Increasing Web UI code coverage and regression methodology improvements for Venus (arm Physical IP logistics project)

Submitted By

Nirali Kotak

16MCEC08

DEPARTMENT OF COMPUTER ENGINEERING

INSTITUTE OF TECHNOLOGY

NIRMA UNIVERSITY

AHMEDABAD-382481

May 2018

Increasing Web UI code coverage and regression methodology improvements for Venus

Major Project

Submitted in partial fulfillment of the requirements

for the degree of

Master of Technology in Computer Science and Engineering

Submitted By

Nirali Kotak

(16MCEC08)

Guided By

Prof. Rupal Kapdi

DEPARTMENT OF COMPUTER ENGINEERING

INSTITUTE OF TECHNOLOGY

NIRMA UNIVERSITY

AHMEDABAD-382481

May 2018

Certificate

This is to certify that the major project entitled **"Increasing Web UI code coverage and regression methodology improvements for Venus"** submitted by **Nirali Kotak (16MCEC08)**, towards the partial fulfillment of the requirements for the award of degree of Master of Technology in Computer Science and Engineering of Nirma University, Ahmedabad, is the record of work carried out by him under my supervision and guidance. In my opinion, the submitted work has reached a level required for being accepted for examination. The results embodied in this major project part-I, to the best of my knowledge, haven't been submitted to any other university or institution for award of any degree or diploma.

Prof. Rupal Kapdi
Guide & Assistant Professor,
CE Department,
Institute of Technology,
Nirma University, Ahmedabad.

Dr. Priyanka Sharma
Professor,
Coordinator M.Tech - CSE
Institute of Technology,
Nirma University, Ahmedabad

Dr. Sanjay Garg
Professor and Head,
CE Department,
Institute of Technology,
Nirma University, Ahmedabad.

Dr Alka Mahajan
Director,
Institute of Technology,
Nirma University, Ahmedabad

Statement of Originality

I, **Nirali Kotak**, **16MCEC08**, give undertaking that the Major Project entitled "**Increasing Web UI code coverage and regression methodology improvements for Venus (arm Physical IP logistics project)**" submitted by me, towards the partial fulfillment of the requirements for the degree of Master of Technology in **Computer Science & Engineering** of Institute of Technology, Nirma University, Ahmedabad, contains no material that has been awarded for any degree or diploma in any university or school in any territory to the best of my knowledge. It is the original work carried out by me and I give assurance that no attempt of plagiarism has been made.It contains no material that is previously published or written, except where reference has been made. I understand that in the event of any similarity found subsequently with any published work or any dissertation work elsewhere; it will result in severe disciplinary action.

_____

Signature of Student

Date:

Place:

Endorsed by

Prof Rupal Kapdi

Acknowledgements

It gives me immense pleasure in expressing thanks and profound gratitude to **Prof. Rupal Kapdi**, Associate Professor, Computer Engineering Department, Institute of Technology, Nirma University, Ahmedabad for her valuable guidance and continual encouragement throughout this work. The appreciation and continual support she has imparted has been a great motivation to me in reaching a higher goal. Her guidance has triggered and nourished my intellectual maturity that I will benefit from, for a long time to come.

It gives me an immense pleasure to thank **Dr. Sanjay Garg**, Hon'ble Head of Computer Engineering Department, Institute of Technology, Nirma University, Ahmedabad for his kind support and providing basic infrastructure and healthy research environment.

A special thank you is expressed wholeheartedly to **Dr. Alka Mahajan**, Hon'ble Director, Institute of Technology, Nirma University, Ahmedabad for the unmentionable motivation she has extended throughout course of this work.

I would also thank the Institution, all faculty members of Computer Engineering Department, Nirma University, Ahmedabad for their special attention and suggestions towards the project work.

- Nirali Kotak

16MCEC08

Abstract

Code coverage is one of the method to identify the software application bugs. It helps to identify the loopholes which can cause a problem in the application and are left untouched while verifying different scenarios. Venus is an arm IP logistics web UI application developed in EXTJs. Fabric test in python is one method to run regression suite and track the code coverage which team has been using currently. The automation framework provided by sencha technology is the another method proposed to increase the code coverage for EXTJs app. In this report, an approach based on sencha test automation framework is presented for parallel execution of the regressions and track the code coverage changes with addition of different scenarios. The results in this report shows improvement in the code coverage in comparison to the fabric test method used which will enable the team to work on further optimization approaches for regressions.

Abbreviations

EXTJs	Extended Javascript.
PDM	Platform Manager
NDM	Naming Database Manager
PCM	Post Contract Manager
ESM	Part Manager

Contents

List of Figures

Chapter 1

Introduction

1.1 General

arm works in areas of designing and delivering processors catering to various need of its customers. The requirements of customers vary in terms of optimization parameters like memory, processing capacity, battery consumption etc. ARM Holdings has grown from a small Acorn in Cambridge - maker of some of the earliest home computers - into one of the world's most important designers of semiconductors, providing the brains for Apple's must-have iPhones and iPads

The ARM processor was licensed to many semiconductor companies for an upfront license fee and then royalties on production silicon. This made ARM a partner to all these companies, effectively trying to speed up their time to market as it benefited both ARM and its partners. For each set of customer requirement arm follows a strict process of delivering the final product (libraries) to keep product quality consistent. To ensure this process is flexible and easy to use, the release work flow is automated using software systems.

Essentially, the release work flow consists of four stages, namely: 1) Packaging the parts, 2) Building the release tree, 3) Validation of the release tree and 4) Upload and certify. Mercury application aims at automating above product release flow.

Each phase of the work flow varies according to the requirements of the customer and hence requires some amount of configuration. This configurations are stored and maintained by yet another software system: Venus.

1.2 Technology Services Group

TSG business group is responsible for providing the infrastructure, software and technology solutions for the arm other business groups. TSG is responsible in introducing cutting edge technology for its solutions deliver the product to customer for other business group

1.3 Venus: Configuration management

Venus is developed to streamline different arm divisions to release a product using technological solutions, user workflows and metadata management. It is basically a database driven approach with respect to Mercury and also pre-production tasks. Venus is web application that stores and provides access to configurations to various other software systems(like mercury). Venus is used by Marketing team, arm Design group, Technology Service Group,etc to configure the build workflow for each customer.The maintainence of Venus is with the Technology Services Group. The technology stack of Venus application is : mysql as database, python/django on the server and EXTJs as the front-end. The application is hosted on RHEL(Red Hat Enterprise Linux) datacentre.

The front-end is using EXTJs 6.2. EXTJs uses MVVC architecture. Venus implements six functionalities using the MVVC architecture of EXTJs. EXTJs is a client side javascript framework and it has a mutually exclusive codebase than python/django. For python/django codebase unit test cases have been implemented that ensures the robustness of application after each development release. There is no any regressions suite to support very few functional use cases. Also there is no regression suites for EXTJs components leaving the chances of software bugs to exist more.

The current regression strategy for Venus application uses CICD which uses Jenkins. Regressions runs are configured to run every day. Current regressions results in failing of the functional use cases. Also it does not support the code coverage and is run in sequential.

Chapter 2

TSG Architecture

2.1 TSG IPLogistics

TSG Technology Services Group of arm has been working to automate the business workflows of arm product release flow. TSG has been successful in delivering and streamlining the process flows based on the business requirements. The stakeholder of TSG team are Marketing Engineering, Software Developers and the client. Before TSG it used to take days/weeks to deliver the product to the client. The IP library files are delivered to the client which is generated through different stages of the release workflow. This required human interference which attracting errors. This was too much of effort and also sometimes required to revisit client requirements during the product release resulting into inconsistency at different stages for the data. TSG approaches to solve this problem through software introducing consistency through the release flow. The software solution enforces different teams to be consistent with data at different stages of the release cycle resulting in product release with high quality.

Different teams work for different workflows of arm business release flow. As shown in

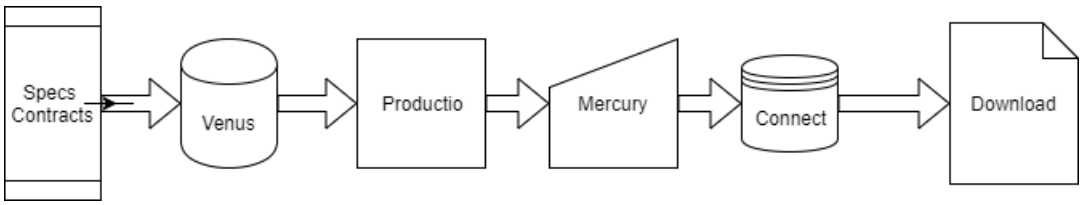

Figure 2.1: TSG

fig 2.2,

The software solutions help the Engineering team in accessing the data through its different stages of release cycle and deliver the product to the client. This data generated is fed to Mercury(named after greek Messenger God) which goes through certain process to release the product to the client as shown in figure 2.1 Pegasus team works for PDG Memory Builder team works for PDG SC and IO. Mercury takes care of product logsitics. It collects the data generated from the applications and does shipment of the files to the server from where the client accesses the data. Venus is the central database for storage and access to the data by different teams. The contract is recorded in Venus which is a pre-production task.

From figure 2.2, at high level Mercury flow has multiple info stream which has led to variety of inconsistencies. Mercury has introduced NDM feature which enforces the Engineering team to enter the file patterns and conventions of the part of product to ensure consistency of the data. Logistics introduction has been successful in achieving 10x performance. Some key features of Mercury are:

- Auto-generated config file based on SAP. Before Mercury config file was created manually
- Ensures staged structure approach
- Enforcement of Marketing's Naming conventions

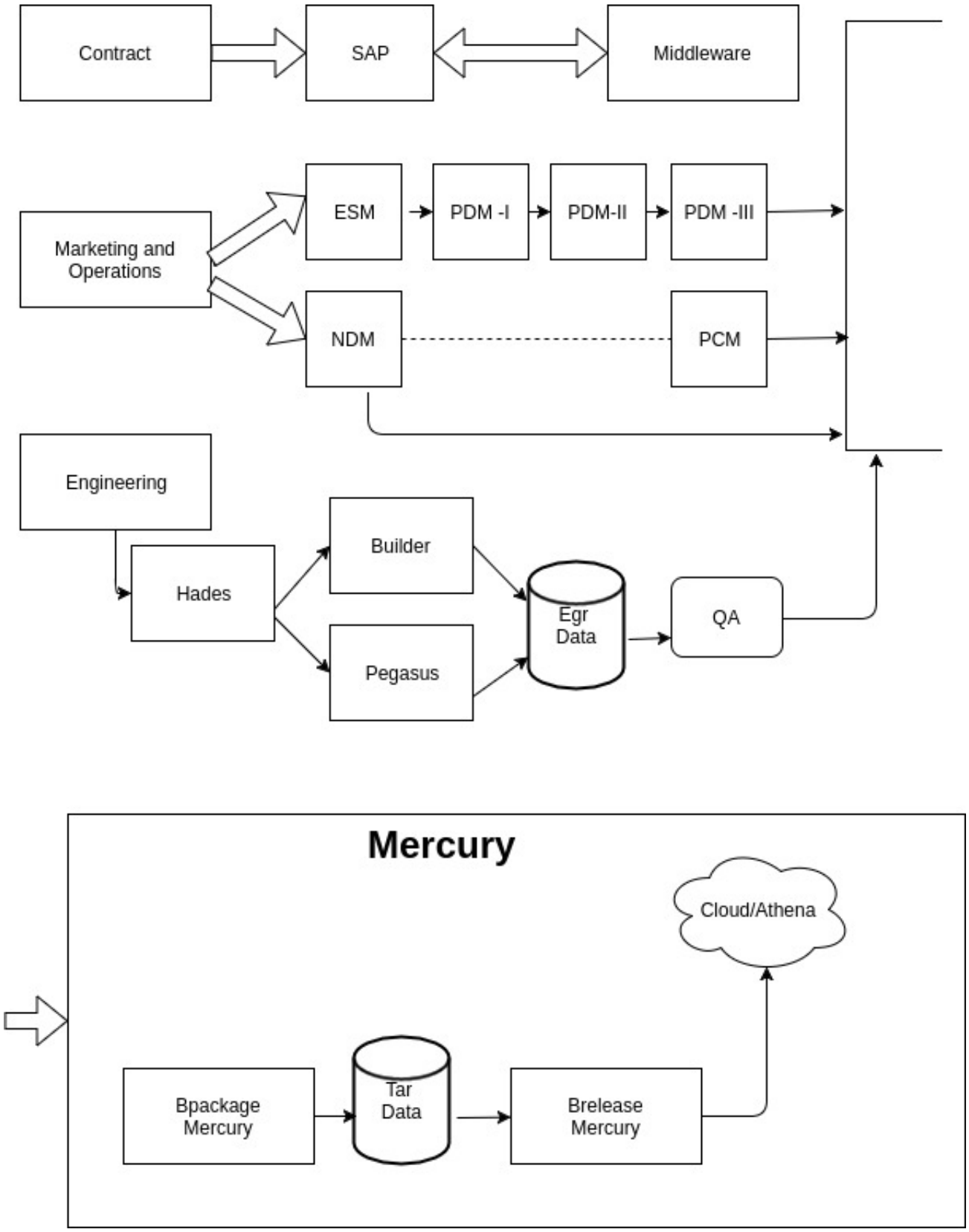

Figure 2.2: TSG Architecture

Chapter 3

Venus Code Coverage

3.1 Technique:-

Code Coverage is a method to identify the degree of the source code covered when a specific designed test suite is executed. The code which is not covered with most of the use cases is more prone to software bug and hence the software quality is compromised. Also the code which seems to be unused can be confirmed by code coverage results and the application can be improved by removing unused code. Code coverage results are identified based on various criteria. The main criteria which helps identifying correct code coverage are:

- Branch Coverage: This coverage ensures that a branch is executed for all the conditions depending on different data combinations for the process flow
- Statement Coverage: This coverage ensures that all the statements of the code is covered.
- Function Coverage: This coverage ensures that all of the functions in the code base is executed and working fine.

Approaches identified:

- Manual Verification with Code Coverage: Chromium latest version supports the code coverage.One approach could be every time before the release developer could

manually run specific scenario and gather the code coverage results for the source code. This has its own disadvantages:

  - Developer might miss to run some use cases
  - Human efforts are more

- Automate the code coverage: Sencha Test uses jasmine framework for developing scenarios. Automating the scenarios will benefit the human efforts required to verify the app. It is like Develop once and use multiple times approach which will help team to to improve the release process with an additional quality check with code coverage alongwith the existing regressions. Sencha test supports Web driver and In browser scenarios. The Webdriver based scenarios uses the selenium webdriverio api and In browser scenario which supports EXTJs api which runs inside the browser. For code coverage in current approach we have configured In browser scenario and sencha command line(STC) with chrome headless browser.This approach is configured to run in parallel. This will help in optimizing the time taken for the nightly regressions

Chapter 4

Implementation and Analysis

4.1 Implementation

Earlier the scenarios are executed in sequential. As the scenarios increase the run time for the regressions increase. This is a major drawback when the regressions are run in nightly builds as it occupies the resources which are dedicated for execution of the regressions. Secondly, the scenarios are run using local databases which can be common to all the developers. This creates inconsistency in the data if a scenario executed performs CRUD operation to the database.

- Currently our implementation is In browser scenario. Test author can use one of the two below implementations:
  - Webdriver scenario

    In webdriver scenario, the test and application are executed differently and the communication between the two can be done through Futures API. Using Futures API, we can locate the already-existing component in application. Under the hood in Webdriver scenario sencha test leverages the WebDriver. The test author has to focus on writing test as interacting with the WebDriver is handled by sencha test.
  - In-browser scenario

    [1] Main advantage of in-browser scenarios for test author is he/she could use the EXTJs API to locate the component or perform an action. The test

and the application run together so it is as good as test is injected inside the application and it executes the test as it is part of the application code. Code coverage tool Istanbul can be configured to track the code coverage of the application

- Parallel execution

  Parallel execution improves time as it utilizes the maximum processing core available in the cluster machine.

  For parallel execution, scenarios should be logically identified by the test author such that the tests are not dependent on each other. If the tests are dependent on any previous data, then it should be executed in sequential.
  For example, Reorder scenario in pdm is enabled only for the owner of the pdm. This requires first creation of the pdm and then reorder Products or PVTS.

- Fixtures

  Application technology stack is :

  - Python 2.7.13 and django 1.10
  - MySql as backend
  - ExtJs as frontend

  [2] django is free and open source Python Web framework. It helps developers to write the application without taking care of much of the web development hassle as it is all handled in django. django is based on MVT architecture where M- Model, V- view and T- template. django takes care of the controller which communicates between the view and model. [3]
  [2] django commands/utilities:

  - dumpdata : This command takes a dump of the database and outputs it to the file mentioned in the command. dumpdata refers to the database mentioned in django settings.py file. It takes default database mentioned under the default subscript if no database name is mentioned For example, python manage.py dumpdata –database venus - This takes dump of venus database python manage.py dumpdata - This takes dump of database in default[db] subscript.

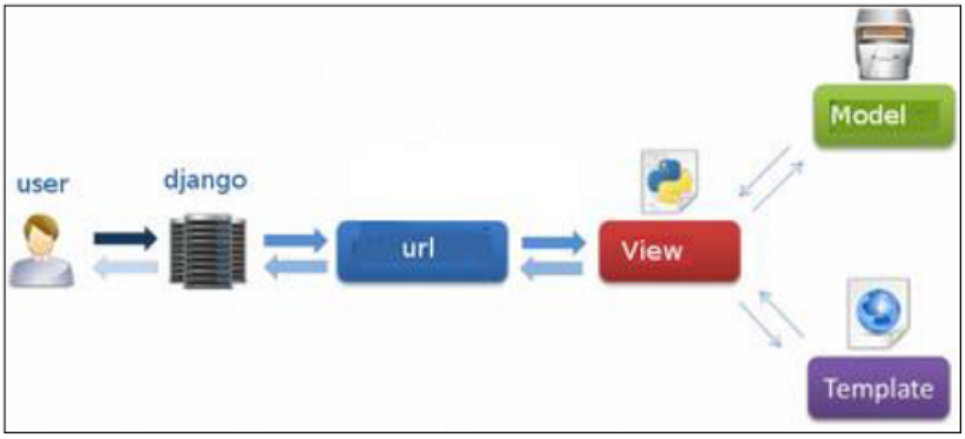

Figure 4.1: Django architecture

- makemigrations : This command is responsible to create migrations for any app level changes. For example, makemigrations ¡applabel¿
- migrate : Synchronizes the database state with the current set of models and migrations.This command handles any migrations in django. Any model changes(any database schema changes will be migrated/applied when migrate command is used) For example migrate applabel 0002venus
- runserver : Spawns a lightweight development server. Default IP address and port is localhost and 8000 respectively. Any server side code changes don't require restarting the server. It refreshes the server with the changes.
- testserver : Spawns a lightweight development server with the predefined fixture file. It creates new database and applies all the migrations. After applying migrations it loads up the data from the fixture file. This server starts up with the new database created with the specified IP address and port

Pre-requisites for Fixture implementation :

- Database cleanup : Removing any foreign-key constraints from the database
- Inconsistency between django model and database tables Also django fixture patch was introduced in the current working environment.

- Periodic execution of regressions

  Currently there is no setup for nightly regressions or regression run at specific interval of time. To configure Jenkins access to rhe7 machine is required where

jenkins is installed. Initially I tried configuring cron job for a periodic execution i.e everyday

Cron job is a linux based utility which executes a script or commands like a user does at specific interval. User needs to configure crontab. Some cronjob commands: crontab -e : opens up the crontab where one can see the cronjobs configured to execute. User can edit the configurations for the cronjobs. Edit may include changing the time when to execute, changing the job to be executed, adding/deleting any job,etc...
crontab -l : Lists the cron job scheduled
MAILTO argument in cronjob sends the standard output to the recipient specified as value to the MAILTO argument.
Continuous Integration/ Continuous Deployment Jenkins is the preferred tool in the organization for the Continuous Integration and Continuous Deployment. Currently venus regressions are running in Jenkins rhe6 machine. Sencha test requires rhe7 machine. To configure Jenkins to run on through rhe6 machine till the time venus regressions are ported to rhe7 or a dedicated cluster machine is made available on rhe7 we are spawning the sencha tests through current regressions.

bsub command requests any available rhe7 host and runs the script. This script executes the commands stepwise which are required to run sencha tests. For example, Activate the virtual environment, source the required modules and run sencha tests.
busb -R rhe7 rusage[mem=10000] ./script.sh

- Display output Earlier the output displayed was a summary of number of scenarios executed, number of tests passed and number of tests failed like below :

  - Summary
    Total time: 1200.884s
    Scenarios ran: 45 (parallel = 45)
    Scenarios failed: 0 (parallel = 0)

Now the summary is improved to display the error message of each failure alongwith the failure test and whether it is a System error or a genuine failure ¡packagename¿:

¡Testname¿ : Expected screenshot mismatch with baseline System Error: ¡packagename¿: ¡Testname¿ : Expected screenshot mismatch with baseline

4.2 Strategy

Currently the project is configured to run in hybrid mode where read based scenarios are run with local database whereas write based scenarios are executed with fixture based database.

Sencha tests are executed using testclientui method. Currently testclientui method spawns two servers : runserver and testserver

runserver (on port 9000+ useroffset)

testserver(on port 10000 + useroffset)

For every scenario, identification of read/write scenario is done based on value of typeoftest attribute.

"typeoftest" : "read"(on port 9300)

"typeoftest" : "write"(on port 10300)

4.3 Implementation Challenges

- Headless browser compatibility : phantomjs vs Chrome headless.
  Chrome headless is preferred headless browser now which replaces the phantomjs browser from the configuration. Earlier with phantomjs browser we were unable to perform certain actions like double click or right click which works with Chrome headless

- EXTJs API challenges
  Right click action is one of the dominant action for any write operation in our application. This can be achieved using the EXTJs right click code as there is no available api to do right click in In-browser scenario.

- Configure Istanbul (sencha compatible code coverage tool) and excluding the sencha libraries from code coverage results.This posed as one of the challenge.

- Fixture implementations
  Initial data dump used was 70 MB in json file format. This file was difficult to open. Any third party tool was not helpful. Only vi tool could do the job but that

too was very slow and it worked with a time lag for any action on the file. Data reduction was the only option for this file. Also fixture loading took 15-20 minutes of time which is very high each time for a developer.

Initial data was identified which are applied through django migrations and as testserver initially applies django migrations the default app were excluded from the dumps. These reduced the size of the file to half and not it was 35 MB. Reducing further required sheer analysis of the relationship between the tables and its content. This data is currently 700 kb. The time taken to start up the testserver is now reduced to 40-45 secs which is a great cut-down and optimizing of time from the previous value.

4.4 Results and Analysis

Results can be viewed as scenario execution, pass scenarios, fail scenarios and code coverage report.

- pdm : newpdm : Expected screenshot failure
  app2 : newpdm : Expected screenshot failure
  System Error:
  app3 : newpdm : Expected screenshot failure

  Summary
  Total time: 500.84s
  Scenarios ran: 63
  Total test cases ran : 187
  Total test cases passed : 126
  Total test cases failed: 61

- Code Coverage : Currently the code coverage is 44.32 percent

/

44.32% Statements 3049/6880 **25.29%** Branches 505/1997 **44.3%** Functions 676/1526 **44.35%** Lines 3041/6857

File	Statements		Branches		Functions	
ambrosia/		90%	9/10	83.33%	5/6	
auspex/		100%	12/12	100%	0/0	
controller/ambrosia/		100%	14/14	100%	0/0	
controller/auspex/		100%	12/12	100%	0/0	
controller/common/		100%	9/9	50%	1/2	
controller/home/		57.14%	8/14	100%	0/0	
controller/mercury/		29.95%	174/581	12%	9/75	
debug_toolbar/js/		24.85%	42/169	16.18%	11/68	
esm/		34.21%	13/38	18.75%	3/16	
home/		100%	4/4	100%	0/0	
iprd/		37.84%	14/37	27.78%	5/18	
mercury/		83.91%	73/87	67.86%	19/28	
metadata/		91.67%	11/12	75%	3/4	
model/ambrosia/		100%	10/10	100%	0/0	
model/auspex/		100%	3/3	100%	0/0	
model/esm/		100%	13/13	100%	4/4	
model/iprd/		100%	7/7	50%	1/2	
model/mercury/		100%	8/8	100%	0/0	
model/metadata/		100%	36/36	83.33%	5/6	
model/metadata/ambrosia/		100%	1/1	100%	0/0	
model/ndm/		100%	9/9	100%	2/2	
model/pcm/		100%	3/3	100%	0/0	
model/pcm/pdm/		50%	2/4	0%	0/2	

Figure 4.2: Code Coverage

model/metadata/ambrosia/		100%	1/1	100%	0/0	
model/ndm/		100%	9/9	100%	2/2	
model/pcm/		100%	3/3	100%	0/0	
model/pcm/pdm/		50%	2/4	0%	0/2	
model/pdm/		97.5%	39/40	85.71%	12/14	
model/pdm/ambrosia/		100%	1/1	100%	0/0	
model/rdm/		85.71%	6/7	100%	0/0	
ndm/		50.63%	40/79	28.13%	9/32	
pcm/		92.31%	12/13	75%	3/4	
pdm/		72.09%	31/43	44.44%	8/18	
rdm/		91.67%	11/12	75%	3/4	
store/ambrosia/		100%	15/15	100%	0/0	
store/auspex/		100%	3/3	100%	0/0	
store/common/		33.33%	1/3	100%	0/0	
store/esm/		92.86%	13/14	100%	0/0	
store/esm/ambrosia/		100%	1/1	100%	0/0	
store/iprd/		100%	1/1	100%	0/0	
store/iprd/metadata/		100%	1/1	100%	0/0	
store/mercury/		100%	8/8	100%	0/0	
store/mercury/metadata/		100%	1/1	100%	0/0	
store/mercury/pdm/		100%	1/1	100%	0/0	
store/metadata/		41.67%	30/72	0%	0/36	
store/metadata/ambrosia/		100%	2/2	100%	0/0	
store/ndm/		100%	7/7	100%	0/0	
store/ndm/ambrosia/		100%	1/1	100%	0/0	
store/ndm/esm/		100%	1/1	100%	0/0	
store/pcm/		100%	3/3	100%	0/0	

Figure 4.3: Code Coverage

view/iprd/viewcontroller/		72.5%	29/40	50%	4/8	
view/iprd/viewmodel/		100%	1/1	100%	0/0	
view/mercury/		100%	77/77	50%	5/10	
view/metadata/		100%	58/58	100%	0/0	
view/metadata/viewcontroller/		34.62%	72/208	22.22%	8/36	
view/metadata/viewmodel/		100%	7/7	100%	0/0	
view/ndm/		100%	6/6	100%	0/0	
view/ndm/common/		58.97%	46/78	38.46%	20/52	
view/ndm/deliverable/		100%	4/4	100%	0/0	
view/ndm/filepattern/		61.54%	16/26	0%	0/4	
view/ndm/ndmsearch/		100%	12/12	100%	2/2	
view/ndm/property/		84%	21/25	0%	0/2	
view/ndm/target/		64%	16/25	100%	0/0	
view/ndm/viewcontroller/		65.85%	268/407	53.85%	42/78	
view/ndm/viewmodel/		100%	7/7	100%	0/0	
view/pcm/		63.41%	26/41	100%	0/0	
view/pcm/viewcontroller/		32.5%	65/200	6.38%	3/47	
view/pcm/viewmodel/		100%	2/2	100%	0/0	
view/pcm/viewmodel/pdm/		100%	1/1	100%	0/0	
view/pdm/		50.75%	202/398	28.66%	47/164	
view/pdm/viewcontroller/		44.81%	604/1348	22.4%	69/308	
view/pdm/viewmodel/		100%	20/20	100%	0/0	
view/rdm/		54.55%	48/88	0%	0/8	
view/rdm/viewcontroller/		4.87%	31/637	0%	0/148	
view/rdm/viewmodel/		100%	5/5	100%	0/0	

Figure 4.4: Code Coverage

Chapter 5

Conclusion and Future Work

5.1 Conclusion

This project exercise has benefitted in identifying some common bugs which exist in the application and to make it more robust. Currently one of the application in venus has introduced Live versioning feature. This has brough about change in th UI and also some new bugs were detected as part of these changes. The bugs were easily identified with the new developed sencha regressions and reported timely before the feature was deployed in production. This helped developers to identify the broken corners of the code and fix them. Regression execution helped to validate the correctiveness of the new feature introduced.

This approach based on sencha test automation framework is feasible for parallel execution of the regressions and track the code coverage changes with addition of different scenarios. The results in this report shows improvement in the code coverage in comparison to the fabric test method used which will enable the team to work on further optimization approaches for regressions.

5.2 Future Work

- Achieve 80 percent code coverage. This activity will help in smooth migration of the application to higher version or any feature changes/improvements as it will identify the new bugs introduced and also the unused code or stale code.
- Fixture implementation for multiple database
- Achieve fixture implementation for read based scenarios and spawn only one server
- Migrate mercury app in venus web application to the current ExtJs 6 version

Bibliography

[1] "Sencha documentations(url: https://docs.sencha.com/sencha_test/2.0.2/guides/product_overview.html)."

[2] "Web documentation (url: https://docs.djangoproject.com)."

[3] "Django tutorial(url: https://www.tutoriapoints.com/django/django_overview.htm)."