Software Requirements Specification

for

GP Genie

Version 1.0 Draft B

Prepared by Susan Mairs

University of St. Thomas

SEIS 610 Fall 2013 SatB

December 7, 2013

Table of Contents

Table of Contents 1

Revision History 1

1. Introduction 2

1.1 Purpose 2

1.2 Document Conventions 2

1.3 Intended Audience and Reading Suggestions 2

1.4 Product Scope 2

1.5 References 2

2. Overall Description 3

2.1 Product Perspective 3

2.2 Product Functions 3

2.3 User Classes and Characteristics 4

2.4 Operating Environment 4

2.5 Design and Implementation Constraints 4

2.6 User Documentation 4

2.7 Assumptions and Dependencies 4

3. External Interface Requirements 5

3.1 User Interfaces 5

3.2 Hardware Interfaces 5

3.3 Software Interfaces 5

3.4 Communications Interfaces 5

4. System Features 5

4.1 User Interface 5

4.2 Training Data 6

4.3 Population 6

4.4 Individual Equation Trees 7

4.5 Fitness Evaluation and Selection 7

4.6 Crossover and Mutation 8

5. Other Nonfunctional Requirements 8

5.1 Performance Requirements 8

5.2 Safety Requirements 8

5.3 Security Requirements 8

5.4 Software Quality Attributes 8

5.5 Business Rules 9

6. Other Requirements 9

Appendix A: Data Dictionary 10

Appendix B: Analysis Models 12

Appendix C: To Be Determined List 15

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
| Susan Mairs | 12/7/2013 | PCR-1 – added equation change and stretch scope for Nov 27 requirement change. | 1.0 B |
| Susan Mairs | 10/25/2013 | Initial Draft from Notes | 1.0 A |

# Introduction

## Purpose

This SRS document describes the program “GP Genie”, a Genetic Programming software program that will be built as a class assignment. This is version 1.0 of the program with no expected future revisions. This document covers requirements for the full scope of the program assignment.

## Document Conventions

Not applicable for this draft.

## Intended Audience and Reading Suggestions

Not applicable for this class assignment.

## Product Scope

The GP Genie program will attempt to create and locate an individual equation that closely matches the equation (-3X^2 + 7) / 2) by generating a population of random binary tree equations and testing their fitness against the target equation using a set of training inputs and comparing the results. The Program will proceed to select the fittest individuals from the general population and perform a parent/child crossover operation on them where elements of their trees are swapped with each other. Finally, the program will further modify the children results with a small mutation operation where one node is swapped with another value of the same type. The fitness of the children will be evaluated against the training data in hopes that the program will have created match using the selection, crossover, and mutation operations.

Initial settings will be given to the program which limit the size of the population(s), the depth of the individual trees, and the level of acceptable fitness for the selection phase. Other parameters will be programmed in, such as the possible values of the tree nodes, the time limit that the program has to run, etc.

Stretch Scope: As time allows, the program should also attempt to create and locate an individual equation that closely matches the equation (-3X^3 + 7) / 2) using training data -2 <= X <= 2.

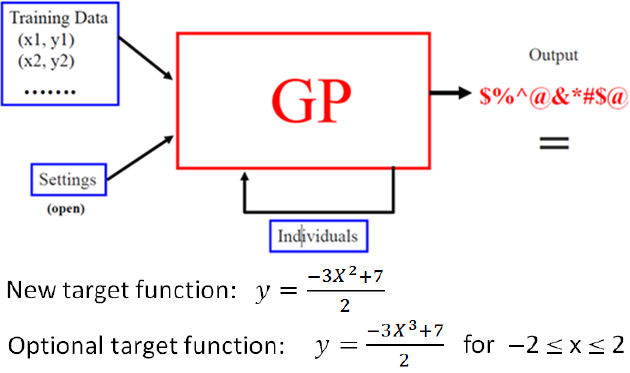
## References

No other documents are referenced by this draft. The diagrams in the appendices can be viewed without access to the source program files used to create them.

# Overall Description

## Product Perspective

The product will meet the requirements given by Professor Lai who describes the overview with this diagram:



## Product Functions

### Program General Flow

Refer to [Figure 1 - Flowchart](#Fig1) in Appendix B.

The program will perform the following major functions:

* Generate a population of equation individuals
* Analyze the fitness of each of the individuals against the target equation
* Select the fittest individuals in the population to perform the following operations
  + Crossover pairs of individuals by swapping portions of their trees
  + Mutate the children individuals by changing one of their node values
  + Analyze the fitness of the children against the target equation
* Reporting on the nearest fit found after fifteen minutes

### Data Flow

Refer to [Figure 2 – Data Flow Diagram](#Fig2) in Appendix B.

The Program will use data stores for the populations and data to represent the individuals in their various states as they are created and modified by the program. Key data elements include

* Population size
* Training data Xs and Ys
* Individual trees and their depths
* Tree node operators and operands
* Fitness criteria for selection
* Crossover Nodes
* Mutation Nodes

## User Classes and Characteristics

Refer to [Figure 3 – Class Diagram](#Fig2) in Appendix B.

The initial classes identified as a high level system design include:

* Training Data
* Population
* Individual
* Node (Operand and Operator types)
* PostSelection Population

## Operating Environment

The program will run on a Windows 7 or 8 machine. Java will be the language the program is written in. It is expected that the program will not have a graphical user interface and will run from a command prompt, eliciting inputs needed from the user at the start of the program.

## Design and Implementation Constraints

<Describe any items or issues that will limit the options available to the developers. These might include: corporate or regulatory policies; hardware limitations (timing requirements, memory requirements); interfaces to other applications; specific technologies, tools, and databases to be used; parallel operations; language requirements; communications protocols; security considerations; design conventions or programming standards (for example, if the customer’s organization will be responsible for maintaining the delivered software).>

### Time Limit

The program must be run and conclude in a maximum time of fifteen minutes.

### Others TBD

## User Documentation

None expected for the class assignment.

## Assumptions and Dependencies

None identified at the time of this draft.

# External Interface Requirements

## User Interfaces

The program will be started and run using the operating system’s command prompt. When the program is run, the user will be prompted to enter input numbers for the population size, maximum tree depth, selection size, and optionally the time constraint.

## Hardware Interfaces

Not applicable for this class assignment.

## Software Interfaces

Not applicable for this class assignment.

## Communications Interfaces

Not applicable for this class assignment.

# System Features

## User Interface

### Description

This is the data required at the start of the program intended for input by the program operator.

### Stimulus/Response Sequences

Welcome to GP Genie.

Input Population Size (n to NN) >

Error if necessary

Correct: respond with next question

Input Maximum Tree Depth (n to NN) >

Error if necessary

Correct: respond with next question

Input Fitness Selection Percent – top items to survive (n to NN) >

Error if necessary

Correct: respond with next question

Input Time Limit in minutes (n to NN) >

Error if necessary

Correct: respond with next question

Program working to find match…

Program complete. Results:

(TBD)

### Functional Requirements

REQ-1: Program shall obtain initial input configuration value from the user for population size.

REQ-2: Program shall obtain initial input configuration value from the user for maximum tree depth.

REQ-3: Program shall obtain initial input configuration value from the user for fitness selection percent to keep.

REQ-4: Program shall obtain initial input configuration value from the user for time limit to run program.

REQ-5: Program will report periodic output data to the user while the program runs including, but not limited to: (TBD)

REQ-6: Program will immediately conclude when a successful termination is reached and report the matching function to the user on screen.

REQ-7: Program will conclude after entered time limit has elapsed and display feedback to user.

## Training Data

### Description

Training data include the target equation, the collection of the X inputs and their Y output results. This set may be generated randomly or hard-coded after a good set is identified. It is not expected that the training data be input by the operator at the outset of the program.

### Stimulus/Response Sequences

TBD

### Functional Requirements

REQ-8: Program shall generate or use a preprogrammed (TBD) set of training data consisting of no more than (TBD) positive and negative integers that vary in range from each other and covering a total range of (TBD).

## Population

### Description

The program will randomly generate a set of individual equations to be evaluated against the training data for fitness to the target equation.

### Stimulus/Response Sequences

TBD

### Functional Requirements

REQ-9: Program shall randomly generate a configurable number of populations of individual candidates for fitness evaluation.

REQ-10: Program will generate populations until a solution match is found or until 15 minutes has passed, whichever comes first.

REG-11: Program shall use user input to determine the size of all the populations for that training execution.

## Individual Equation Trees

### Description

The individuals generated for the population will be binary trees of a depth input at the start of the program. The nodes will be operators which take either operators or operands for their next level. These will be generated randomly operators include + - \* and / while operands include the variable X and integers whose range is TBD.

### Stimulus/Response Sequences

TBD

### Functional Requirements

REQ-12: Program shall generate populations’ individual equations using a binary tree building method.

REQ-13: Program shall not allow trees to exceed a depth greater than a user input number.

REQ-14: Program shall randomly generate each tree node from a finite list of operands (positive and negative integers, range TBD) and operators (+ - / \*).

REQ-15: Program shall prevent errors of bad operations (divide by zero) and bad configuration inputs (negative generations).

## Fitness Evaluation and Selection

### Description

Each tree Individual will be evaluated for closeness to match with the target equation. This is figured by a method of running the training data Xs through the equation and evaluating the closeness of its Y result to the Y result of the same X in the training data. This delta is added for all training pairs and the sum compared to the sum of the Ys in the training data. When all equations have been analyzed, those with the closest fit, as determined by the percent of fit items entered at the start of the program, are kept for further processing. If a match is found (Fitness Value = 0) the program is stopped.

### Stimulus/Response Sequences

TBD

### Functional Requirements

REQ-16: Program shall evaluate the fitness, to match equation Y = (-3X^2 + 7) / 2), of each individual in each population using provided training data.

REQ-16.1 Stretch: Program shall evaluate the fitness, to match equation Y = (-3X^3 + 7) / 2), of each individual in each population using training data -2 <= X <= 2.

REQ-17: Program shall immediately approve of any individual matching the target equation exactly.

REQ-18: When a population doesn’t produce an approved equation, the program shall modify the population to create a new population using Selection, Reproduction, and Mutation operations.

REQ-19: Program shall perform natural selection by using the input setting for a population’s fitness bar to retain the top fit percent of the population.

## Crossover and Mutation

### Description

The program will create children form pairs of fit parents and modify the children individuals with a small mutation operation where one node is swapped with another value of the same type. The fitness of the children will be evaluated against the training data in hopes that the program will have created match using the selection, crossover, and mutation operations.

### Stimulus/Response Sequences

### Functional Requirements

REQ-20: Program shall pair the postselection population individuals randomly creating pairs of parents.

REQ-21: Program shall perform reproduction to produce new children individuals by a method of crossover - swapping portions of parent trees at a random level.

REQ-22: Program shall perform mutation on each child individual by a method of changing out a value of a node in their trees - either an operator or operand at some level.

REQ-23: Program shall evaluate the fitness of each child.

# Other Nonfunctional Requirements

## Performance Requirements

TBD

## Safety Requirements

Not applicable in this class assignment.

## Security Requirements

Not applicable in this class assignment.

## Software Quality Attributes

TBD: Reliability, determining correct ranges of input settings, on-time performance of the algorithms, ability to hone inputs to get likelihood of close matches.

## Business Rules

TBD: Input range limits, time limits, etc.

# Other Requirements

None identified for this draft.

Appendix A: Data Dictionary

|  |  |  |  |
| --- | --- | --- | --- |
| **Data Name** | **Data Type** | **Description/Process** | **Possible Values** |
| Crossover NodeIDs A, B | Integer | Randomized Node ID of each of two Individuals’ trees that will be the crossover target in each |  |
| Crossover SubTrees A.B | EquationTree | Subset of each of the crossover individuals’ trees which will be placed at the Crossover Node ID in the other tree |  |
| Crossover TreeIDs A, B | Integer | Randomized IDs of two remaining trees in the PostSelection Population list; trees will be selected and paired for crossover with each other until none remain. |  |
| Crossover Trees AMod(n), BMod(n) | EquationTree Array(Nx2) | Randomly selected Pairs of PostSelection trees which will be/have been targets for the crossover section of the other in the Crossover operation |  |
| Fitness Bar | Integer | User input percent of the population which will be kept after Natural Selection | 1 - 99 |
| Individual Fitness Value | Integer | Sum of the Tree Y Deltas from 1 to quantity of training Ys | 0 to any with 0 being an exact match to the target equation result |
| Individual ID | Integer | Unique identifier for each candidate in a population | 100-1000 |
| Individual Tree(n) | EquationTree | Collection and sequence of Nodes making up a candidate equation | Example:  (x \* x + 20 -6 / 4/ 9 \* 114) / (2 \* x + 5 – 6 / 2 – x) |
| Most Fit Equation | Integer | ID of the tree after crossover and mutation whose resulting fitness value is the lowest |  |
| Mutation NodeID(n) | Integer | Randomized NodeID generated for each tree in Crossover Trees population |  |
| Mutation Operand Node | Integer | Random replacement operand for a node in a crossover tree |  |
| Mutation Operator Node | Integer | Random replacement operator for a node in a crossover tree |  |
| Node ID | Integer | Unique identifier of a specific tree node | 1-100? |
| Population | List | List of Individual Trees for fitness evaluation |  |
| Population Size | Integer | Quantity of individual equations in the population | 100-1000 |
| PostSelection Population | List | Even numbered collection of Individual tree equations remaining after Natural Selection |  |
| Program Duration | Integer | Minutes input by user for the length of time the program has to run | Default: 15 |
| Target Equation - Required | EquationTree | The target for individuals’ fitness when being evaluated  (Not sure this is needed in the program as long as training data is provided as input) | (-3X^2 + 7) / 2) |
| Target Equation - Optional | EquationTree | The target for individuals’ fitness when being evaluated  (Not sure this is needed in the program as long as training data is provided as input) | (-3X^3 + 7) / 2) |
| Training Data | Nx2 Array | Collection of the training X and Y pairs for the target equation | (x1, y1) – (xN, yN) |
| Training Data Size | Integer | Number of X and Y pairs in the training data that will be used to iterate over the training data with each Individual equation |  |
| Tree Depth | Integer | Limit to number of levels deep a candidate tree can be | 1-30? |
| Tree Node Count | Integer | Number of total nodes in an Individual’s tree used for iterating over nodes to locate crossover and mutation locations |  |
| Tree Operand Max | Integer | User input for the highest an operand value can be | 1 - 100 |
| Tree Operand Min | Integer | User input for the lowest an operand value can be | -100 - 0 |
| Tree Operand Node | TreeNode | Operator or Operand node of the candidate equation tree | x, integers -100 – 100? |
| Tree Operator Node | TreeNode | Operator or Operand node of the candidate equation tree | + - \* / |
| Tree Y Delta (n) | Integer | Absolute value of the difference between the Tree Y Result (n) and the Training Data Y (n) | Any positive integer |
| Tree Y Result(n) | Integer | Result of running the training Xs through the new equation tree individual | Any. Rounding will be fine given the wide range of training data Xs we will use |

Appendix B: Analysis Models



Figure – Flowchart

Figure 2 – Data Flow Diagram





Figure 3 - Class Diagram

Appendix C: To Be Determined List

<Collect a numbered list of the TBD (to be determined) references that remain in the SRS so they can be tracked to closure.>

Source: http://www.frontiernet.net/~kwiegers/process\_assets/srs\_template.doc