G.R.U. Software
400 E. 8th Ave
Ellensburg, WA 98926

Test Plan

SALMS

Stochastic Asset/Liability Model Sampling

Editor:	Mitsuharu Yasuda
Contributors:	Doug Love
	Keith Lambert
	Paul Reed

Table of Contents

Introduction

Project Overview

G.R.U. Software is designing software for Dr. Yvonne Chueh so she can calculate the probabilities of different interest rate scenarios. Dr. Chueh, a math professor at Central Washington University, is doing research on formulas that find the probability of different 30 one-year interest rate scenarios occurring. SALMS will greatly assist Dr. Chueh in this research.

Project Scope

SALMS will accept input from the user, calculate the probabilities of different interest rate scenarios occurring and display the results. The following parameters are entered by the user: universe size, sample size, path to the results file, path to the universe file, minimum interest rate, maximum interest rate, first year interest rate and a weight variable. The universe, which consists of different 30 one-year interest rate scenarios, is processed by SALMS. This universe could contain up to 100,000 different scenarios.

The probabilities will be found using three different algorithms: Modified Euclidean Distance Formula, Relative Present Value Distance Formula, and the Significance Method. After the algorithm has been executed, the results will be displayed to the screen. The results appear as a list of the scenarios with their probabilities, the interest rate values for that scenario and a line graph showing the interest rate values. A user can generate reports with these results.

Document Preview

This document outlines the test plan that G.R.U. Software will use to ensure that SALMS meets the requirement specifications. The test plan is organized into six sections, which include the testing process, requirements traceability, testing items, testing schedule, test recording procedures, and hardware and software requirements/constraints. We are confident that our testing procedures will minimize the number of defects in the final product.

Testing Process

Section Overview

Testing is the set of activities that ensures the software correctly implements the specific functions and requirements. G.R.U. Software's goal is to ensure that SALMS will contain the minimum number of defects.

Because errors occur during development, G.R.U. Software has implemented a defect tracking system. For each error found a Defect Tracking Form (Appendix B) will be completed. This will ensure that all errors found are acknowledged and corrected in a timely manner.

The testing process is comprised of five types of testing: code reviews, unit testing, integration testing, system testing and validation testing.

Code Reviews

G.R.U. Software has developed standards for the coding and commenting of source code. We will review the code to ensure the design is correct and meets the requirement specification. When the code reviews are completed, we will test each algorithm using a verification vehicle provided by our client.

Unit Testing

Unit testing is the process of ensuring that each algorithm works as it is intended. Our user interface will be developed using Visual Basic and our processing component will be developed as a DLL in C++. We will include black and white box testing of each component as it is developed. This type of testing will verify that the code produces the expected output based on the given input. In order to test results of the sampling, we use a Verification Vehicle to ensure we are getting the correct output. See our Testing Item section for a complete explanation of how we plan to test the algorithm correctness

Integration Testing

Integration testing verifies that the separate modules work as a whole. Defects may occur as the different modules are put together. Integration testing will be completed using black box testing technique developed by the tester.

System Testing

System testing verifies that the completed program works on the target machine. By performing system testing we can ensure that the necessary memory and processor speed is available.

Validation Testing

Validation testing ensures that the system meets the requirements specification. G.R.U. Software will install a copy of SALMS on both of the client's machine. The client will run SALMS and test the functionality. If it has met all of her requirements, she will sign off on the project. All functional and performance requirements must meet the client's satisfaction.

[redacted]

Requirements Traceability

Section Overview

This section gives an outline of the system performance that our client desires. The requirement specification document outlines the functional requirements. This section is organized by the listing the requirements required to meet these specifications. The tester will fill out a Black Box Testing Form (see Appendix A) before testing begins.

- Creates new universe
    1) Enter parameters
        - Universe file path
        - Result file path
        - Size of universe
        - Size of sample
        - First year interest rate
        - Minimum interest rate
        - Maximum interest rate
    2) Create universe
    3) Process universe
    4) Display results
- Loads previous universe
    1) Select universe file to load
    2) Select result file to save
    3) Select sample size
    4) Select algorithm
    5) Load universe file
    6) Process universe
    7) Display results
- Loads previous results
    1) Select results file to load

2) Load results file
3) Display results

- Recovers the data
  1) Load recover data
  2) Load universe file
  3) Resume processing universe
  4) Display results
- Batch mode
  1) Add batch file
     - Create universe
       - Get parameters
         - Universe file path
         - Result file path
         - File name and path
         - Universe size
         - Sample size
         - First year interest rate
         - Minimum interest rate
         - Maximum interest rate
     - Load universe
       - Universe file path
     - Choose algorithms
  2) Create all universe
  3) Loop thru queue
  4) Perform algorithm
  5) Save the result
  6) Delete batch file

Testing Items

Section Overview

This section describes how we will test SALMS. G.R.U Software will test using equivalence partitioning to reduce the total number of test cases. The test cases will include valid and invalid input values. This section includes tables that display the test cases. Each test case will contain the name of the item being tested, the input, and the expected result. The expected result is compared with the actual result to determine if a defect is present. If so, the defect tracking process will be followed.

Testing each of the algorithms will involve the use of a verification vehicle provided by our client. The verification vehicle is a Microsoft Excel spreadsheet with macros that execute each of the algorithms. Our plan is to run the same sample universe in the spreadsheet and on SALMS using the same sample size. We will do this for all of the algorithms. After they are executed, we will manually compare the values to ensure correctness. The following tables detail our testing items.

Item	**Input Condition**	**Result**
Universe path	Valid path	OK
	Invalid path	FILE_NOT_FOUND
Result path	Valid path	OK
	Invalid path	FILE_NOT_FOUND
Recover path	Valid path	OK
	Invalid path	FILE_NOT_FOUND

• Table 1 - File path

Item	**Input**	**Result**
Modified Euclidean – universe size	-1	UNIVERSE_SIZE_ERR
	0	EMPTY_UNIVERSE
	1,000	OK
	100,000	OK
Modified Euclidean – sample size	-1	SAMPLE_SIZE_ERR
	0	SAMPLE_SIZE_ERR
	1	OK
	150	OK
	200	OK
	201	SAMPLE_SIZE_ERR
Modified Euclidean – weight	-1	WEIGHT_ERR
	0	WEIGHT_ERR
	1.06	OK
	10	OK

• Table 2 - Modified Euclidean algorithm

Item	**Input**	**Result**
Relative present – universe size	-1	UNIVERSE_SIZE_ERR
	0	EMPTY_UNIVERSE
	1,000	OK
	100,000	OK
Relative present – sample size	-1	SAMPLE_SIZE_ERR
	0	SAMPLE_SIZE_ERR
	1	OK
	150	OK
	200	OK
	201	SAMPLE_SIZE_ERR

• Table 3 - Relative present algorithm

Item	**Input**	**Result**
Significance method – universe size	-1	UNIVERSE_SIZE_ERR
	0	EMPTY_UNIVERSE
	1,000	OK
	100,000	OK
Significance method – sample size	-1	SAMPLE_SIZE_ERR
	0	SAMPLE_SIZE_ERR
	1	OK
	150	OK
	200	OK
	201	SAMPLE_SIZE_ERR

• Table 4 - Significance method algorithm

Testing Schedule

Section Overview

The testing schedule outlines the milestones for unit, integration and validation testing. We will test SALMS with three different machines to ensure that the requirements are met. The following schedule will ensure that SALMS is completed by the end of winter quarter.

Date	Testing
January 30	Start DLL testing
February 1	Start test for displaying result and loading results path
February 4	Start test for loading universe path
February 7	Start test for creating universe path
February 11	Start to test batch mode
February 14	Start to test recovery function
February 15	Start to test Printing function
February 17	Start to test Calibration function
February 22	Start to test Progress bar function
February 25	Validation testing begins with target machines

• Table 5 – Testing Schedule

Test Recording Procedures

Section Overview

G.R.U. Software has instituted a rigorous testing procedure that will be well documented. This section will describe the documentation process.

Test Procedures

When a module is ready for Black Box testing, the developer will inform the tester. The tester then fills out a Black Box Testing Form (see Appendix A). When the module is tested, the tester will complete the form by filling in the data used to test the module, the desired output and the actual output. If an error is discovered, a Defect Tracking Form (Appendix B) will be filled out by the tester and given to the developer.

The same procedure is used for testing the C++ DLL. However, instead of using the Black Box Testing Form, the tester will use the DLL Function Testing Form (see Appendix C). As each portion of the DLL is completed, the tester will verify that the actual output coincides with the desired output. If an error is discovered, a Defect Tracking Form will be filled out by the tester and given to the developer.

In either case, the developer and the tester will keep the rest of the team apprised of the status of each stage and module. Also, any and all errors discovered while testing will be passed onto the team leader and the QA manager for tracking purposes.

Hardware Requirements and Constraints

System Requirements

SALMS is designed to run on two different machines – Dr. Chueh's office and home computers. The office machine is a Pentium IV 1.7 GHz processor and the home machine is a Pentium IV 2 GHz processor. Both machines have 256 MB RAM, running Windows 2000 Professional operating system and will use a 1024x768 16-bit color screen resolution.

Testing System

G.R.U. Software will be testing SALMS on three different machines. All three systems will use a 1024x768 16-bit color screen resolution. Primary testing will be done using a Pentium III 667 MHz processor that has 512 MB RAM running Windows XP Home Edition. Additionally, because both target systems use the Windows 2000 Professional operating system, further testing will be done on a 533 MHz Intel Celeron processor system. This system has 512 MB RAM and uses the same operating system as the target machines.

The development machine G.R.U. Software will use has a 1 GHz Intel Celeron processor, 512 MB RAM and is running Windows XP Professional operating system. This machine will also be used as an initial testing platform during development.

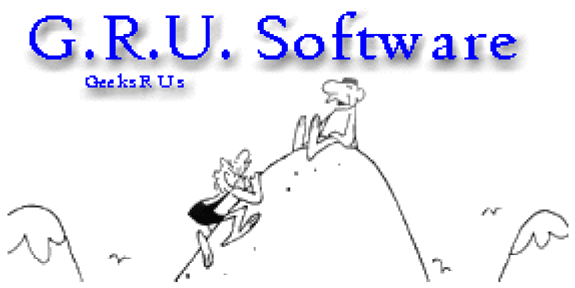
SALMS has significant memory requirements. Values in the SALMS universe are of type double. Each double requires 8 bytes of memory and each row of the universe will contain 30 values. Thus, each row of the universe requires 240 bytes of memory. In a worst case scenario there will be 100,000 rows in the universe, requiring approximately 24 MB RAM. Further the universe requires approximately 35 MB of disk space.

Conclusion

Testing demonstrates that the software appears to be working according to the required specification. We will attempt to produce software that contains the minimum number of defects and meets out client's requirements as detailed in our Requirements Specification. G.R.U. Software is taking steps to achieve this by detailing our design in our Software Design Specification. We are dedicated to ensuring that SALMS is developed as a high quality system. All documents can be found online at http://tor.lab.cwu.edu:9005/cs480f2001_07/archives.

Appendices

Appendix A: Black Box Testing Form



Tester:	
Date:	
Unit:	
Inputs:	
Desired Output:	
Actual Output:	
The actual output is correct with desired output: Yes No	
Type of Error: Fatal Moderate Slight None	
How the error is resolved:	
Fixed by:	
Fixed Date:	

Appendix B: Defect Tracking Form

G.R.U. Software

Geeks R Us

Tester: ______________________________

Date: ______________________________

Module: ______________________________

Type of error: __________

F - Fatal
M - Moderate
S – Slight

Defect Description:

____

____

How the defect was found:

____

____

Comments:

____

____

____

Date Fixed: _________________

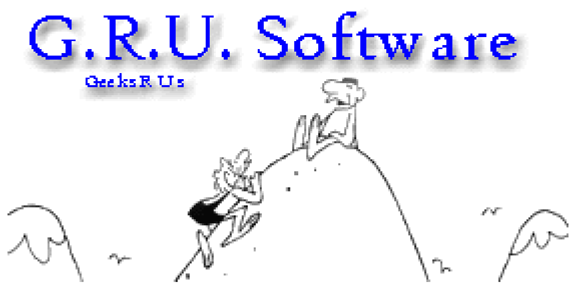
Cause of Error:

____

____

____

Appendix C: DLL Function Testing Form



Tester:	
Date:	
Method:	___ Modified Euclidean ___ Modified Euclidean Universe Size ___ Relative Present ___ Relative Present Universe Size ___ Significance Method ___ Significance Method Universe Size ___ Get Universe Size ___ Load Universe
Input:	Valid / Invalid Universe Path: ☐ ☐ Result Path: ☐ ☐ Recover Path: ☐ ☐ Universe Size: ____________ Sample Size: ____________ Weight: ____________
Desired Output:	
Actual Output:	
The actual output is correct with desired output: Yes No	
How was the error resolved:	
Fixed by:	Fixed Date: