Implementation of effective test automation with instrumented customer experience data

R. kavitha, P. Subha,

***Abstract:** In the B2C & B2B ecommerce arena, the Measurable Business Results (MBR) of an application is its ability to retain customers and its prospects. And in an ephemeral product and services world, customer experience (CX) is a pillar of value creation. A superior customer experience is a means to stay ahead in the competitive environment. The issues that arise on the customer experience has a greater visibility on the smaller social world and is a direct impact to MBR. With all said, a greater priority of resolving such issues with an effective test automation that leverages the CX oracles in automating the test suites is a solution to mitigate the issues around customer experience. The approach involves flooding the test oracles created with the real customer experience data to the test automation suites that cover the 360 degrees of the functional, regression and integration testing of the application.*

Index Terms: Customer experience, test automation.

I. INTRODUCTION

The application testing with respect to functional, regression and integration testing is a continuous process with the discovery of new data set that suits the changes that has been incorporated as a new feature or changes to the existing one. Every time, the data set identification and streamlining the data set for automated testing is a herculean task and often involve manual efforts to make it happen. There are different mechanisms to validate the correctness of the system under test. The approach taken here is the continuous flooding of the test oracles that get generated by the instrumentation mechanism of the application. The application under test is continuously instrumented gathering the data of customer experience that deal with each specific class and methods of the application. In a nutshell, it is the Integration of Technology with Customer Experience with Open Source API & Frameworks towards enhancing a Java/Web Application with better quality using the automated testing with the following modules.

1. Bytecode Instrumentation to trace CX Behavioral and Interaction Data
2. Automated Testing with CX Data

A. THE CHALLENGE OF TODAY'S IT ENVIRONMENT

The following are few challenges that we see as an inherent issue in the testing world

1. Lack of API Testing
2. Lack of Automated Testing
3. Lack of visibility into production applications
4. Difficulty in managing environments

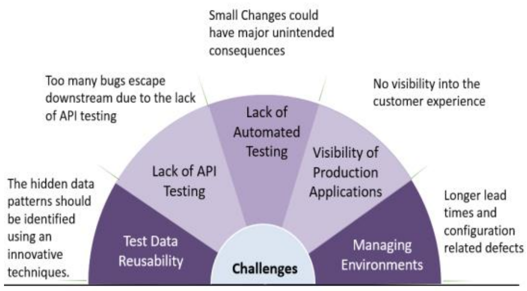

Small Changes could have major unintended consequences
Too many bugs escape downstream due to the lack of API testing
No visibility into the customer experience
Lack of Automated Testing
Lack of API Testing
Visibility of Production Applications
The hidden data patterns should be identified using an innovative techniques.
Longer lead times and configuration related defects
Test Data Reusability
Challenges
Managing Environments

Fig. 1.1 The Challenges of IT Environment

The results of the above challenges when turned out to be an issue in the production system will result in

1. Voice of Customer – The interaction and behavior data of the customer are analyzed only when the application suffers a huge threat with its stability and reliability
2. Negative Scores of Customer Experience is direct impact to the brand.
3. CX Analytics - is the baby of top management for any organization directly proportional to the following

– Conversion and Revenue
– Scorecards & Competitor Ranking's

B. THE INTEL EFFECT

One of the greatest security issue identified in the early Jan of 2018 was the Meltdown & Spectre Attack. The fundamental design flaw of the intel chips was identified after decades of it being the market leader in the industry. The vulnerability was the leakage of kernel memory to any application when introduced with chip-level security bug. The defense mechanism adapted by various operating systems in the Kernel Address Space Layout Randomization (KASLR) is defected by this vulnerability. This vulnerability can be exploited by hackers and malware to read the kernel's memory and the complete system is under threat inclusive of its network.

C. THE VOICE OF THE CUSTOMER

The feedback from the customer about the product or services to different mediums is the voice of the customer (VoC) and materializing this input to the testing arena before it reaches the public forum will benefit to a larger extent. The following are the different source of VoC data.

Manuscript published on 30 June 2019.
* Correspondence Author (s)
Dr.R.kavitha, Department of CSE, Vel Tech Rangarajan Dr.Sagunthala R & D Institute of Science and Technology, Chennai, India.
P.Subha, Department of IT, Sri Sai Ram Institute of Technology, Chennai, India.

1. Social Media Conversations
2. Contact Center Interactions
3. Blogs and news
4. Comments from third party sites and forums

The weightage or priorities set by these inputs to the functional areas should be formulized to the continuous testing process. It is more of the sentiment analysis of the customer and prioritizing the actions accordingly. The automation and plugging in of the sentiment analyzed inputs to the testing life cycle could be another opportunity of study with the proposed system of this project.

D. BRAND TOUCHPOINTS

The brand is a story unfolding across all customer touch points. Defining touchpoints, the customer journey and the experience are all important to a brand's success. More importantly, the factor that govern the brand value is the effectiveness in the testing strategy employed by organizations. In addition to the typical testing strategy employed in general as said below, the complete functional, regression and integration testing leverage the instrumented customer experience data to the continuous automation process will help the brand value.

II. RELATED WORK

VahidGarousiet. al (2017) [1] performed a work in detailing out the process involved in the automated testing and the tools used for achieving the desired results supplemented with case studied and survey results [IEEE 7888399]. This talks about the automation across the software testing process that is inclusive of the test case design, scripting, execution, evaluation, reporting and Test Management and other engineering activities. It talks about the practice regading automation of testing activities other than test execution that involves the prototype tools such as Evosuite, AUSTIN (Augmented Search Based Testing) that perform search based test data generation. These were supported with the survey and case studies of different projects and application under test. AbdelilahSakti, Gilles Pesant et al 2015, [2] performed work on th mechanism for the improved coverage of test automation tools [IEEE 06926828]. It uses three mechanisms. One is the instantiation of the class that is under the scanner of the search based test generation process. It employs the splitting mechanism of the search space and leverages different means of instantiation. The second mechanism is the Diversification Strategy which generates the needed instances by the simplest of the instantiation method and factors the complexity of the instantiation mechanism on the overall complexity of the test automation. Both the first and the second mechanism deal mostly on the means of instantiation and the complexity factor around it. The third mechanism is the seeding strategy of the test data generation process. For each primitive data type, it collects constants from the source code and largely dependent on the source code to provide the test data that it operates on.

III. EXISTING SYSTEM

Performance Engineering tools are capable of providing deep insights to technical artifacts but doesn't have the CX data in to play.UX Applications both commercial and open source are capable of extracting user behavioral data and session playback details for analytics and triaging production issues. Few of the notable products are

1. IBM Tealeaf CX
2. Oracle Cloud CX

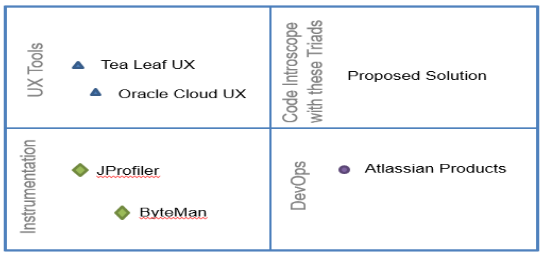


Fig. 3.1 The Magic Quadrant of Proposed Solution

The data extracted would help for manual triaging of issues and of not much useful towards automating the DEVOps cycle and towards fixing the application issues with respect to code, standards and security unless it provides deep insight to the application code, design and the reverse engineered path towards the reproducible defects.

IV. PROPOSED WORK

Use The four-layered architecture that spans across the end user requests till the complete continuous integration using DEVOps workflow is illustrated below. With the data acquisition that enabled through the Java Instrumentation mechanism through the Instrumentation Agent that gets attached to the individual JVM's with appropriate runtime configurations is leveraged for the test data generation for the search based test automation process. The recordings of these were either pushed to the flat file, database, repository or to the elastic search which is more appropriate to the application that may use this in future. The Search Based Test (SBT) Automation framework will be used to leverage the real-time customer data that was captured in the recordings as highlighted below which is taken into the Continuous Integration (CI) cycle that gets triggered by the event of new data flood.

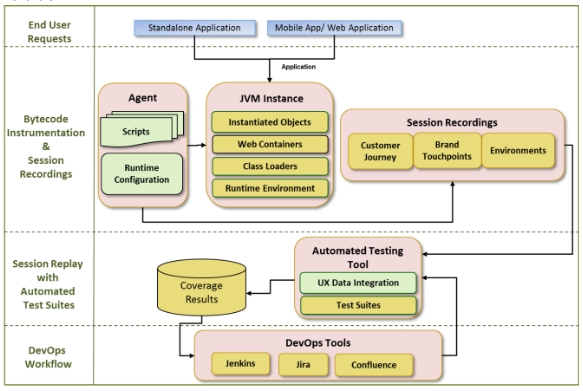


Fig. 4.1 The Architecture Diagram

A. INSTRUMENTATION APPROACH

The rules for the application under test that is required to be traced at line number or the variables nested deep inside the method or any complex conditional statements will be dealt with extending the Rule Generator Adapter of the ByteMan API as shown below.

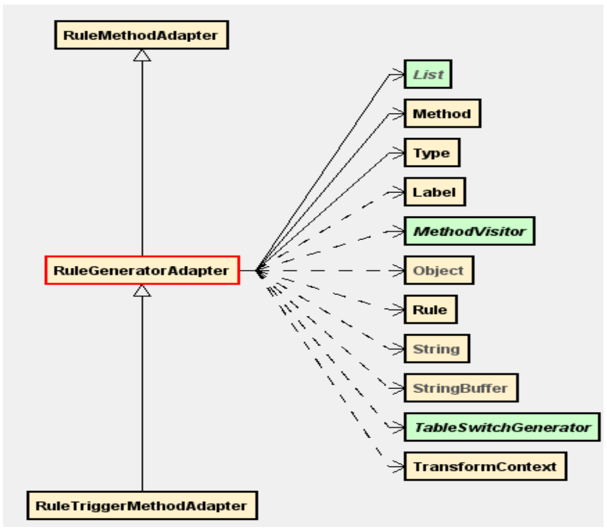


Fig. 4.2 The Class Diagram of RuleGeneratorAdapter

The Main class that does the Instrumentation of the Application under Test.

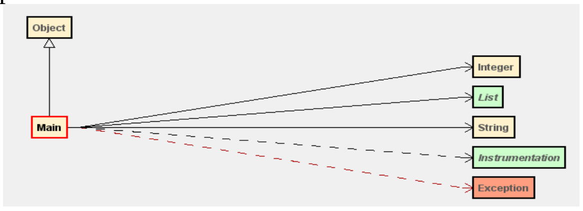


Fig. 4.3 The Class Diagram of Byteman Main Class

B. TEST AUTOMATION APPROACH

Evosuite is the Search Based Test (SBT) Automation framework that is employed for test automation will be extended for the change in CX Instrumented data in the implementation of the Bytecode Instrumentation that it does for SBT.

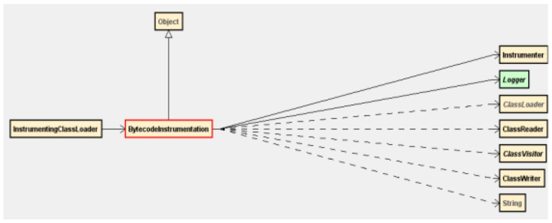


Fig. 4.4 The Class Diagram of bytecodeInstrumentation

The JobExecutor does the sequencing of the test suite generation for the AUT.

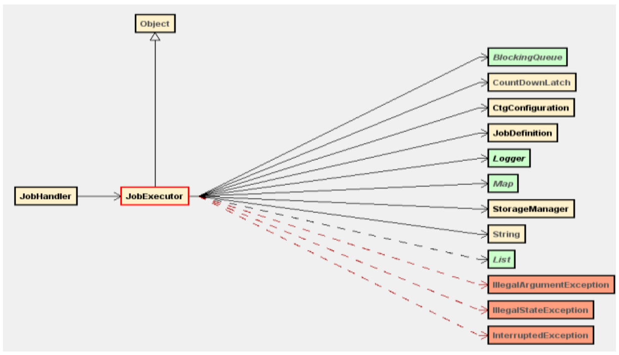


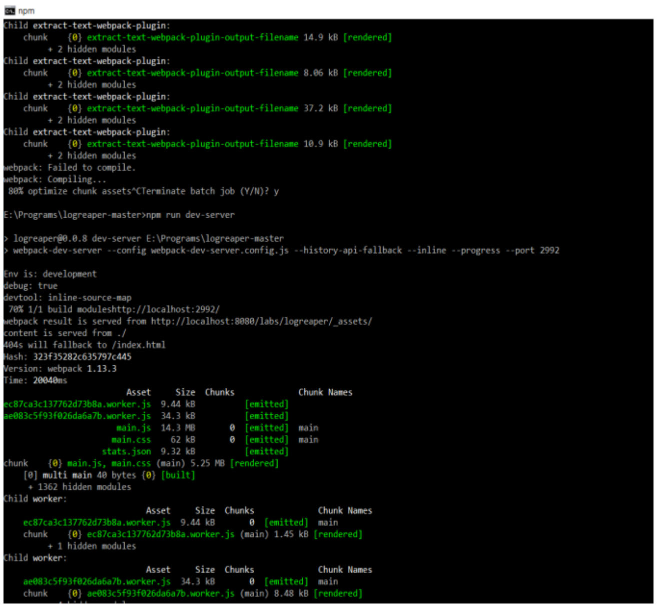
Fig. 4.5 The Class Diagram of bytecodeInstrumentation

C. RULE IMPLEMENTATION – SIMPLIFIED APPROACH

There are different approaches by which the application under test can be scrutinized for the loaded classes and to get them instrumented with the Byteman script using Java Instrumentation mechanism for test data generation. Here is one approach that leverages the Byteman Helper class that could leverage the run-time changes just in time when the operation is performed using the same class loader of the method invocation in the application. To run this script, the helper class and its extension has to be compiled to a jar and set as the instrumentation jar in the application under test. Here are the steps involved. The below is the script to check the compilation of the script. This will report errors when the script has any issues semantically.

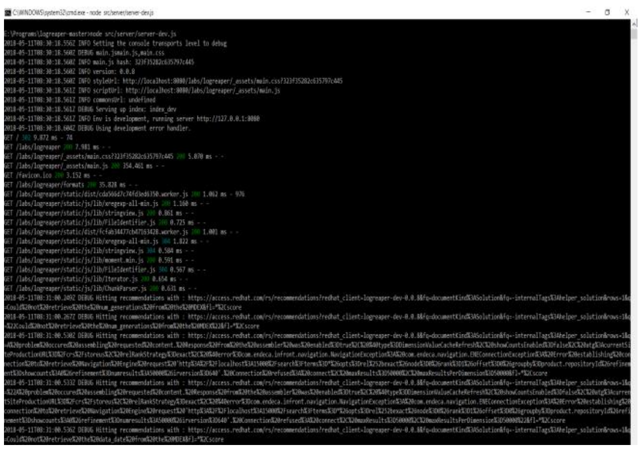
D. THE LOG ANALYSIS OF THE AUT

The Application Under Test (AUT) exihibits no difference in the user interface of the application when the instrumentation agent is attached to its server and the logs of the same is being pushed along with the instrumented data. The below given is the screenshot of the Kibana Query page where we can find the logs of the application. Usually the standard out logs will not be available in the kibana unless they are logged with one of the Kibana log properties (key=value). Having said this, we can get the instrumented data grepped with logging under a key for Kibana or can be manually parsed from the log files to get the instrumented data.



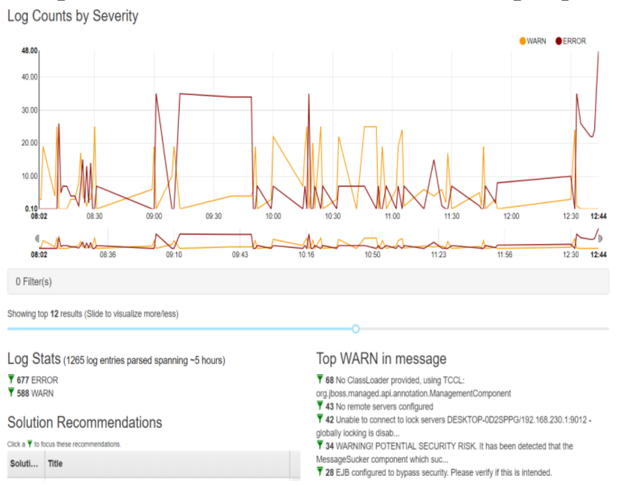
```
npm
Child extract-text-webpack-plugin:
    chunk    {0} extract-text-webpack-plugin-output-filename 14.9 kB [rendered]
         + 2 hidden modules
Child extract-text-webpack-plugin:
    chunk    {0} extract-text-webpack-plugin-output-filename 8.06 kB [rendered]
         + 2 hidden modules
Child extract-text-webpack-plugin:
    chunk    {0} extract-text-webpack-plugin-output-filename 37.2 kB [rendered]
         + 2 hidden modules
Child extract-text-webpack-plugin:
    chunk    {0} extract-text-webpack-plugin-output-filename 10.9 kB [rendered]
         + 2 hidden modules
webpack: Failed to compile.
webpack: Compiling...
 80% optimize chunk assets^CTerminate batch job (Y/N)? y

E:\Programs\logreaper-master>npm run dev-server

> logreaper@0.0.8 dev-server E:\Programs\logreaper-master
> webpack-dev-server --config webpack-dev-server.config.js --history-api-fallback --inline --progress --port 2992

Env is: development
debug: true
devtool: inline-source-map
 70% 1/1 build moduleshttp://localhost:2992/
webpack result is served from http://localhost:8080/labs/logreaper/_assets/
content is served from ./
404s will fallback to /index.html
Hash: 323f35282c635797c445
Version: webpack 1.13.3
Time: 20040ms
                          Asset     Size  Chunks             Chunk Names
ec87ca3c137762d73b8a.worker.js  9.44 kB          [emitted]
ae083c5f93f026da6a7b.worker.js  34.3 kB          [emitted]
                        main.js  14.3 MB       0  [emitted]  main
                       main.css    62 kB       0  [emitted]  main
                     stats.json  9.32 kB          [emitted]
chunk    {0} main.js, main.css (main) 5.25 MB [rendered]
    [0] multi main 40 bytes {0} [built]
     + 1362 hidden modules
Child worker:
                              Asset     Size  Chunks             Chunk Names
    ec87ca3c137762d73b8a.worker.js  9.44 kB       0  [emitted]  main
    chunk    {0} ec87ca3c137762d73b8a.worker.js (main) 1.45 kB [rendered]
         + 1 hidden modules
Child worker:
                              Asset     Size  Chunks             Chunk Names
    ae083c5f93f026da6a7b.worker.js  34.3 kB       0  [emitted]  main
    chunk    {0} ae083c5f93f026da6a7b.worker.js (main) 8.48 kB [rendered]
```

Fig. 4.6 The Logreaper to extract & perform Log Analysis



```
C:\WINDOWS\system32\cmd.exe - node src/server/server-dev.js
E:\Programs\logreaper-master>node src/server/server-dev.js
2018-05-11T08:30:18.556Z INFO Setting the console transports level to debug
2018-05-11T08:30:18.560Z DEBUG main.jsmain.js,main.css
2018-05-11T08:30:18.560Z INFO main.js hash: 323f35282c635797c445
2018-05-11T08:30:18.560Z INFO version: 0.0.8
2018-05-11T08:30:18.560Z INFO styleUrl: http://localhost:8080/labs/logreaper/_assets/main.css?323f35282c635797c445
2018-05-11T08:30:18.561Z INFO scriptUrl: http://localhost:8080/labs/logreaper/_assets/main.js
2018-05-11T08:30:18.561Z INFO commonsUrl: undefined
2018-05-11T08:30:18.561Z DEBUG Serving up index: index_dev
2018-05-11T08:30:18.561Z INFO Env is development, running server http://127.0.0.1:8080
2018-05-11T08:30:18.604Z DEBUG Using development error handler.
GET / 302 9.872 ms - 74
GET /labs/logreaper 200 7.981 ms - -
GET /labs/logreaper/_assets/main.css?323f35282c635797c445 200 5.070 ms - -
GET /labs/logreaper/_assets/main.js 200 354.461 ms - -
GET /favicon.ico 200 3.152 ms - -
GET /labs/logreaper/formats 200 35.828 ms - -
GET /labs/logreaper/static/dist/cda566d7c74fd3ed6350.worker.js 200 1.062 ms - 976
GET /labs/logreaper/static/js/lib/xregexp-all-min.js 200 1.160 ms - -
GET /labs/logreaper/static/js/lib/stringview.js 200 0.861 ms - -
GET /labs/logreaper/static/js/lib/FileIdentifier.js 200 0.725 ms - -
GET /labs/logreaper/static/dist/fcfab34477cb47163428.worker.js 200 1.001 ms - -
GET /labs/logreaper/static/js/lib/xregexp-all-min.js 304 1.822 ms - -
GET /labs/logreaper/static/js/lib/stringview.js 304 0.584 ms - -
GET /labs/logreaper/static/js/lib/moment.min.js 200 0.591 ms - -
GET /labs/logreaper/static/js/lib/FileIdentifier.js 304 0.567 ms - -
GET /labs/logreaper/static/js/lib/Iterator.js 200 0.654 ms - -
GET /labs/logreaper/static/js/lib/ChunkParser.js 200 0.631 ms - -
2018-05-11T08:31:00.249Z DEBUG Hitting recommendations with : https://access.redhat.com/rs/recommendations?redhat_client=logreaper-dev-0.0.8&fq=documentKind%3ASolution&fq=-internalTags%3Ahelper_solution&rows=1&q
=Could%20not%20retrieve%20the%20num_generations%20from%20the%20MDEX&fl=*%2Cscore
2018-05-11T08:31:00.267Z DEBUG Hitting recommendations with : https://access.redhat.com/rs/recommendations?redhat_client=logreaper-dev-0.0.8&fq=documentKind%3ASolution&fq=-internalTags%3Ahelper_solution&rows=1&q
=%22Could%20not%20retrieve%20the%20num_generations%20from%20the%20MDEX%22&fl=*%2Cscore
2018-05-11T08:31:00.530Z DEBUG Hitting recommendations with : https://access.redhat.com/rs/recommendations?redhat_client=logreaper-dev-0.0.8&fq=documentKind%3ASolution&fq=-internalTags%3Ahelper_solution&rows=1&q
=A%20problem%20occured%20assembling%20requested%20content.%20Response%20from%20the%20assembler%20was%20enabled%3Dtrue%2C%20%40type%3DDimensionValueCacheRefresh%2C%20showCountsEnabled%3Dfalse%2C%20atg%3AcurrentSi
teProductionURL%3D%2Fcrs%2Fstoreus%2C%20relRankStrategy%3Dexact%2C%20%40error%3Dcom.endeca.infront.navigation.NavigationException%3A%20com.endeca.navigation.ENEConnectionException%3A%20Error%20establishing%20con
nection%20to%20retrieve%20Navigation%20Engine%20request%20'http%3A%2F%2Flocalhost%3A15000%2Fsearch%3Fterms%3D*%26opts%3Drel%252bexact%26node%3D0%26rank%3D1%26offset%3D0%26groupby%3Dproduct.repositoryId%26refinem
ent%3Dshowcounts%3A0%26refinement%3Dnumresults%3A50000%26irversion%3D640'.%20Connection%20refused%3A%20connect%2C%20maxResults%3D50000%2C%20maxResultsPerDimension%3D50000&fl=*%2Cscore
2018-05-11T08:31:00.533Z DEBUG Hitting recommendations with : https://access.redhat.com/rs/recommendations?redhat_client=logreaper-dev-0.0.8&fq=documentKind%3ASolution&fq=-internalTags%3Ahelper_solution&rows=1&q
=%22A%20problem%20occured%20assembling%20requested%20content.%20Response%20from%20the%20assembler%20was%20enabled%3Dtrue%2C%20%40type%3DDimensionValueCacheRefresh%2C%20showCountsEnabled%3Dfalse%2C%20atg%3Acurren
tSiteProductionURL%3D%2Fcrs%2Fstoreus%2C%20relRankStrategy%3Dexact%2C%20%40error%3Dcom.endeca.infront.navigation.NavigationException%3A%20com.endeca.navigation.ENEConnectionException%3A%20Error%20establishing%20
connection%20to%20retrieve%20Navigation%20Engine%20request%20'http%3A%2F%2Flocalhost%3A15000%2Fsearch%3Fterms%3D*%26opts%3Drel%252bexact%26node%3D0%26rank%3D1%26offset%3D0%26groupby%3Dproduct.repositoryId%26refi
nement%3Dshowcounts%3A0%26refinement%3Dnumresults%3A50000%26irversion%3D640'.%20Connection%20refused%3A%20connect%2C%20maxResults%3D50000%2C%20maxResultsPerDimension%3D50000%22&fl=*%2Cscore
2018-05-11T08:31:00.536Z DEBUG Hitting recommendations with : https://access.redhat.com/rs/recommendations?redhat_client=logreaper-dev-0.0.8&fq=documentKind%3ASolution&fq=-internalTags%3Ahelper_solution&rows=1&q
=Could%20not%20retrieve%20the%20data_date%20from%20the%20MDEX&fl=*%2Cscore
```

Fig. 4.7 Run the Node JS Server of the Logreaper



Log Stats (1265 log entries parsed spanning ~5 hours)
677 ERROR
588 WARN

Solution Recommendations
Click a to focus these recommendations.

Top WARN in message
68 No ClassLoader provided, using TCCL: org.jboss.managed.api.annotation.ManagementComponent
43 No remote servers configured
42 Unable to connect to lock servers DESKTOP-0D2SPPG/192.168.230.1:9012 - globally locking is disab...
34 WARNING! POTENTIAL SECURITY RISK. It has been detected that the MessageSucker component which suc...
28 EJB configured to bypass security. Please verify if this is intended.

Fig. 4.8 Analysis Report

V. CONCLUSION

The proposed project is concluded by proposing two modules one getting the instrumented customer or user data for use in the application and another with the test automation suite with the instrumented data that is available in the data storage as discussed above. The activity cycle of both the User and Developer is shown below.

1. Automated Functional, Regression & Integration Testing with CX Data
2. Lesser Efforts on Triaging and Reproduction of Customer Defects
3. Improved Productivity
4. Results in Reliable and Scalable Application
5. Improved Customer Experience, Conversion and Revenue
6. Improved Application Metrics

REFERENCES

1. J.S. Bridle, "Probabilistic Interpretation of Feedforward Classification Network Outputs, with Relationships to Statistical Pattern Recognition," Neurocomputing—Algorithms, Architectures and Applications, F. Fogelman-Soulie and J. Herault, eds., NATO ASI Series F68, Berlin: Springer-Verlag, pp. 227-236, 1989. (Book style with paper title and editor)
2. VahidGarousi, Frank Elberzhager, "Test Automation: Not just for test execution", Volume:34, Issue 2, 90-96, IEEE Transactions, 2017
3. AbdelilahSakti, Gilles Pesant, "Instance Generator and Problem Representation to Improve Object Oriented Code Coverage", VOL. 41, NO. 3, IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, 2015
4. A. Arcuri, G. Fraser, and R. Just, "Private API Access and Functional Mocking in Automated Unit Test Generation," in ICST'11: Proceedings of the 10th International Conference on Software Testing, Verification and Validation, 2017.
5. M. M. Almasi, H. Hemmati, G. Fraser, and A. Arcuri, "An Industrial Evaluation of Unit Test Generation: Finding Real Faults in a Financial Application," in 39th International Conference on Software Engineering, ICSE 2017, Software Engineering in Practice Track, 2017, pp. 263-272.
6. S. Shamshiri, R. Just, J. M. Rojas, G. Fraser, P. McMinn, and A. Arcuri, "Do Automatically Generated Unit Tests Find Real Faults? An Empirical Study of Effectiveness and Challenges," in Proceedings of the 30th IEEE/ACM International Conference on Automated Software Engineering (ASE), 2015, pp. 201-211
7. J. M. Rojas, J. Campos, M. Vivanti, G. Fraser, and A. Arcuri, "Combining Multiple Coverage Criteria in Search-Based Unit Test Generation," in Search-Based Software Engineering, Springer, 2015, pp. 93-108.
8. A. Arcuri, G. Fraser, and J. P. Galeotti, "Automated Unit Test Generation for Classes with Environment Dependencies," in IEEE/ACM Int. Conference on Automated Software Engineering (ASE), New York, NY, USA, 2014
9. G. Fraser and A. Arcuri, "Whole Test Suite Generation," IEEE Transactions on Software Engineering, vol. 39, iss. 2, pp. 276-291, 2013.
10. J. Campos, R. Abreu, G. Fraser, and M. d'Amorim, "Entropy-based Test Generation for Improved Fault Localization," in IEEE/ACM Int. Conference on Automated Software Engineering (ASE), 2013, pp. 257-267.
11. J. P. Galeotti, G. Fraser, and A. Arcuri, "Improving Search-based Test Suite Generation with Dynamic Symbolic Execution," in IEEE International Symposium on Software Reliability Engineering (ISSRE), 2013, pp. 360-369
12. 12.http://ieeexplore.ieee.org/search/searchresult.jsp?queryText=Test %20Automation&newsearch=true&refinements=4291944246&ranges =2017_2017_Year&punumber=52
13. 13.http://ieeexplore.ieee.org/search/searchresult.jsp?queryText=Test %20Automation&newsearch=true&refinements=4291944246&ranges =2016_2016_Year&punumber=52

AUTHORS PROFILE

Dr.R.Kavitha received her Master's in Software Engineering from College of Engineering , Anna University and Ph. D in Computer Science and Engineering from Vel Tech, Chennai. Her research areas are Data Mining, Image Processing and Software Engineering. Presently working as Associate Professor at Vel Tech, Chennai having 10 yrs of Teaching experience.

P.Subha received her Master's in Software Engineering from Vel Tech Multitech Engineering College, Chennai. Her research areas are Data Mining, Image Processing and Software Engineering. Presently working as Associate Professor at Vel Tech, Chennai having 8 yrs of Teaching experience.

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication (BEIESP)