Mutode: Generic JavaScript and Node.js Mutation Testing Tool

Diego Rodríguez Baquero
d.rodriguez13@uniandes.edu.co - 201223538

Adviser: Mario Linares Vásquez
m.linaresv@uniandes.edu.co

I. Topic

Mutode: Generic JavaScript and Node.js Mutation Testing Tool for npm based programs, applications, packages and modules.

II. Introduction

Mutation testing is a technique in which faults (mutants) are inserted into a program or application to assess its test suite effectiveness. It works by inserting the mutant and running the program's test suite to identify if the mutant is detected (killed) or not (survived). Although computationally expensive, it has proven to be an effective method to assess application test suites [1]. Statement and branch coverage are not enough [2][3] and studies have shown that mutation testing is more effective than data flow testing [4].

Many mutation testing frameworks and tools have been built for the various programing languages, covering general programming mutation operators as well as specific mutation operators, applicable only to each one of the languages. There are various tools in the literature and practice. For the Java language, we found tools such as PIT [5], Javalanche [6], Bacterio [7] and Judy [8]. For the C language we found MILU [9]. For the C++ language we found Mull [10] and Mutate++ [11]. For C# we found VisualMutator [12], NinjaTurtles [13] and NinjaTurtlesMutator [14]. For the Python language we found Cosmic Ray [15], Mutmut [16] and MutPy [17]. For the Ruby language there is mutant [18]. For the more web-focused language PHP, we found HumBug [19], Infection [20]. For the Android ecosystem we found MDroid+ [21]. However, very few tools have been built for the JavaScript language, more specifically, there's a lack of tools for the Node.js runtime and npm based applications.

Node.js is an asynchronous, event driven JavaScript runtime design to run any type of application, even backend services with I/O access to virtual memory, network and disk. It's based in Chromium's V8 JavaScript engine and it is fully open source and community driven.

npm is the package manager for Node.js, built as an open source project in 2009 and later registered as a company in 2014. It's also an ever growing registry of open source modules built for the JavaScript, Node.js and Web ecosystems [22]. The over 600,000 packages hosted in npm are downloaded over 4 billion times per week [23]. More and more software is published in npm every day, representing a huge opportunity to share code and solutions, but also to share bugs and faulty software.

We will analyze prior work for mutation operators in JavaScript and Node.js and propose a new framework which leverages npm package ecosystem to build a generic tool that allows easy, zero-configuration mutation testing on

current generation applications using any testing framework. By generic we mean that it will support any type of test suite that runs with the `npm test` command, without plugins or configuration. The approach will be implemented under the name Mutode and released as an Open Source module on the npm registry. It's important to note that it is assumed that the application to be mutated has an automated test suite. Mutode generates mutants and run the test suite without knowledge of what the tests are and which are executed.

Finally, we will empirically evaluate Mutode effectiveness by running it on the top npm modules with automated test suites.

III. Prior work

In the literature, we found Mutandis [24], a framework built in Java, focused on browser JavaScript applications. It utilizes Mozilla's Rhino AST parser to mutate the application and intercepts network requests to analyze their content. If a test suite is available for a given application, it's used to profile it. This tool, however, wasn't made for Node.js applications and npm modules. Besides the initial work 5 years ago, no work or maintainance has been done to the library since. There is also a lack of documentation in its GitHub repository and instructions on how to use it.

In the practice, the main mutation testing framework available for JavaScript and Node.js is Stryker [25]. An open source framework available on the code hosting platform GitHub. It offers over 30 mutation operators and the availability to run tests with Karma, Mocha or other frameworks through a custom built plugin. It requires configuration and doesn't support other test frameworks besides Karma and Mocha without a plugin. This represents a room for improvement for a new tool that works out of the box with any test suite. Another tool is mutant [26], however it's unmaintained, only had 5 versions and was considered in the proof-of-concept stage.

There is a big room of opportunity to create a mutation testing tool that targets the latest practices for software development in JavaScript and Node.js, both backend and frontend code; one that works with any type of testing framework; and one which requires zero or minor configuration. We fill this gaps with Mutode and offer a practical solution to make mutation testing as easy as running a command.

IV. Project Goals

Our main goal is to build a generic mutation testing tool that can be used to assess npm-based JavaScript and Node.js applications' test suites.

A. Specific goals

- Build a generic framework that leverages npm package.json definition using the `npm test` command to execute applications' test suites.
- Implement a catalog of mutation operators based on PIT, prior work and new research.
- Empirically evaluate Mutode on the top 20 tested npm packages.

V. Methodology

The project is divided in three phases:

A. Phase I

- Implement the basic generic framework core using basic mutation operators applicable to any programming language.

- Search and review prior work in the topic.
- Implement parallel tests execution.
- Implement basic execution configuration.

B. Phase II

- Implement all generic operators based on PIT.
- Search, review and implement previously defined JavaScript and Node.js specific mutation operators.
- Define and implement new Javascript and Node.js specific mutation operators.
- Create tests to assess the framework and operators efficacy.

C. Phase III

- Empirically evaluate the framework on the top 20 npm modules with automated test suites.
- Optional: Implement a dashboard and generate execution reports.
- Optional: Save execution state to test based on code difference.

Project wide, we will document the implementation process, the code and the results obtained in tests and the empirical evaluation.

VI. Schedule

Based on proposed tasks in the methodology, the following schedule is proposed:

Activity/Week	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
Generic framework	X															
Prior work	X	X				X	X	X								
Parallel execution		X	X	X												
Execution configuration				X	X											
PIT Generic Operators					X	X	X									
JS and Node.js operators in prior work						X	X	X								
New JS and Node.js Operators							X	X	X							
Tests				X	X	X	X	X	X							
Empirical evaluation										X	X					
Dashboard and reports												X	X			
Execution state														X	X	
Documentation	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

VII. Expected Results

At the end of the project, Mutode should be a working, tested and documented JavaScript mutation testing tool. It should have an empirical benchmarking of the top 20 npm modules which have automated test suites and also be published as an npm package accessible to the public.

Expected features

- Allow mutants tests to be executed in parallel, harnessing full CPU and reducing execution time.
- Unit tests to assess the tool's core and mutation operators.
- Dashboard and execution reports.
- Documentation of the tool and its operators.
- Save execution state to improve performance by reducing generated mutants based on code difference only.

References

[1] Jia, Yue, and Mark Harman. "An analysis and survey of the development of mutation testing." *IEEE transactions on software engineering* 37.5 (2011): 649-678.

[2] Andrews, James H., et al. "Using mutation analysis for assessing and comparing testing

coverage criteria." *IEEE Transactions on Software Engineering* 32.8 (2006): 608-624.

[3] Just, René, et al. "Are mutants a valid substitute for real faults in software testing?." *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*. ACM, 2014.

[4] Offutt, A. Jefferson, et al. "An experimental evaluation of data flow and mutation testing." Softw., Pract. Exper. 26.2 (1996): 165-176.

[5] Coles, Henry. “PIT Mutation Testing”. http://pitest.org/

[6] Schuler, David, and Andreas Zeller. "Javalanche: efficient mutation testing for Java." *Proceedings of the the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering*. ACM, 2009.

[7] Mateo, Pedro Reales, and Macario Polo Usaola. "Bacterio: Java mutation testing tool: A framework to evaluate quality of tests cases." *Software Maintenance (ICSM), 2012 28th IEEE International Conference on*. IEEE, 2012.

[8] Madeyski, Lech, and Norbert Radyk. "Judy–a mutation testing tool for Java." *IET software* 4.1 (2010): 32-42.

[9] Jia, Yue, and Mark Harman. "MILU: A customizable, runtime-optimized higher order mutation testing tool for the full C language." *Practice and Research Techniques, 2008. TAIC PART'08. Testing: Academic & Industrial Conference*. IEEE, 2008.

[10] Denisov, Alex, and Stanislav Pankevich. “Mull: Mutation testing system built on top of LLVM”. https://github.com/mull-project/mull

[11] Lohmann, Niels. “Mutate++: C++ Mutation Test Environment”. https://github.com/nlohmann/mutate_cpp

[12] Trzpil, Piotr. “VisualMutator: Mutation testing integrated within the .NET programming environment”. https://visualmutator.github.io/web/

[13] Musgrove, David. “NinjaTurtles: .NET mutation testing”. http://www.mutation-testing.net/

[14] Roussel, Tony. “NinjaTurtlesMutation”. https://github.com/criteo/NinjaTurtlesMutation

[15] Bingham, Austin. “Cosmic Ray: mutation testing for Python”. https://cosmic-ray.readthedocs.io/en/latest/

[16] Hovmöller, Anders. “Mutmut: Mutation testing lib”. https://github.com/boxed/mutmut

[17] Halas, Konrad. “MutPy: mutation testing tool for Python 3.x source code”. https://github.com/mutpy/mutpy

[18] Schirp, Markus.. “Mutant: Mutation testing for Ruby”. https://github.com/mbj/mutant

[19] Brady, Pádraic. “Humbug: Mutation Testing for PHP”. https://github.com/humbug/humbug

[20] Rafalko, Maks. “Infection: PHP Mutation Testing Framework”. https://github.com/infection/infection

[21] Linares-Vásquez, Mario, et al. "Enabling mutation testing for Android apps." *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*. ACM, 2017.

[22] Npm, inc. “About npm”. https://docs.npmjs.com/company/about

[23] Voss, Laurie (seldo). "npm users downloaded 4 billion packages in the last week, representing more than 12 billion package installations i… https://t.co/DmY2ZWPcmj". 08 Feb 2018, 01:45 UTC. Tweet

[24] Mirshokraie, Shabnam, Ali Mesbah, and Karthik Pattabiraman. "Efficient JavaScript mutation testing." *Software Testing, Verification and Validation (ICST), 2013 IEEE Sixth International Conference on*. IEEE, 2013.

[25] Stryker. http://stryker-mutator.github.io/

[26] Hartley, Ben. “Mutant: JavaScript Mutation Testing Framework”. https://github.com/benhartley/mutant