**Sentinel UI Report**

Omar Ismail

Stephen Palagi

Alyssa Seifert

Josh Tranfaglia

**Objective and Motivation of the project**

As programmers, we want to build tools that make our work easier and more efficient. Mutation testing is one of the tools programmers use to further enhance the codebase. Current mutation tools are tedious in that they require multiple steps to find out the mutants’ states and what their mutation is. Typically it involves a number of steps: 1) running the mutation testing tool, 2) looking at a file, usually csv, of the list of mutants and their corresponding state, 3) running a diffing script with a mutant ID to display the actual mutation. Doing it many times becomes tedious and repetitive.

Our motivation was to streamline this task entirely. The ultimate goal was to create a tool that looks at all the mutations and allows the programmer to investigate specific ones to determine what changes should be made to the code base. We did this by creating a User Interface, with a command-line tool, to represent all this information such that the programmer can visually examine all the mutants, their states, and their mutations, all in one place, with additional helpful functionality including filtering by state. We believe this is much simpler than manually running a show\_mutation script, passing in the Mutant ID, every time you want to investigate a mutant. We are confident this tool could be used by programmers and make their jobs more efficient.

**Assumptions & How to Use the Project**

There are two main components to this project: 1) Command-line interface (CLI), and 2) User Interface. The CLI consists of a python script that requests two pieces of information: 1) the path to the file containing mutation results (currently only in csv), and 2) the path to a shell script that gives the difference between the source code and the mutant. This python script then takes in all the information on the mutants, from the existing status csv and the diffing shell script, and aggregates the output into a separate output.csv file. The programmer then opens up the index.html file, where he can upload the output.csv file. The User Interface will parse this information and display all the appropriate information. Here are the fully fleshed out steps for using this project:

Command Line Interface Prerequisites

* Must have run the mutation script (“./mutation.sh”, “ant mutant” or any other command) and produced a csv with mutant IDs and mutant statuses
* Must have a show\_mutant shell script that takes the Mutant ID as a single argument, and gives the difference between the source and the mutant
* Must include the “visualize\_mutations.py” file inside your working directory.

Executing Python Script

* Execute python script from the command line: “python visualize\_mutations.py”
* Enter file type (only “csv” for now)
* Enter the path to the Mutant Killed CSV file when prompted.
* Enter the path to the script that details the difference between the source and the mutant when prompted

Results and Web Component

* A CSV file will be produced by the name of “aggregated\_mutant\_output.csv”
* Open “index.html”
* Select “Choose File” and import the “aggregated\_mutant\_output.csv”
* Explore!

**Design Decisions**

We designed the program to be extensible in both the python script as well as the UI. In the python script, we currently only accept “csv” files of the mutants, but made it extensible such that if a new file was desired, all a programmer needed to do was come in and add a new class to parse that new file type. The same approach was done in the UI, so it could easily be able to parse many other formats, from JSON to XML. We tried to separate the drawing logic from the parsing logic, with the intent to follow appropriate design patterns based on the premise of separation of concerns. We chose to use a library when dealing with the csv files as it provided a robust toolset for specifically working with that file type and allowed for even more extensibility in where we draw the files from if were to expand upon that in the future.

**Future Work**

There are a number of ideas we would like to implement in the future given the time. First and foremost is to make the entire process controllable through the HTML User Interface, streamlining the process even further and moving away from the Command Line. This entails running the mutation tool, the python script, the parser, and uploading the final output file automatically behind the scenes. Secondly, we would like to include more information in the UI. Currently we only show the various states which the user can filter. In the future, we could enhance the display functionality to include the contents of the source code, or the surrounding context of each mutant.