Muğla Sıtkı Koçman University

**Senior Design Project I**

*Stateflow Mutation Testing*

Analysis & Design Report

Halim Abdurrahman Ceylan

Supervisor: Zeynep Filiz Eren

January 19, 2023

**Contents**

**1 Introduction 3**

**2 Motivation 3**

**3 Literature Review 3**

**4 Method 3**

**5 Future Work 3**

**6 References 3**

Stateflow Mutation Testing

# Introduction

Mutation testing is like a quality check for your software tests. It involves making small, intentional changes to your program and then running your tests to see if they catch these changes. The goal is to figure out how good your tests are at finding potential mistakes or bugs in your code. Stateflow is a product that provides a graphical language that includes state transition diagrams, flow charts, state transition tables, and truth tables. You can use Stateflow to describe how MATLAB algorithms and Simulink models react to input signals, events, and time-based conditions.

# Motivation

Testing in software engineering is like checking homework before submitting it. It helps find and fix mistakes early, making the final product better. Testing ensures the software works well, is secure, and meets standards. Mutation testing is a specialized technique in testing that intentionally makes small changes to the code, allowing us to see if our tests can catch these changes. It's like intentionally inserting errors into the homework to check if our proofreading catches them.

# Literature Review

In software testing, mutation testing is like a super tool that checks how good a test suite is. It does this by making small changes in the source code and seeing what happens in the test results. The idea is to pretend there are problems in the code and see if the test suite catches them. People have been studying mutation testing since the late 1970s, thanks to Richard Lipton and others. It's become a big deal in figuring out how good test suites are and finding ways to make testing in software development even better.

Now, our research is all about using mutation testing in Matlab Simulink and Stateflow libraries. Specifically, we're looking at how well the test suites handle changes in Stateflow signal statements, like logical operations, math operations, and so on. We want to see if the test suites are flexible enough in these environments. By purposely making changes in important parts of Stateflow models, we're figuring out how good the test suites are at catching these changes. This research not only helps us understand how Stateflow deals with these changes but also gives us ideas on how to make testing practices better in Matlab Simulink and Stateflow.

| Mutation Test | Mutation Test |
| --- | --- |
| Simulink | Simulink & Stateflow |

# Method

1. **Selecting a Stateflow Project:** Choose an existing Stateflow project available on the web that suits the scope of your testing and research.
2. **Understanding the Project:**Familiarize yourself with the chosen Stateflow project, understanding its structure, components, and functionality.
3. **Writing a Testing Script:**Develop a script tailored to test the chosen Stateflow project. This script should include instructions to simulate various scenarios, interactions, or inputs that the Stateflow model might encounter in real-world usage.
4. **Executing the Script:**Run the testing script on the Stateflow project, allowing it to interact with the model and collect relevant data and results.
5. **Analyzing Test Results:**Analyze the results obtained from the script execution. Identify any unexpected behaviors, errors, or areas of improvement within the Stateflow project.
6. **Making Improvements:**Based on the analysis of test results, make necessary improvements or fixes to the Stateflow project. This may involve addressing identified issues, optimizing performance, or enhancing specific functionalities.
7. **Script Refinement:**Refine the testing script to incorporate any adjustments made to the Stateflow project. Ensure that the script is updated to cover new test cases or scenarios.
8. **Iterative Testing and Improvement:**Repeat the testing process iteratively, running the script, analyzing results, making improvements, and refining the script until the Stateflow project meets the desired level of robustness and functionality.

# Future Work

Enhance performance, expand testing scenarios, adapt to new changes, and implement updates, bug fixes, and other improvements.

# References

Pill, Ingo et al. “SIMULTATE: A Toolset for Fault Injection and Mutation Testing of Simulink Models.” *2016 IEEE Ninth International Conference on Software Testing, Verification and Validation Workshops (ICSTW)* (2016): 168-173.

Gregg Rothermel and Mary Jean Harrold. 1997. A safe, efficient regression test selection technique. ACM Trans. Softw. Eng. Methodol. 6, 2 (April 1997), 173–210.