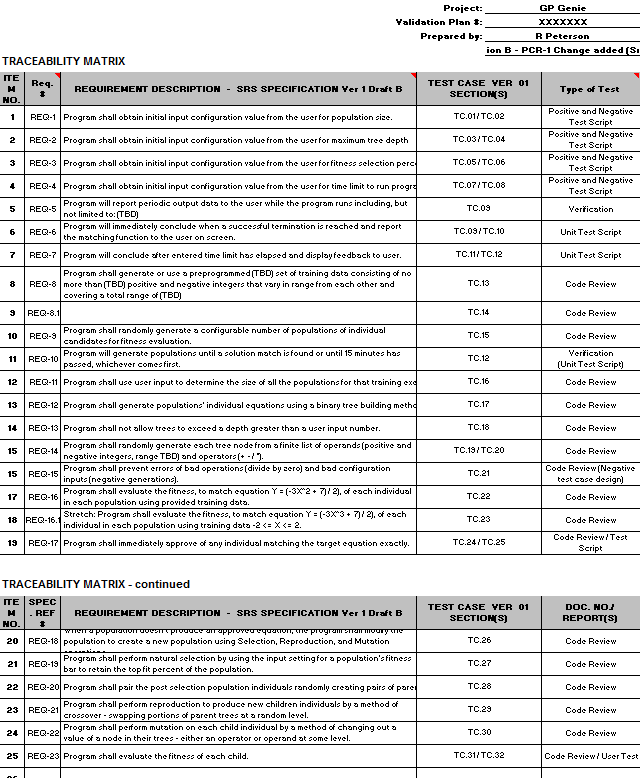
# Final System Design



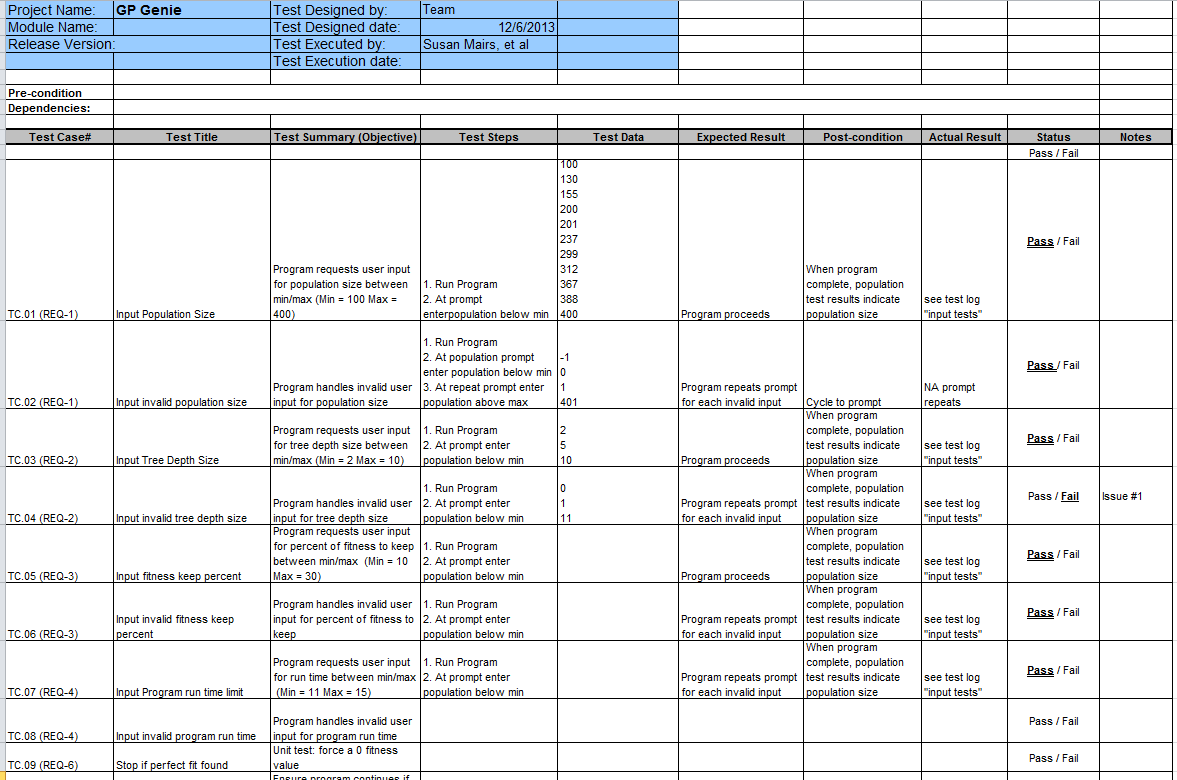
# System Testing Plan and Test Results

## System Testing Plan

The team used a traceability matrix to ensure that all requirements would be covered in testing, either by user testing of code verification steps.



This pointed to the Test Case Design Document for the project which contained the detailed plan for testing each requirement.



## Test Results

Test results were captured in spreadsheets and their needed action was captured in the GitHub tool as Issues would could be labels is bugs or notes and then assigned, tracked, then cleared. These files are stored in GitHub in a TestResults section for the team to continue to use for testing and project analysis.

# Lessons-Learned

## What Worked

### Strong Lead in Each Section

Having a lead who had a strong understanding and background in each stage of the project helped in (1) guiding that section of the project and explaining the objective and tasks associated; (2) improving efficiency of completing the tasks associated with that stage; and (3) having a designated “expert” to refer to. Also, the diversity in technical experience within the team allowed team members to learn from each other.

### Weekly Team Meetings

The GP Genie team conducted weekly team meetings throughout the semester starting in October, during which the check-in and team review of the updated materials were performed. This helped in keeping members accountable for their assigned tasks and the project moving forward. These meetings were especially useful in troubleshooting any unexpected issues, addressing concerns, and/or adjusting a course of action to improve performance or efficiency of the project.

### Team Review

During the meetings, major portion was given to team review of the materials developed so far. This helped in (1) getting all members on the same level of understanding on what needs to be done in that particular stage and the project overall; (2) getting multiple perspectives for optimization and problem-solving; and (3) double-checking work as it is being done to make sure that all requirements are met.

### Building Our Own Code

Developing our own code gave the team a better understanding of requirements for code validation and testing.

## What Didn’t Work

### Time for Tool and Test Environment Set Up

There was a significant time that took for every team member to get set up and become familiar with GitHub, JDK, and Eclipse. This delayed communications and updates on documents, weekly review of the code, module testing and transparency in the progress of development.

### Delay in Parallel Testing

Due to insufficient set up of development tools and code sharing, granular code testing, module testing and test scripts were not able to be done in parallel to writing the code, which would have improved the efficiency during the coding and testing stages of the project.

# Short User Manual

## Description

GP Genie is a generic programming software that aims to generate an equation equivalent to the target function of the user’s choice, either A or B. The user inputs the following basic criteria: (1) Population Size, (2) Maximum Tree Depth, (3) Fitness Selection Percent, (4) Time Limit, and (5) Program: Required Function or the Optional Target Function.

## Instructions

1. Open GP Genie.
2. Choose and input population size, between 100 and 400.
3. Input maximum tree depth, between 2 and 10.
4. Input fitness selection percentage, between 10% and 30%.
5. Input time limit, between 1 to 15 minutes.
6. Choose which program to run, either Required Function or the Optional Target Function.
   1. Target Function A: Y =
   2. Optional Function B: Y = for x -2 ≤ X ≤ 2
7. The program is designed to choose the training data associated with the equation, according to the user’s choice.
8. Within the maximum time limit, the program will provide the fittest equation and its fitness value.

# Project Summary

The objective of this project was to create a generic program that would generate an equation that would be as close to the target function as possible, which is measured through the fitness value. GP Genie developed its own java code to generate trees that would create different equations, cross over, mutate, and then would narrow the results depending on the fitness value. Testing was performed on the developed code to make sure there were no bugs, and to refine the test results to determine which inputs generated the equation with the smallest fitness value.

By using the v-model for the project process, conducting weekly check-in meetings and using GitHub for sharing files and code, GP Genie developed a program that successfully created an equation that met the requirements.

# Post-Project Analysis

## Review of the Project Objective

The objective of the project was to find an equation that is closest to the target function given, , or the optional function, through generic programming. GP Genie developed its own code that the user can choose which of the two functions to aim, given user inputs on population size, maximum tree depth, fitness selection percentage, and time limit.

## Team-Analysis

As a team, we quickly realized that the diverse nature of the project required an array of skills to be successful. This required each member to leverage his/her experience and educational back ground to add value to our final deliverable. This also provided a learning opportunity for each member in sharing our ideas and expertise. This is something we all valued and embraced throughout the process.

In realizing this, the project manager was able to more effectively allocate responsibilities. Tasks were assigned to each member/partner in a manner that would enable them to leverage their expertise while challenging them to explore new frontiers and learning opportunities. This was especially beneficial to experienced engineers in helping with the coding process, while challenging them to explore more effective communication avenues. Similarly, more grounded communication students were assigned tasks that would permit them to improve upon their coding and software development skills.

With this, the team was able to more effectively research and develop software programs that were value added. But more importantly, it provided the opportunity to further explore the team’s strength and weaknesses. In all, all team members improved upon their skills while discovering and augmenting upon their respective weaknesses.

## Revisiting the Original Project Plan

With the new target function and optional functional given towards the end of the project in late-November, the team revisited the original project plan and adjusted the requirements analysis, system design, and the code accordingly.

## Analyzing What Happened During The Project

The team conducted weekly meetings for check-in and progress review. During code development and SCM changes to GitHub, the team struggled to get adjusted to the new environment of tools and share code/files, which in turn, delayed the testing. However, the team was successful in having a solid plan in place early accounting for requirements change, keeping track of task progress, and leveraging each other’s strengths to complete the project.

## Review the Project Outcomes

The GP Genie team delivered a generic program that met all the requirements set out by the client, and was able to develop a program that generated an equation closest to the selected target function (either the target function or the optional target function).

# Appendix A: Project Related Materials

## A1. Weekly SCM files/folders (in plain text file)

## A2. Visual Materials (PPT and printed PPT)

## A3. Post-project Analyses (in plain text file)