


ECE 322 Lab Report #4

 Benjamin Kong
1573684

November 23, 2021

Introduction

The purpose of this lab was to introduce integration white box testing techniques. We will use Python and Python unittest `unittest.mock` to gain experience with integration testing.

Integration testing is the logical extension of unit testing. There are two common approaches:

- Non-incremental testing (big bang): where each module is tested independently, then the system is tested as a whole, and
- Incremental testing: where the set of previously tested modules are combined with the next module to be tested before running tests. There are two common methods of incremental testing:
 - Bottom up: where the lowest level modules are tested in isolation, then higher level modules are added incrementally, and
 - Top down: where the highest level modules are tested in isolation (stubbing lower level modules), then lower level modules are added incrementally.

Usually, when integration testing is performed, we need to use stubs and drivers. In this lab, we will be creating mock objects which are essentially used as stubs in our integration tests.

- Stubs are used as a stand in for lower level modules that aren't currently being tested. Stubs return dummy values or makes an assertion so that the higher level modules can still run and be tested.
- Drivers are a piece of testing code that make it possible to call the submodule of an application by itself. Often, driver code requires stub setup and object initialization.

We will be testing a simple database system that has seven modules, Module A to module F. We will be focusing on non-incremental testing (big bang integration).

Task

For this lab, we tested the database application using the big bang testing technique. We created unit tests for each module A-F. Any time a module depended on another module, we mocked that module. The code for the tests have been included with the submission under `Lab4_src/tests`. The table of tests and results is displayed below.

Table 1: Test cases for `bisect.py`

Test ID	Description	func	x1	x2	Expected	Pass
1	Normal case	$x + 1$	-10	10	0	Y

An image of the coverage report is provided below.

Figure 1: Coverage report of `mybisect.py`. The only relevant value has been marked with a red arrow.

Conclusion

The purpose of this lab was to introduce integration white box testing techniques. We used Python and Python unittest `unittest.mock` to gain experience with integration testing.

Integration testing is the logical extension of unit testing. There are two common approaches: non-incremental testing (big bang) where each module is tested independently, then the system is tested as a whole, and incremental testing where the set of previously tested modules are combined with the next module to be tested before running tests. There are two common methods of incremental testing: Bottom up where the lowest level modules are tested in isolation, then higher level modules are added incrementally, and top down where the highest level modules are tested in isolation (stubbing lower level modules), then lower level modules are added incrementally.

Usually, when integration testing is performed, we need to use stubs and drivers. In this lab, we created mock objects which are essentially used as stubs in our integration tests. Stubs are used as a stand in for lower level modules that aren't currently being tested. Stubs return dummy values or makes an assertion so that the higher level modules can still run

and be tested. Drivers are a piece of testing code that make it possible to call the submodule of an application by itself. Often, driver code requires stub setup and object initialization.

In the lab, we tested a simple database system that had seven modules, Module A to module F. We focused on non-incremental testing (big bang integration).