Florida International University
School of Computing and Information Sciences

CIS 4911 - Senior Capstone Project
Software Engineering Focus

Final Deliverable

Project Title: Learning OpenFlow SDN for Research and Experimentation - Version 1.0

Team Members:
Jahkell Lazarre
Steven Lyons

Product Owner(s):
Jeronimo Bezerra
Vasilka Chergarova

Mentor(s):
Mohsen Taheri

Instructor: Masoud Sadjadi

Abstract

This document presents the information necessary to gain a good understanding of the Learning OpenFlow SDN for Research and Experimentation - Version 1.0 project. The project involves multiple tools and scripts to assist network engineers with the configuration, testing, and validation of SDN networks. In the following pages, user stories, diagrams, and information will be presented that will describe the project in detail.

Table of Contents

INTRODUCTION 5

CURRENT SYSTEM 5

PURPOSE OF NEW SYSTEM 5

USER STORIES

IMPLEMENTED USER STORIES 7

PENDING USER STORIES 10

PROJECT PLAN

HARDWARE AND SOFTWARE RESOURCES 12

SPRINTS PLAN 13

Sprint 1 13

Sprint 2 13

Sprint 3 14

Sprint 4 15

Sprint 5 16

Sprint 6 17

Sprint 7 18

SYSTEM DESIGN

ARCHITECTURAL PATTERNS 20

SYSTEM AND SUBSYSTEM DECOMPOSITION 21

DEPLOYMENT DIAGRAM 22

DESIGN PATTERNS 22

SYSTEM VALIDATION 23

GLOSSARY 37

APPENDIX 38

APPENDIX A - UML DIAGRAMS 38

Static UML Diagrams 38

Dynamic UML Diagrams 40

APPENDIX B - USER INTERFACE DESIGN 52

APPENDIX C - SPRINT REVIEW REPORTS 69

APPENDIX D - USER MANUALS, INSTALLATION/MAINTENANCE DOCUMENT, SHORTCOMINGS/WISHLIST DOCUMENT AND OTHER DOCUMENTS 74

REFERENCES 80

INTRODUCTION

Research in the field of Software Defined Networking (SDN) is largely unaware of several restrictions encountered when deploying a proposed architecture on real devices. OpenFlow (OF) implementation documentation has limitations when working with switches from different vendors. Different versions of OF may also give unpredictable errors and in some cases crash the switch. AmLight is a production network and cannot have such errors.

Current System

This is the first iteration of *Learning OpenFlow SDN for Research and Experimentation*; however, network engineers have been performing many of the tasks that this project aims to accomplish - that is, test and validate OpenFlow switches. Currently, network engineers must run switch validation software such as *OFTest* and *Ryu*, parse verbose output from these software packages, and manually record the results in an Excel spreadsheet.

Purpose of New System

The new system *SwitchTester* is simply an automated process, implemented in software, that runs any of the aforementioned switch validation software, generates a report in which test case names, test case results (e.g. OK, ERROR, FAIL), and error descriptions are listed. The new system also defines an application profile that allows the end-user to specify a compatibility list of test cases - a feature that is has yet to be fully implemented and publicly known in current testing frameworks as of the publication date of this document. The compatibility list is used to ensure that the target switch passes the specified test cases; if one fails, the target switch is said to have not passed validation.

USER STORIES

The following section provides the detailed user stories that were implemented in this iteration of the *Learning OpenFlow SDN for Research and Experimentation - Version 1.0* project. These user stories served as the basis for the implementation of the project's features. This section also shows the user stories that are to be considered for future development.

Implemented User Stories

USER STORY #126 Implement the sample topology using Mininet and OESS.

- As a project member I need to simulate the sample topology so that I can show that my testing environment and understanding works properly for future development and testing.

Acceptance Criteria

- Be able to demo the working environment.
- Creating OpenFlow entry in working environment.
- Host connected to Mininet must support VLAN configuration.

USER STORY #121 Create an SDN validation environment to test the switch's Control, Data and Management Planes with OFTest (Part 1)

- As a user of AmLight I need a way to validate switches automatically to ease the installation of switches into AmLight.

Acceptance Criteria

- Understand what is OFTest
- Run on Mininet

USER STORY #143 Create an SDN validation environment to test the switch's Control, Data and Management Planes with Ryu (Part 1)

- As a user of AmLight I need a way to validate switches automatically to ease the installation of switches into AmLight.

Acceptance Criteria

- Understand what is Ryu
- Run on Mininet

USER STORY #142 Create an SDN validation environment to test the switch's Control, Data and Management Planes with OFTest (Part 2)

- As a user of AmLight I need a way to validate switches automatically to ease the installation of switches into AmLight.

Acceptance Criteria

- Generate Report

USER STORY #144 Create an SDN validation environment to test the switch's Control, Data and Management Planes with Ryu (Part 2)

- As a user of AmLight I need a way to validate switches automatically to ease the installation of switches into AmLight.

Acceptance Criteria

- Understand what is Ryu
- Run on Mininet

USER STORY #143 Create an SDN validation environment to test the switch's Control, Data and Management Planes with OFTest (Part 3)

- As a user of AmLight I need a way to validate switches automatically to ease the installation of switches into AmLight.

Acceptance Criteria

- Run the script on 2 real devices
- Validate your report
- Define the application profile
- Create another report

USER STORY #145 Create an SDN validation environment to test the switch's Control, Data and Management Planes with Ryu (Part 3)

- As a user of AmLight I need a way to validate switches automatically to ease the installation of switches into AmLight.

Acceptance Criteria

- Run the script on 2 real devices.
- Validate your report
- Define the application profile
- Create another report

Pending User Stories

USER STORY #125 Add visibility of flows managed by ONOS to Zabbix.

- As a user of Zabbix and ONOS, I would like to see the flow statistics from ONOS in Zabbix, as Zabbix is my preferred way of monitoring my network, but cannot see flows managed by ONOS.

Acceptance Criteria

- Get familiar with Zabbix and ONOS
- Install ONOS and Zabbix on VMs

Get familiar of how to collect stat from ONOS:
https://wiki.onosproject.org/display/ONOS/Developer%27s+Guide

PROJECT PLAN

This section describes the planning that went into the realization of this project. This project incorporated the agile development techniques and as such required the sprints to be planned. These sprint plannings are detailed in the section. This section also describes the components, both software and hardware, chosen for this project.

Hardware and Software Resources

The following is a list of all hardware and software resources that were used in this project:

- Mininet Virtual Machine
  - Ubuntu Linux Distribution (default)
  - Git (default)
  - OFTest (default)
  - Ryu
  - Python 2.7 (default)
  - xmlrunner Python module
  - > 768 MB RAM
  - > 20GB disk space
  - > 1.9 GHz processor (x1)
- Windows PC
  - Windows 7
  - Oracle VirtualBox
  - 4 GB RAM
  - > 200 GB disk space
  - > 1.9 GHz processor (x1)
- Mingle

Sprints Plan

Sprint 1

Attendees: Steven Lyons, Jahkell Lazarre, Juan Medino, Steven Gamatan
Start time: 3:12pm
End time: 3:23pm

The product owner chose the following user stories to be done during the next sprint. They are ordered based on their priority.

- User Story #126
  As a project member I need to have a recreation of AmLight's topology on Mininet so that I may simulate OpenFlow entries with OESS for development.

The team members indicated their willingness to work on the following user stories.

- Steven Lyons, Jahkell Lazarre, Juan Medino, Steven Gamatan

User Story #126
As a project member I need to have a recreation of AmLight's topology on Mininet so that I may simulate OpenFlow entries with OESS for development.

Sprint 2

Attendees: Steven Lyons, Jahkell Lazarre
Start time: 2:41 pm
End time: 3:03 pm

After discussion, the velocity of the team were estimated to be 10.

The product owner chose the following user stories to be done during the next sprint. They are ordered based on their priority.

- User Story #125, Add visibility of flows managed by ONOS to Zabbix

The team members indicated their willingness to work on the following user stories.

- Steven Lyons
    - User Story #125, Add visibility of flows managed by ONOS to Zabbix
- Jahkell Lazarre
    - User Story #125, Add visibility of flows managed by ONOS to Zabbix

Sprint 3

After discussion, the velocity of the team were estimated to be: N/A

The product owner chose the following user stories to be done during the next sprint. They are ordered based on their priority.

- User Story #121, Part #1 Create an SDN validation environment to test the switch's Control, Data and Management Planes with OFTest
- User Story # 143 Part #1 Create an SDN validation environment to test the switch's Control, Data and Management Planes with Ryu

The team members indicated their willingness to work on the following user stories.

- Jahkell Lazarre
    - User Story #121, Part #1 Create an SDN validation environment to test the switch's Control, Data and Management Planes with OFTest

- Steven Lyons
    - User Story #143, Part #1 Create an SDN validation environment to test the switch's Control, Data and Management Planes with Ryu

Sprint 4

Attendees: Steven Lyons, Jahkell Lazarre

Start time: 2:00 pm

End time: 2:10 pm

After discussion, the velocity of the team were estimated to be 40.

The product owner chose the following user stories to be done during the next sprint. They are ordered based on their priority.

- User Story #141, Create an SDN validation environment to test the switch's Control, Data and Management Planes with OFTest (Part 2)
- User Story #144, Create an SDN validation environment to test the switch's Control, Data and Management Planes with Ryu (Part 2)

The team members indicated their willingness to work on the following user stories.

- Jahkell Lazarre
    - User Story #141, Create an SDN validation environment to test the switch's Control, Data and Management Planes with OFTest (Part 2)
- Steven Lyons
    - User Story #144, Create an SDN validation environment to test the switch's Control, Data and Management Planes with Ryu (Part 2)

Sprint 5

After discussion, the velocity of the team were estimated to be 40.

The product owner chose the following user stories to be done during the next sprint. They are ordered based on their priority.

- User Story #142, Create an SDN validation environment to test the switch's Control, Data and Management Planes with OFTest (Part 3)
- User Story #145, Create an SDN validation environment to test the switch's Control, Data and Management Planes with Ryu (Part 3)

The team members indicated their willingness to work on the following user stories.

- Jahkell Lazarre
    - User Story #142, Create an SDN validation environment to test the switch's Control, Data and Management Planes with OFTest (Part 3)
- Steven Lyons
    - User Story #145, Create an SDN validation environment to test the switch's Control, Data and Management Planes with Ryu (Part 3)

SYSTEM DESIGN

This section contains information on the design decisions that went into this project. The architecture patterns are outlined and explained. The entire system is shown in a package diagram and the subsystems are explained. Finally, the design patterns used in the project are discussed.

Architectural Patterns

Although *SwitchTester* has a relatively simple design as its target goal is not inherently complex, the software was designed with modularity heavily in mind. This lends itself to the component-based architectural style. *SwitchTester* is an application that relies on the *CoreTester* framework. *CoreTester* exposes core functionalities common to all concrete switch validation

software. *OFTTester* and *RyuTester* are both logically-independent switch validation software that implement these core functions, providing a clean and intuitive interface. All throughout, *SwitchTester* is designed so that exportability is simple and easy. Although the system can be used as an application, these core functionalities can be exported and used as a module in external software. This renders *SwitchTester* as a reusable component, allowing custom code to be added for validating OpenFlow switches.

System and Subsystem Decomposition

As mentioned, *SwitchTester* is divided into two subsystems *OFTTester* and *RyuTester*, which both rely on *CoreTester* for configuration and function implementation.

OFTTester Subsystem: Uses *OFTest* testing framework for executing test cases and outputting results.

RyuTester Subsystem: Uses *Ryu* testing framework for executing test cases and outputting results.

Deployment Diagram

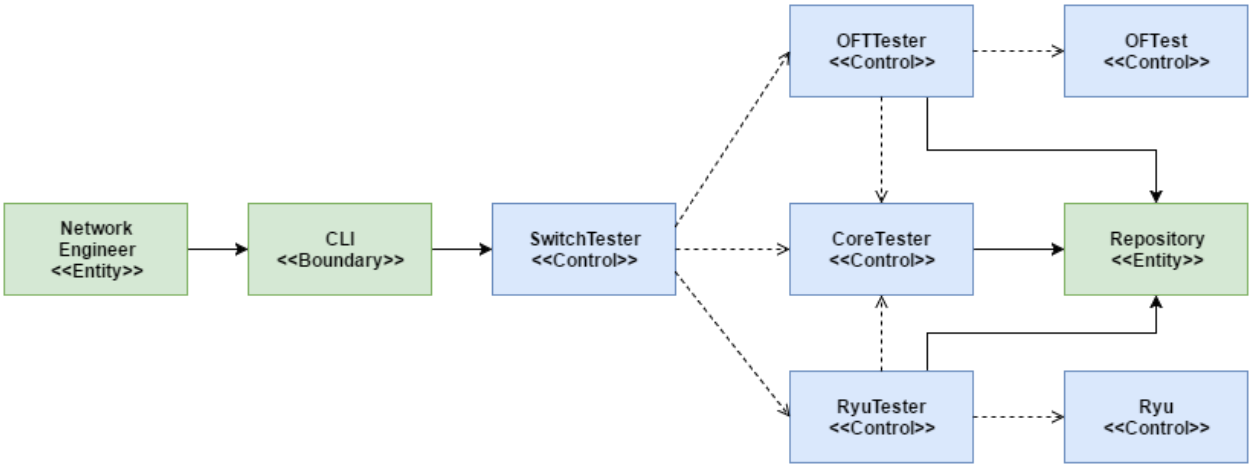

Figure 1: Minimal Class Diagram

Design Patterns

The following design patterns were chosen for the system:

- Strategy: This design was chosen due to the two different strategies for testing, which both follow the relatively same pattern, but need to be picked based on the context of the input given (***OpenFlow*** version).
- Adapter: This design was chosen due to the use of the pre-existing ***Ryu*** and ***OFTest*** testing frameworks that would need to be used.

SYSTEM VALIDATION

Unit Tests:

U-121-1

- Purpose: To test correctness of generated log that uses output that would be generated from OFTest.
- Precondition: The sample output (XML files like the ones generated from running tests in OFTest) is contained in a directory ready to be read and parsed.
- Expected Result:


```
{
   "test-suite":[
      {
        "test-case-name":"sample_class1.SampleProtocol2",
         "result":"error",
         "num-errors":1,
         "num-failures":0,
         "run-time-sec":4.079
      },
      {
         "test-case-name":"sample_class1.SampleProtocol1",
         "result":"ok",
         "num-errors":0,
         "num-failures":0,
         "run-time-sec":2.079
      },
      {
         "test-case-name":"sample_class14.SampleProtocol32",
         "result":"fail",
         "num-errors":1,
```

```
        "num-failures":1,
        "run-time-sec":4.079
      },
      {
        "test-case-name":"sample_class14.SampleProtocol36",
        "result":"fail",
        "num-errors":1,
        "num-failures":1,
       "run-time-sec":7.0
      }
   ],
   "total-run-time-sec":17.237
}
```


- Actual Result:


```
{
   "test-suite":[
      {
        "num-failures":0,
        "test-case-name":"sample_class1.SampleProtocol2",
        "result":"error",
        "num-errors":1,
        "run-time-sec":4.079
      },
      {
        "num-failures":0,
        "test-case-name":"sample_class1.SampleProtocol1",
        "result":"ok",
        "num-errors":0,
        "run-time-sec":2.079
      },
      {
        "num-failures":1,
        "test-case-name":"sample_class14.SampleProtocol32",
        "result":"fail",
        "num-errors":1,
        "run-time-sec":4.079
      },
      {
```

```
        "num-failures":1,
        "test-case-name":"sample_class14.SampleProtocol36",
        "result":"fail",
        "num-errors":1,
        "run-time-sec":7.0
      }
   ],
   "total-run-time-sec":17.237
}
```


- Status: PASSED

U-141-1

- Purpose: To test correctness of generated JSON report.
- Precondition: The sample output (XML files like the ones generated from running tests in OFTest) is contained in a directory ready to be read and parsed.
- Expected Result:


```
{
   "test-suite":[
      {
         "num-failures":0,
         "test-case-name":"basic.DescStatsGet",
         "result":"ok",
         "num-errors":0,
         "run-time-sec": -
      },
      {
         "num-failures":0,
         "test-case-name":"basic.BadMessage",
         "result":"error",
         "num-errors":1,
         "run-time-sec": -
      },
      {
         "num-failures":0,
         "test-case-name":"basic.PacketIn",
         "result":"error",
         "num-errors":1,
```

```
    "run-time-sec": -
  },
  {
    "num-failures":0,
    "test-case-name":"basic.PortConfigMod",
    "result":"error",
    "num-errors":1,
    "run-time-sec": -
  },
  {
    "num-failures":0,
    "test-case-name":"basic.TableStatsGet",
    "result":"error",
    "num-errors":1,
    "run-time-sec": -
  },
  {
    "num-failures":0,
    "test-case-name":"basic.PacketOutMC",
    "result":"ok",
    "num-errors":0,
    "run-time-sec": -
  },
  {
    "num-failures":0,
    "test-case-name":"basic.FlowMod",
    "result":"error",
    "num-errors":1,
    "run-time-sec": -
  },
  {
    "num-failures":0,
    "test-case-name":"basic.PacketOut",
    "result":"ok",
    "num-errors":0,
    "run-time-sec": -
  },
  {
    "num-failures":0,
```

```
        "test-case-name":"basic.FlowStatsGet",
        "result":"error",
        "num-errors":1,
        "run-time-sec": -
      },
      {
        "num-failures":0,
        "test-case-name":"basic.Echo",
        "result":"error",
        "num-errors":1,
        "run-time-sec": -
      },
      {
        "num-failures":0,
        "test-case-name":"basic.PortConfigModErr",
        "result":"error",
        "num-errors":1,
        "run-time-sec":  -
      },
      {
        "num-failures":0,
        "test-case-name":"basic.EchoWithData",
        "result":"error",
        "num-errors":1,
        "run-time-sec": -
      },
      {
        "num-failures":0,
        "test-case-name":"basic.PacketInBroadcastCheck",
        "result":"error",
        "num-errors":1,
        "run-time-sec": -
      }
    ]
}
```


- Actual Result:


```
{
  "test-suite":[
```

```
{
  "num-failures":0,
  "test-case-name":"basic.DescStatsGet",
  "result":"ok",
  "num-errors":0,
  "run-time-sec":1.035
},
{
  "num-failures":0,
  "test-case-name":"basic.BadMessage",
  "result":"error",
  "num-errors":1,
  "run-time-sec":20.005
},
{
  "num-failures":0,
  "test-case-name":"basic.PacketIn",
  "result":"error",
  "num-errors":1,
  "run-time-sec":20.004
},
{
  "num-failures":0,
  "test-case-name":"basic.PortConfigMod",
  "result":"error",
  "num-errors":1,
  "run-time-sec":20.008
},
{
  "num-failures":0,
  "test-case-name":"basic.TableStatsGet",
  "result":"error",
  "num-errors":1,
  "run-time-sec":20.005
},
{
  "num-failures":0,
  "test-case-name":"basic.PacketOutMC",
  "result":"ok",
```

```
    "num-errors":0,
    "run-time-sec":1.209
  },
  {
    "num-failures":0,
    "test-case-name":"basic.FlowMod",
    "result":"error",
    "num-errors":1,
    "run-time-sec":20.004
  },
  {
    "num-failures":0,
    "test-case-name":"basic.PacketOut",
    "result":"ok",
    "num-errors":0,
    "run-time-sec":1.234
  },
  {
    "num-failures":0,
    "test-case-name":"basic.FlowStatsGet",
    "result":"error",
    "num-errors":1,
    "run-time-sec":20.004
  },
  {
    "num-failures":0,
    "test-case-name":"basic.Echo",
    "result":"error",
    "num-errors":1,
    "run-time-sec":20.007
  },
  {
    "num-failures":0,
    "test-case-name":"basic.PortConfigModErr",
    "result":"error",
    "num-errors":1,
    "run-time-sec":3.076
  },
  {
```

```
        "num-failures":0,
        "test-case-name":"basic.EchoWithData",
        "result":"error",
        "num-errors":1,
        "run-time-sec":3.988
      },
      {
        "num-failures":0,
        "test-case-name":"basic.PacketInBroadcastCheck",
        "result":"error",
        "num-errors":1,
        "run-time-sec":1.047
      }
   ]
}
```


- Status: PASSED

U-142-1

- Purpose: To test correctness of generated CSV report given an application profile.
- Precondition: The application profile has basic.PacketOut, basic.DescStatsGet, basic.PacketInBroadcastCheck, and basic.EchoWithData in the compatibility list.
- Expected Result:

```
basic.PacketOut,OK
basic.DescStatsGet,OK
basic.PacketInBroadcastCheck,ERROR
basic.EchoWithData,ERROR
```

- Actual Result:

```
basic.PacketOut,OK
basic.DescStatsGet,OK
basic.PacketInBroadcastCheck,ERROR
basic.EchoWithData,ERROR
```

- Status: PASSED

U-142-2

- Purpose: To test correctness of generated JSON report given an application profile.
- Precondition: The application profile has basic.PacketOut, basic.DescStatsGet, basic.PacketInBroadcastCheck, and basic.EchoWithData in the compatibility list.

- Expected Result:


```
{
   "basic.PacketOut":{
      "result":"OK",
      "detail":"none"
   },
   "basic.PacketInBroadcastCheck":{
      "result":"ERROR",
      "detail":"some description"
   },
   "basic.DescStatsGet":{
      "result":"OK",
      "detail":"none"
   },
   "basic.EchoWithData":{
      "result":"ERROR",
      "detail":"some-description"
   }
}
```


- Actual Result:


```
{
   "basic.PacketOut":{
      "result":"OK",
      "detail":"none"
   },
   "basic.PacketInBroadcastCheck":{
      "result":"ERROR",
      "detail":"BCast packet received on port 2"
   },
   "basic.DescStatsGet":{
      "result":"OK",
      "detail":"none"
   },
   "basic.EchoWithData":{
      "result":"ERROR",
      "detail":"Did not complete features_request for handshake"
   }
}
```

- Status: PASSED

U-144-1

Purpose:

A text file consisting of a Ryu test result was used for quick testing of several functions, along with sample profiles that had expected outputs, such as `judge_results` ending in either a `PASS` or `FAIL` situation, as expected of the profile.

Precondition:

Ryu test result has "`act OUTPUT,OK`" in the simplified `csv` file.

sunny.profile:

```
{
    "model":"sunny",
    "of_version":"of13",
    "compatibility": [
        "act OUTPUT"
    ]
}
```

Expected Result:

`sunny PASSED`

Actual Result:

`sunny PASSED`

Status:

Passed

U-144-2

Purpose:

A text file consisting of a Ryu test result was used for quick testing of several functions, along with sample profiles that had expected outputs, such as `judge_results` ending in either a `PASS` or `FAIL` situation, as expected of the profile.

Precondition:

Ryu test result has "`act OUTPUT,OK`" in the simplified `csv` file.

rainy.profile:

```
{
    "model":"rainy",
    "of_version":"of13",
    "compatibility": [
        "act OUTPUT",
        "doesnt exist"
```

```
    ]
}
```

Expected Result:
`rainy FAILED`
Actual Result:
`rainy FAILED`
Status:
Passed

U-145-1
Purpose:
A text file consisting of a Ryu test result was used for quick testing of several functions, along with sample profiles and target switch configurations that had expected outputs, such as `judge_results` ending in either a `PASS` or `FAIL` situation, as expected of the profile.
Precondition:
Ryu test result has "`act OUTPUT,OK`" in the simplified `csv` file. Target switch with name of "`test-sw`" and of-version of "`1.3`".
sunny.profile:

```
{
    "model":"sunny",
    "compatibility": [
        "act OUTPUT"
    ]
}
```

Expected Result:
`sunny: test-sw PASSED`
Actual Result:
`sunny: test-sw PASSED`
Status:
Passed

U-145-2
Purpose:
A text file consisting of a Ryu test result was used for quick testing of several functions, along with sample profiles and target switch configurations that had expected outputs, such as `judge_results` ending in either a `PASS` or `FAIL` situation, as expected of the profile.
Precondition:
Ryu test result has "`act OUTPUT,OK`" in the simplified `csv` file. Target switch with name of "`test-sw`" and of-version of "`1.3`"

rainy.profile:

```
{
    "model":"rainy",,
    "compatibility": [
        "act OUTPUT",
        "doesnt exist"
    ]
}
```

Expected Result:

```
rainy: test-sw FAILED
```

Actual Result:

```
rainy: test-sw FAILED
```

Status:

Passed

Integration Tests

I-142-1

- Purpose: To test correctness of generated CSV report given an application profile once integrated with RyuTester Since these applications are logically independent programs with implementing a common framework (CoreTester), the tests remain the same as the unit tests. This integration test is just to ensure that the OFTTester still outputs the correct result when integrated with RyuTester.
- Precondition: The application profile has basic.PacketOut, basic.DescStatsGet, basic.PacketInBroadcastCheck, and basic.EchoWithData in the compatibility list.
- Expected Result:

  basic.PacketOut,OK
  basic.DescStatsGet,OK
  basic.PacketInBroadcastCheck,ERROR
  basic.EchoWithData,ERROR

- Actual Result:

  basic.PacketOut,OK
  basic.DescStatsGet,OK
  basic.PacketInBroadcastCheck,ERROR
  basic.EchoWithData,ERROR

- Status: PASSED

I-142-2

- Purpose: To test correctness of generated JSON report given an application profile once integrated with RyuTester Since these applications are logically independent programs with implementing a common framework (CoreTester), the tests remain the same as the unit tests. This integration test is just to ensure that the OFTTester still outputs the correct result when integrated with RyuTester.
- Precondition: The application profile has basic.PacketOut, basic.DescStatsGet, basic.PacketInBroadcastCheck, and basic.EchoWithData in the compatibility list.
- Expected Result:


```
{
   "basic.PacketOut":{
      "result":"OK",
      "detail":"none"
   },
   "basic.PacketInBroadcastCheck":{
      "result":"ERROR",
      "detail":"some description"
   },
   "basic.DescStatsGet":{
      "result":"OK",
      "detail":"none"
   },
   "basic.EchoWithData":{
      "result":"ERROR",
      "detail":"some-description"
   }
}
```


- Actual Result:


```
{
   "basic.PacketOut":{
      "result":"OK",
      "detail":"none"
   },
   "basic.PacketInBroadcastCheck":{
      "result":"ERROR",
      "detail":"BCast packet received on port 2"
   },
   "basic.DescStatsGet":{
      "result":"OK",
```

```
      "detail":"none"
   },
   "basic.EchoWithData":{
      "result":"ERROR",
      "detail":"Did not complete features_request for handshake"
   }
}
```


- Status: PASSED

I-143-1

Purpose:

Run Ryu inside of a MiniNet simulated network and get the test report.

Precondition:

Ryu and MiniNet are installed and setup.

Expected Result:

A Ryu test report

Actual Result:

A Ryu test report

Status:

Passed

I-144-1

Purpose:

With a configuration, profile, Ryu, and switches being used, a result can be concluded after running Ryu tests and checking those tests against the profile.

Precondition:

Ryu is installed and run through ryu_manager and getting an expected output (or at least an expected output pattern).

Expected Result:

sunny PASSED

Actual Result:

sunny PASSED

Status:

Passed

I-145-1

Purpose:

With a configuration, application profile, target switch configuration, Ryu, and switches being used, a result can be concluded after running Ryu tests and checking those tests against the profile and target switch.

Use case utilizes Ryu to test an OpenFlow 1.3 switch. Using RyuTester, a module developed in the previous sprint for testing an application profile against target switches using Ryu.

Precondition:

Ryu is installed and run through ryu_manager and getting an expected output (or at least an expected output pattern).

SwitchTester is able to access and use RyuTester.

Expected Result:

sunny: test-sw PASSED

Actual Result:

sunny: test-sw PASSED

Status:

Passed

GLOSSARY

OpenFlow: A communications protocol that gives access to the forwarding plane of a network switch or router over the network and enables network controllers to determine the path of network packets across a network of switches.

Software-defined Networking (SDN): An approach to computer networking that allows network administrators to manage network services through abstraction of lower-level functionality.

Network Engineer: Designs and implements computer networks.

OFTest: A Python-based OpenFlow switch test framework and collection of test cases.

Ryu: A component-based software-defined networking framework. It provides software components with well-defined APIs that make it easy for developers to create new network management and control applications.

APPENDIX

Appendix A - UML Diagrams

User Story #126: Implement the sample topology using Mininet and OESS.

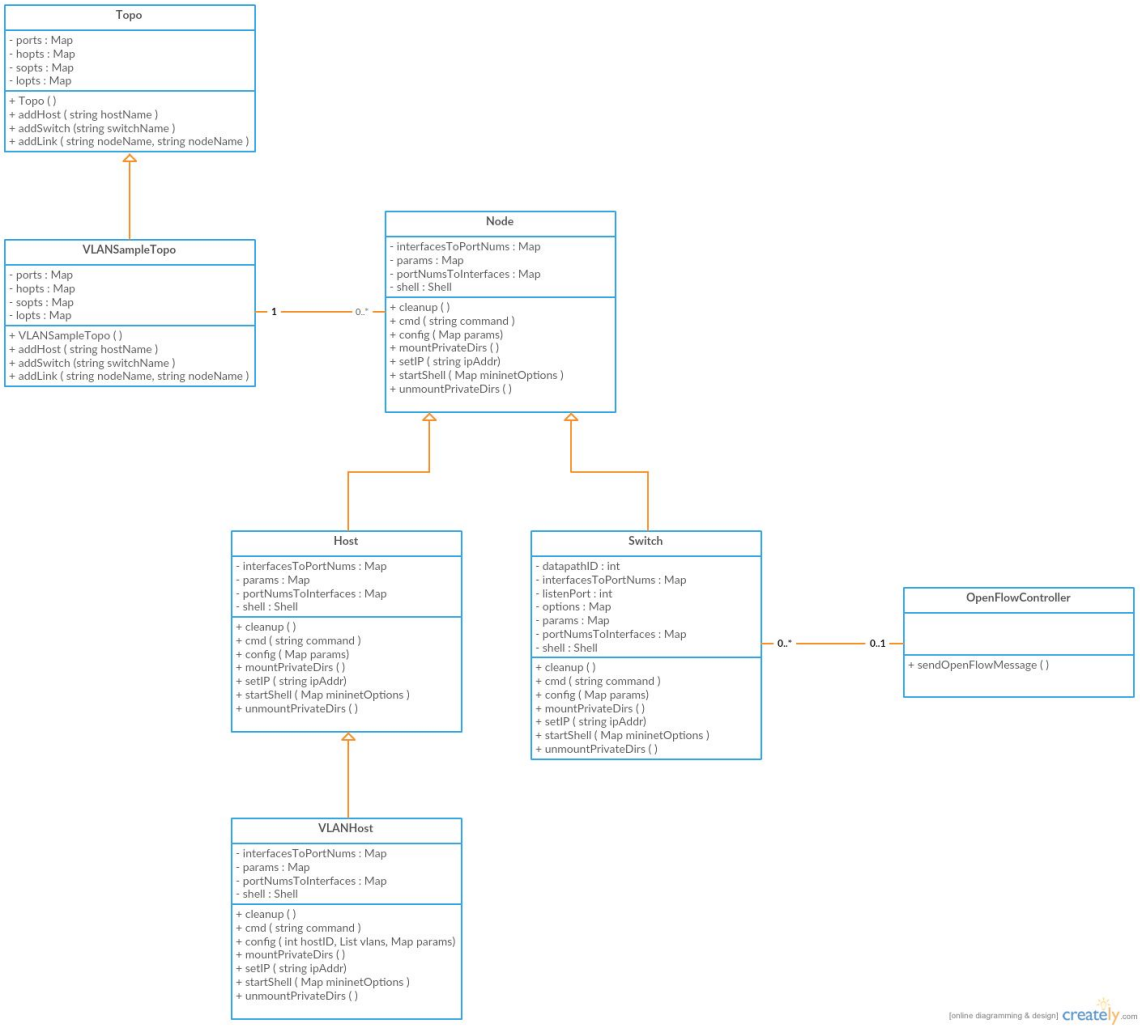

Figure CD-126-1

User Story #121: Create an SDN validation environment to test the switch's Control, Data and Management Planes with OFTest (Part #1)

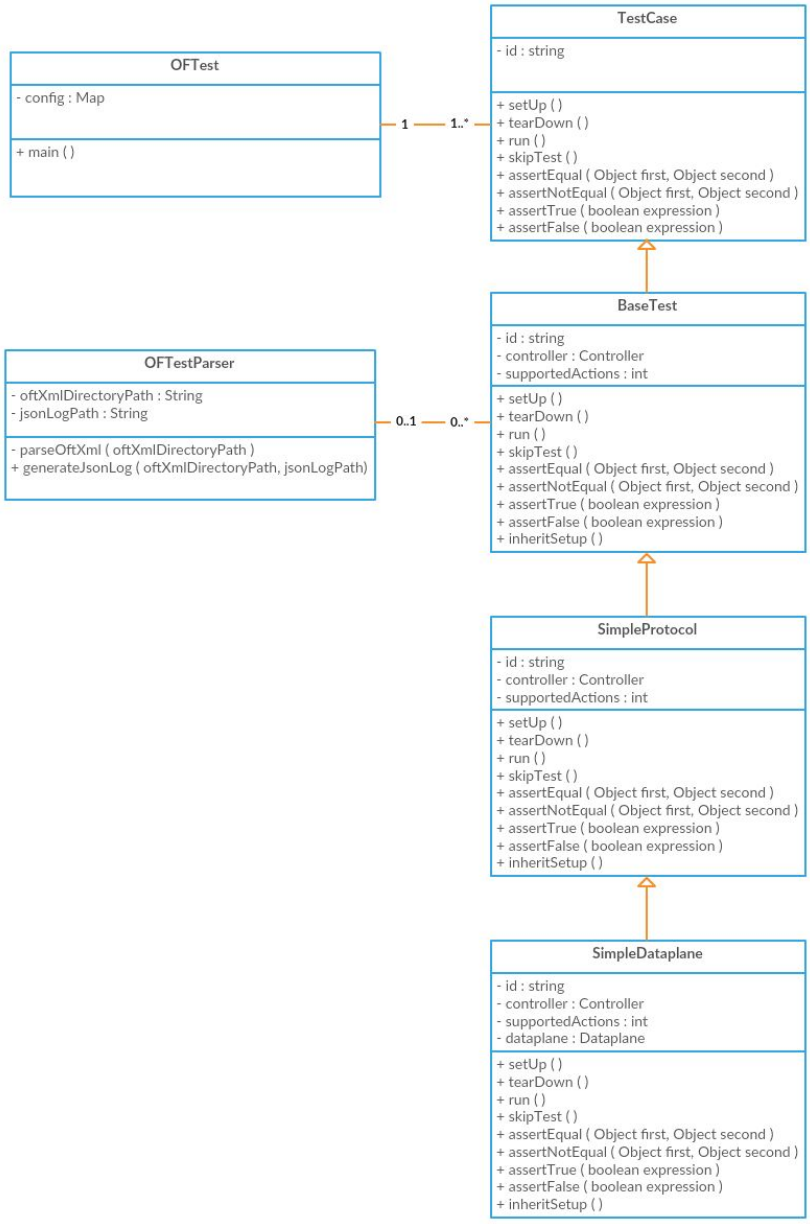

Figure CD-121-1

User Story 141: Create an SDN validation environment to test the switch's Control, Data and Management Planes with OFTest (Part #2)

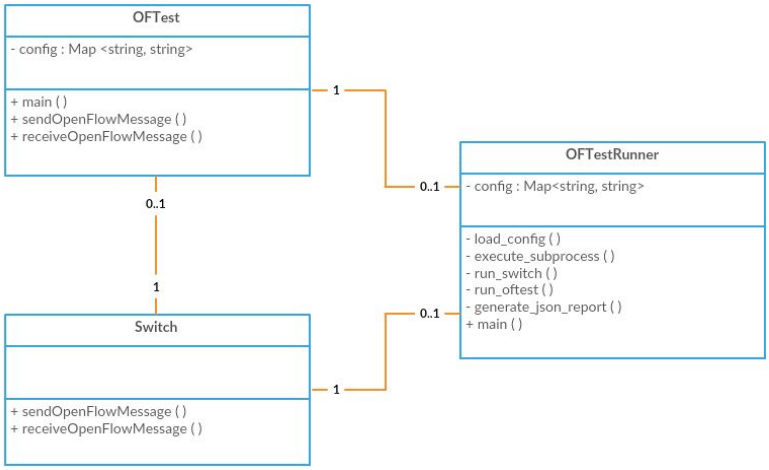

Figure CD-141-1

User Story 142: Create an SDN validation environment to test the switch's Control, Data and Management Planes with OFTest (Part #2)

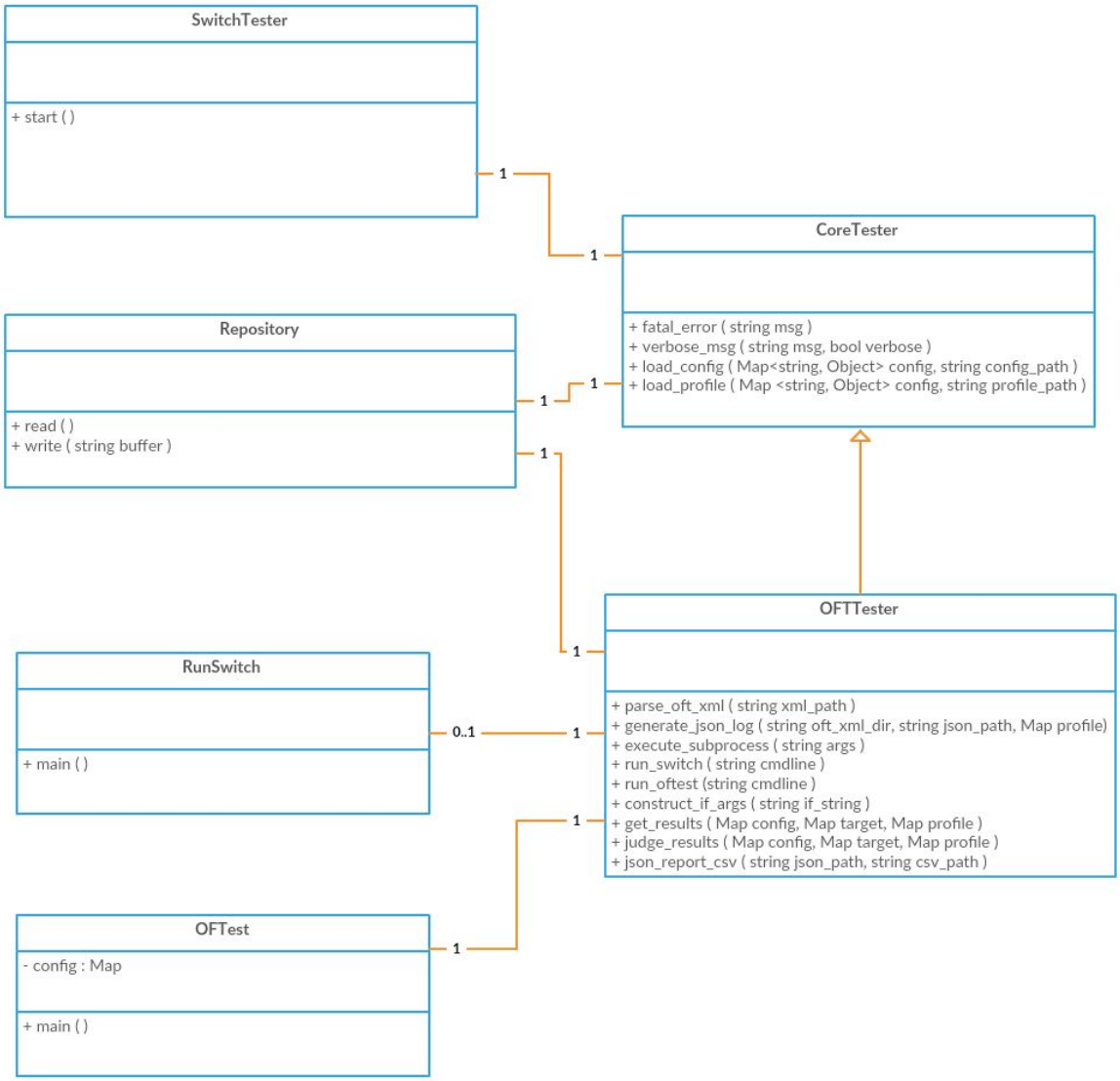

Figure CD-142-1

User Story# 143: Create an SDN validation environment to test the switch's Control, Data and Management Planes with Ryu (Part 1)

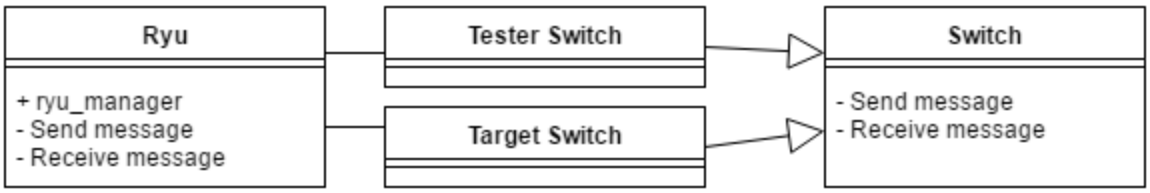

Figure CD-143-1

User Story# 144: Create an SDN validation environment to test the switch's Control, Data and Management Planes with Ryu (Part 2)

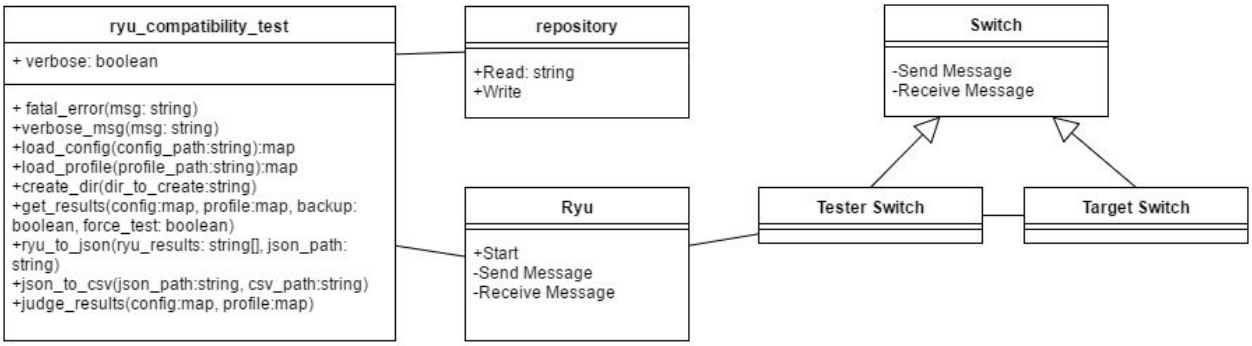

Figure CD-144-1

User Story #145: Create an SDN validation environment to test the switch's Control, Data and Management Planes with Ryu (Part 3)

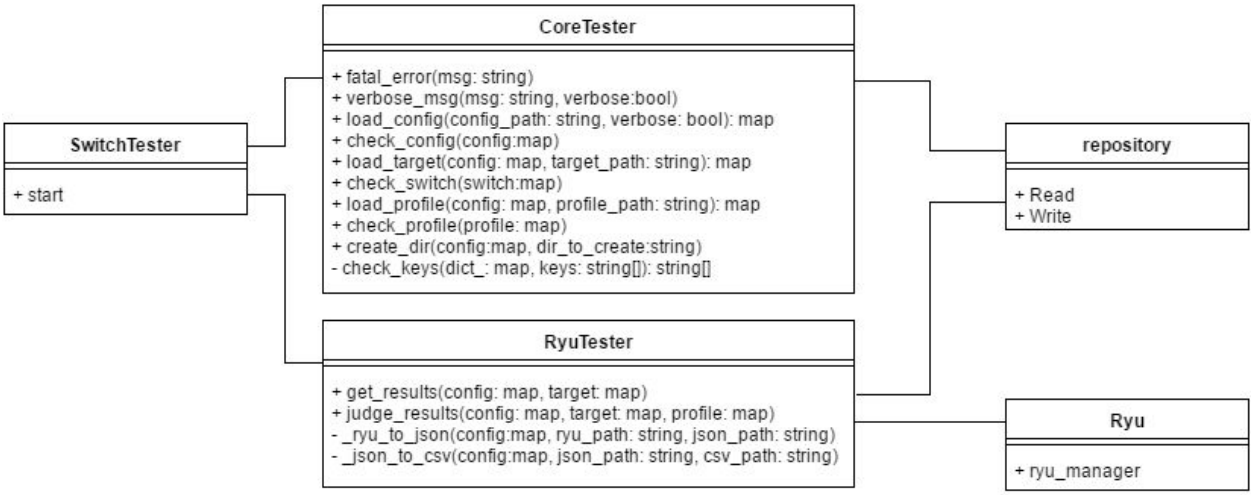

Figure CD-145-1

Appendix B - User Interface Design

```
mininet@mininet-vm:~/switch_tester$ sudo python SwitchTester.py -v -t target.json
Config: Loading config file /home/mininet/.switch_tester.conf.
Config: /home/mininet/.switch_tester.conf loaded successfully.
Target: Loading target switch file target.json.
Target: target.json loaded successfully
WARNING: No route found for IPv6 destination :: (no default route?)

Running tests...
----------------------------------------------------------------------
  basic.EchoWithData ... ERROR (4.002s)
  basic.PacketInBroadcastCheck ... OK (3.028s)
  basic.DescStatsGet ... OK (1.050s)
  basic.PacketOutMC ... OK (1.206s)
```

Figure UI-142-1 - Running SwitchTester using OFTest and verbose option.

```
$ python SwitchTester.py -p ~/sunny.profile -target ~/test.target
sunny: test-sw PASSED
$
```

Figure UI-145-1 - Running SwitchTester using Ryu. Passing result.

Appendix C - Sprint Review Reports

Sprint 1

September 9, 2016
Attendees: Steven Lyons, Juan Medina, Jahkell Lazarre, Steven Gamatan
Start time: 2:04pm
End time: 2:28pm

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners:

- User Story #117
  As a project member I need to perform preliminary research on SDN/OpenFlow as it is necessary to understand and work on the project.

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Spring Planning meeting.

- N/A

Sprint 2

September 23, 2016
Attendees: Steven Lyons, Jahkell Lazarre
Start time: 2:03 pm
End time: 2:28 pm

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners:

- User Story #126
  As a project member I need to simulate the sample topology so that I can show that my testing environment and understanding works properly for future development and testing.

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Spring Planning meeting.

- None

Sprint 3

Attendees: Steven Lyons, Jahkell Lazarre
Start time: 2:10 pm
End time: 2:15 pm

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners:

- N/A

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Spring Planning meeting.

- User Story #125, Add visibility of flows managed by ONOS to Zabbix

Notes:

- Priorities of user stories changed mid-sprint

Sprint 4

Attendees: Steven Lyons, Jahkell Lazarre
Start time: 2:10 pm
End time: 2:15 pm

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners:

- User Story #121 – Create an SDN validation environment to test the switch's Control, Data and Management Planes with OFTest (Part 1)
- User Story #143 – Create an SDN validation environment to test the switch's Control, Data and Management Planes with Ryu (Part 1)

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Spring Planning meeting.

- None

Sprint 5

Attendees: Steven Lyons, Jahkell Lazarre
Start time: 2:03 pm
End time: 2:19 pm

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners:

- User Story #141, Create an SDN validation environment to test the switch's Control, Data and Management Planes with OFTest (Part 2)
- User Story #144, Create an SDN validation environment to test the switch's Control, Data and Management Planes with Ryu (Part 2)

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Spring Planning meeting.

- None

Sprint 6

Attendees: Steven Lyons, Jahkell Lazarre
Start time: 2:03 pm
End time: 2:28 pm

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners:

- User Story #142, Create an SDN validation environment to test the switch's Control, Data and Management Planes with OFTest (Part 3)
- User Story #145, Create an SDN validation environment to test the switch's Control, Data and Management Planes with Ryu (Part 3)

REFERENCES

Mininet Development Team (2016) Mininet Overview
http://mininet.org/overview/,

OFTest Development Team (2016) OFTest GitHub
https://github.com/floodlight/oftest,

OFTest Development Team (2016) Project Floodlight
http://www.projectfloodlight.org/oftest/,

Ryu Development Team (2016) Ryu GitHub IO
https://osrg.github.io/ryu/,