ECE 322 Lab 5 Report

Introduction

- Statement of problem
- Identification of applications under test
- Identification of the methodologies being used

Answer:

The objective of this lab was to apply Incremental Integration testing with a top-down and bottom-up approach alongside unit testing. The testing methodologies used are therefore White Box and Black Box testing. This testing was performed on a simple database program consisting of a hierarchy of 7 modules with an aim of achieving 100% statement coverage.

Part 1

Q1: Definition of the problem, including what the program does, what testing method is used:

Answer:

The goal of the lab is two employ the two main approaches of Incremental Integration testing, i.e., top-down and bottom-up on the given database program. The database program allows a user to store a name and phone number pair in a .txt file which persists after the program is exited. The user can add, update, delete any pair from the database as well as sort and display the entries for the user to see. The program can also interact with any .txt file specified by the user. Integration testing is performed (Black Box testing) using two approaches, top-down and bottom-up. In conjunction with unit testing (White Box testing) where applicable.

Q 2: List module/modules that you will be test in order according to test strategy:

Answer:

Top Down:

- A

- AB

- AC

- AD

- AE

- ABCDEFG

Bottom Up :

- F (unit test)
- G (unit test)
- BF
- CF
- DF
- DG
- E (unit test)
- ABCDEFG

Q3 Test Case Tables (Top Down and Bottom Up):

Answer:

Bottom Up

ID	Input	Modules under test	Description	Expected Result	Actual Result
1	A test Entry object with test name and number	F	Testing Module F displaysData to terminal correctly by redirecting it to a .txt file and validating against an expected .txt file	displayData method prints items in data list to terminal	displayData method prints items in data list to terminal
2	Two Entry objects with test names and numbers	G	Testing Module G updateData method updates the given .txt file with data list in program.	Contents of .txt file matches data list in program	Contents of .txt file matches data list in program
3	Two pairs of test names and numbers	BF	Testing loadFile method of Module B which retrieves data from a .txt and updates data list, also calls lower level ModuleF to print data	Data is retrieved from .txt file and displayed to terminal	Data is retrieved from .txt file and displayed to terminal
4	Three Entry objects with test names and numbers	CF	Testing sortData method of Module C which sorts data list and displays to terminal using lower level ModuleF	Data list is sorted and displayed to terminal in sorted order	Data list is sorted and displayed to terminal in sorted order
5	Index to delete	DF	Testing deleteData method of Module D that deletes data at given	Selected index is deleted from list	Selected index is deleted from list

			index and prints updated list to terminal		
6	Test name and number to add to list	DF	Testing insertData method of Module D that inserts data at end of list and prints updated list to terminal	Entry is inserted at end of list	Entry is inserted at end of list
7	**Test number and name update another entry in list**	**DF**	**Testing updateData method of Module D that updates data at given index and prints updated list to terminal**	**An entry in data list is updated with a new name and number**	**Incorrect entry is updated in list**
8	Index to delete	DG	Testing deleteData method of Module D that deletes data at given index from .txt file	Selected index is deleted from .txt file	Selected index is deleted from .txt file
9	Test name and number to add to file	DG	Testing insertData method of Module D that inserts data at end of .txt file	Entry is inserted at end of .txt file	Entry is inserted at end of .txt file
10	**Test number and name update another entry in .txt file**	**DG**	**Testing updateData method of Module D that updates data at given index in .txt file**	**An entry in .txt file is updated with a new name and number**	**Incorrect entry is updated in .txt file**
11	None	E	Testing exitProgram method of Module E prints message to terminal and calls sys.exitI()	Message is printed to terminal and sys.exit is called	Message is printed to terminal and sys.exit is called
12	Test name and number to add to DB	ABCDEFG	Testing parseAdd method of Module A adds test name and number to program	Call to method returns true	Call to method returns true
13	Index to delete	ABCDEFG	Testing parseDelete method of Module A deletes entry from DB	Call to method returns true	Call to method returns true
14	.txt file	ABCDEFG	Testing parseLoad method of Module A loads data from .txt file into program	Call to method returns true	Call to method returns true
15	Index, name and number	ABCDEFG	Testing parseUpdate method updates entry in DB with new name and number	Call to method returns true	Call to method returns true
16	**None**	**ABCDEFG**	**Testing runExit method of Module A**	**Program prints to terminal and calls sys.exit**	**AttributeError**
17	None	ABCDEFG	Testing of runSort method of Module A	Call to method returns true	Call to method returns true
18	“add” command w/ new name and number	ABCDEFG	Testing run method of Module A with “add” command, prints are redirected to .txt file	Data Entry list is updated at specified index w/ correct data	Data Entry list is updated at specified index w/ correct data

19	Index to delete along with data list	ABCDEFG	Testing run method of Module A with "delete" command, prints are redirected to .txt file	Data Entry list is updated	Data Entry list is updated
20	**None**	**ABCDEFG**	**Testing run method of Module A with "exit" command, prints are redirected to txt file**	**System exit, and print message to terminal**	**Attribute Error**
21	.txt file to load	ABCDEFG	Testing run method of Module A with "load" command, prints are redirected to .txt file	Data is loaded from .txt into data list	Data is loaded from .txt into data list
22	"sort" command	ABCDEFG	Testing run method of Module A with "sort" command, prints are redirected to .txt file	Data list is sorted and printed to terminal	Data list is sorted and printed to terminal
23	"update" command w/ new name and number	ABCDEFG	Testing run method of Module A with "update" command, prints are redirected to .txt file	Data list is updated as well as .txt file is updated	Data list is updated as well as .txt file is updated
24	Data.txt, commands, names/ number pairs, index's	ABCDEFG	Test Entire system by loading data from file, adding, updating and deleting data, etc. AND displayHelp method	System performs all commands and prints all info to terminal	System performs all commands and prints all info to terminal

Top-Down

ID	Input	Modules under test	Description	Expected Result	Actual Result
1	None	A	Test displayHelp method of Module A.	Correct display help message printed to .txt file	Correct display help message printed to .txt file
2	Name/ number entry	A	Test parseAdd method of Module A.	Method call sets data != None	Method call sets data != None
3	**Index to delete**	**A**	**Test parseDelete method of Module A.**	**ModuleD.deleteData called with correct parameters**	**ModuleD.deleteData called with incorrect parameters, index not matching**
4	.txt file	A	Test parseLoad method of Module A.	Method call returns true or false depending on if data == none or not	Method call returns true or false depending on if data == none or not
5	**Name, number and index**	**A**	**Test parseUpdate method of Module A.**	**ModuleD.updateData is called with correct arguments**	**ModuleD.updateData is called with incorrect arguments, index do not match**

6	None	A	Test runExit method of Module A.	runExit method is called once	runExit method is called once
7	None	A	Test runSort method of Module A.	runSort method returns true or false depending on if data == none or not	runSort method returns true or false depending on if data == none or not
8	“” and “help”	A	Test empty command and help command in run method of Module A.	Correct display help message printed to .txt file	Correct display help message printed to .txt file
9	**Index and “delete” command**	**A**	**Test delete command in run method of Module A.**	**ModuleD.deleteData method is called with correct arguments**	**ModuleD.deleteData method is called with incorrect arguments, index do not match**
10	“exit” command	A	Test exit command in run method of Module A.	ModuleE.exitProgram method is called once	ModuleE.exitProgram method is called once
11	.txt file name and “load” command	A	Test load command in run method of Module A.	ModuleB.loadFile is called once	ModuleB.loadFile is called once
12	“sort” command	A	Test sort command in run method of Module A.	ModuleC.sortData is called once	ModuleC.sortData is called once
13	**Name & number entry w/ “update” command**	**A**	**Test update command in run method of Module A.**	**ModuleD.updateData is called with correct arguments**	**ModuleD.updateData is called with incorrect arguments, index’s do not match**
14	.txt file name	AB	Test load method in Module A with implementation of Module B	parseLoad method returns true when called	parseLoad method returns true when called
15	“load” command w/ .txt file name	AB	Test “load” command of run method in Module A with implementation of Module B	Messages printed to terminal are as expected	Messages printed to terminal are as expected
16	Data list to sort	AC	Test sort method in Module A with implementation of Module C	runSort method returns true when called	runSort method returns true when called
17	“sort” command	AC	Test “sort” command of run method in Module A with implementation of Module C	Messages printed to terminal are as expected	Messages printed to terminal are as expected
18	Name/ number entry to add	AD	Test parseAdd method of Module A with implementation of Module D	parseAdd method returns true when called	parseAdd method returns true when called

19	Index to delete, data list	AD	Test parseDelete method of Module A with implementation of Module D	parseDelete method returns true when called	parseDelete method returns true when called
20	Name/ number entry, and index	AD	Test parseUpdate method of Module A with implementation of Module D	parseUpdate method returns true when called	parseUpdate method returns true when called
21	"add" command & name/ number entry	AD	Test "add" command of run method in Module A with implementation of Module D	Messages are printed to terminal as expected	Messages are printed to terminal as expected
22	"delete" command & index to delete	AD	Test "delete" command of run method in Module A with implementation of Module D	Messages are printed to terminal as expected	Messages are printed to terminal as expected
23	"update" command with new name, number and index	AD	Test "update" command of run method in Module A with implementation of Module D	Messages are printed to terminal as expected	Messages are printed to terminal as expected
24	**None**	**AE**	**Test runExit method of Module A with implementation of Module E**	**Message prints to terminal and sys.exit is called**	**AttributeError**
25	**"exit" command**	**AE**	**Test "exit" command of run method in Module A with implementation of Module E**	**Message prints to terminal and sys.exit is called**	**AttributeError**
26	Name and number to add	ABCDEFG	Test parseAdd method of entire system.	Method returns true when called	Method returns true when called
27	Index to delete	ABCDEFG	Test parseDelete method of entire system.	Method returns true when called	Method returns true when called
28	.txt file to load from	ABCDEFG	Test parseLoad method of entire system.	Method returns true when called	Method returns true when called
29	Name, number and index to update	ABCDEFG	Test parseUpdate method of entire system.	Method returns true when called	Method returns true when called
30	**"exit" command**	**ABCDEFG**	**Test exit method of entire system.**	**Message prints to terminal and sys.exit is called**	**AttributeError**
31	"sort" command	ABCDEFG	Test sort method of entire system.	Method returns true when called	Method returns true when called
32	"add" command and new	ABCDEFG	Test "add" command of entire system.	Message is printed terminal as expected	Message is printed terminal as expected

	name/ number				
33	"delete" command and index	ABCDEFG	Test "delete" command of entire system.	Message is printed terminal as expected	Message is printed terminal as expected
34	**"exit" command**	**ABCDEFG**	**Test "exit" command of entire system.**	Message is printed terminal as expected and sys.exit is called	**AttributeError**
35	"load" command and .txt file	ABCDEFG	Test "load" command of entire system.	Message is printed terminal as expected	Message is printed terminal as expected
36	"sort" command	ABCDEFG	Test "sort" command of entire system.	Message is printed terminal as expected	Message is printed terminal as expected
37	"update" command along with new name, number and index	ABCDEFG	Test "update" command of entire system.	Message is printed terminal as expected	Message is printed terminal as expected
38	Varying commands, names, numbers, index's, etc.	ABCDEFG	Test entire system, using all commands in specific order as a user would interact with the system, load, add, delete, etc	Messages is printed terminal as expected	Messages is printed terminal as expected

Q4: Discussion

Comments on the effectiveness of integration testing

Q4.1: Is module isolation effective?

Answer:

Not necessarily. In Bottom up approach the lowest level modules are unit tested. In Top Down approach no modules are unit tested. The overall Incremental approach focuses on interactions between modules, not testing them in isolation. Big Bang or non-Incremental testing focuses more on isolation of each model before integration.

Q4.2 : What does each type of integration testing target?

Answer:

Top Down integration testing targets system architecture. While Bottom up integration testing targets correctness of modules.

Q4.3: Would this type of testing scale well to a large scale system?

Answer:

Bottom up integration testing would scale well to a large system because it most likely allows work to proceed in parallel. Top down could also work if a test driven development approach is taken and higher level modules are built relatively early.

Q4.4: Are stubs and drivers useful and effective?

Answer:

Yes. They help test interactions between only the modules you wish, at any level in the hierarchy.

Q4.5 : Which type of integration testing do you think is best?

Answer:

It depends on the approach of building the software. If a bottom up approach is taken to build the software then bottom up testing will work best for testing and vice versa.

Q4.6 : Which would be the most appropriate for a test driven development environment?

Answer:

I believe that Top Down integration testing would work best for a test driven development environment since stubs are used. All the tests are written beforehand for all functionality (test driven) in the highest level module and then when lower level modules are built they can be substituted in for testing.

Q4.7: Which would be the most appropriate for library development?

Answer:

Bottom up testing. Since librarys are generally built from the bottom up. Building on previously implemented functionality.

Q4.8: Which is the easiest to maintain?

Answer:

Since generally drivers are easier to implement then stubs Bottom up is a better option generally.

- Q4.9: Additional comments: Consider these (Q4.1 to Q4.8) and similar questions in your discussion of integration testing to show you have a strong understanding of the underlying concepts and motivations of this testing technique.

Answer:

In this lab Top down and Bottom up integration testing served as a necessary and effective step in testing the provided program alongside partial unit testing. Without integration testing it is unknown if the modules interface with each other as expected. Several errors were discovered in the program. The majority of these errors were indexing errors in Modules A and D. I believe there was some confusion in the design of the program, for example in Module D the delete & update methods perform their actions at a given index, while the update method performs its action an index+1. Also, in Module A there were indexing errors when calling the aforementioned methods in Module D. I believe this had to do with the fact that the records in the Database started indexing at 1, instead of 0. I believe that handling the indexing all in Module A is the most elegant solution instead of having it spread across both Module A and D. The other major type of error was when commanding the program to exit, there was a spelling mistake in calling ModuleE's exit method, this gives an AttributeError because ModuleE does not in fact have a attribute/method called 'exitProgam". The obvious fix is to add the letter 'r' to fix this issue. The modules were not isolated effectively using either Top Down or Bottom Up. Incremental integration testing is better suited to effectively testing the interactions between modules as they are combined one by one. It must also be paired with unit testing of every module if testing modules in isolation is a priority, else Non-Incremental testing may be a better approach. It is possible for either Top Down or Bottom Up to be an effective approach to integration testing depending on how the system is being built. It should not be forgotten that stubs are generally more complicated to implement than drivers, and Top Down integration relies heavily on stubs.

Code component:

See other directories for code. Or can be found at https://github.com/mattheas

Top down integration tests are found in the /tests directory while Bottom up integration tests are found in /bottomuptests.

As a note, whenever running either TestRunnerTopDownIntegration.py test suite or TestRunnerBottomUpIntegration.py or any TestModuleX.py individually I used the following command from inside the /Lab4_src:

"PYTHONPATH=$(pwd) python3 tests/ TestRunnerTopDownIntegration.py"

I ran all my tests through the command line, nothing through an IDE. I am almost certain test cases will fail if not called from the /Lab4_src directory with a similar command as above because it is hardcoded where to find specific .txt files. An example is shown below:

```
matt@matt-HP-G62-Notebook-PC:~/Desktop/ECE_322_Lab5/Lab4_src$ PYTHONPATH=$(pwd) python3 bottomuptests/TestRunnerBottomUpIntegration.py
```

Please ignore any irrelevant files in any directorys. The TestRunner files contain the correct tests and ordering for Top Down and Bottom Up integration tests. Also remember that data.txt is updated every time a test suite is run. So it MUST be cleared of its contents and reset to its initial given values, i.e., names and numbers.

Conclusion

Answer:

The purpose of the lab was to explore integration testing further and implement Incremental approaches as well as compare two approaches of incremental integration testing (Top down vs. Bottom up). Benefits and drawbacks compared to Big Bang integration testing was also explored.