I J C T A, 9(24), 2016, pp. 261-276 ISSN: 0974-5572
© International Science Press

Insprint Automation in Agile Scrum– A Case Study

Kiran Jammalamadaka* and Ramakrishna V.**

ABSTRACT

Test automation is pivotal with focus on reducing test execution time and also acts as a safety net to catch regression defects in house. In agile environment with shorter delivery cycles, emphasis for test automation extends beyond regression testing to new features. Ability to develop sustainable, faster and reliable automation is important in developing automation in parallel to development and late integration of scripts to create the end to end workflow. This requires a new mind set and practices which brings new challenges with maintainability and reliability of test automation. In this paper we discussed a spiral strategy blended with critical review of automation pyramid and came up with right automation at right level concept, and also discussed a few challenges we faced and how we overcame to sustain the speed in the whole development process. And also presented a perspective of actual and projected savings with' in sprint' automation.

Keyword: Agile, test automation, in sprint automation, projected savings, actual savings, right automation, automation pyramid

1. INTRODUCTION

In the current era of software development, Agile has become very popular and attracted many researches towards it because of continuous delivery of working software, and focus on shorter feedback cycles to improve overall quality.

Approach towards quality is major difference in the sequential and agile models. Sequential models spend enormous effort and budget at the later stages of the development cycles, which result in quality compromises due to schedule/effort constraints. Agile focuses on iteratively building valuable software what is important to customer. Testing is no more a phase and it has become part of the development process. In agile every sprint on sprint no of test cases get added to the test suite, here the major challenge is to test the new software with in the sprint to ensure no regressions at the same time need to ensure no regressions with the delivered features in previous sprints, this increases more focus on the test automation to be in place and should start adding value to the product by reducing test efforts.

Agile advocates sustainable development, with ability to welcome changes. Balance between speed of iterative development, along with assurance of existing and legacy feature development will be paramount important. It's very obvious to look at automation of validation processes to cope up with the speed and quality.

1.1. Software Testing

Software testing is the process to check the computer code whether it's working as it is designed for or not and does not do anything unintended [7]. The main focus of the testing is to uncover software failures, which means where the software is not performing as expected, and at the same time it is almost impossible to verify the software with all the data combinations and environments. Hence, testing does not guarantee

* K L University Andhra Pradesh, India, *Email: jvskkiran@gmail.com*

** K L University Andhra Pradesh, India, *Email: vramakrishna@kluniversity.in*

the computer program is defect free, however it gives confidence before shipping to the stakeholders. The intensity of the testing varies from application to application and a lot factors like and not limited to target audience, possible environment conditions and purpose of the application etc. Each application has its own target audience, for example a medical application has target audience could be people related to health care and for a video game target audience could be general public including kids and adults, and for educational software teachers and students mainly.

And criticality of the applications influence the testing in terms of time and data combinations' a normal informative website has less implications in case of any defects with respect to a critical medical application which is used during a surgery, the amount of risk of the defect is high at the later state.

The process of software failure will happen through an error made by a human for various reasons including requirement gaps or misunderstanding or misinterpreting of the requirements or any other technical issues, once the error gets executed it's become a fault and when then when it is in operational a fault becomes failure.

The cost to a fix a defect is proportional to the stage where it's uncovered, the early the cheaper [11].

If a defect is uncovered after the release it would have been much cheaper if it's uncovered at the requirement stage itself.

In early stages of software development and testing were not two different parts both have been done by the software developer, and till 1980s testing and debugging were considered as same. Glenford J. Myers in 1979 separated debugging from testing [13].

In 1988, Gelperin D and Hetzel B, mentioned the focus areas and time lines in The Growth of software Testing [14].

Time lines and focus areas of Software testing are represented in the below table I.

Figure 1: Evolution of an error to a failure

Figure 2: Cost of the defect at various stages

Table 1
Time lines and Focus areas of testing

Time Line	*Focus*
Until 1956	Debugging
1957–1978	Demonstration
1979–1982	Destruction
1983–1987	Evaluation
1988–2000	Prevention

1.2. Testautomation

Software testing play a major role in software development life cycle, and according to the normal business application is around 40% of the total project[4], and in case of mission critical applications its even more up to 90%, where applications cannot effort to slip any defect for example health care applications. The major challenges with manual testing is repetitive which is time consuming and a few tasks which humans are easily prone to do errors , like comparing a huge amount of data or comparing two images are a few examples. Hence automation is mimicking the human actions with perfection in order to test the application. However, it's been proved for many years that test automation is a good strategy in order to reduce the testing time without compromising on the quality [6].

Automation tests safe guard the application from the new errors getting induced during the new feature development and manual tests are traditionally for new features [5].

1.3. Testautomation Frameworks

Test automation framework is an environment to execute the automated tests, a framework is required for optimum usage of resources and minimal maintenance of the test scripts, there are different types of approaches

1) Linear: This approach is about procedural code, probably generated by a tool, a few tools generate code as per the operations made by the test automation developer, this is also known as record & play back tool. Typical use case would be test automation developer mimics the action of the real user by using mouse and keyboard to navigate and key in the data for a particular test case. As per the actions made by the test developer the tool will generate the code in linear fashion, the same can be saved and replayed again and again, however there are a few advantages as well disadvantages of this approach which is not in the scope of this paper. John Kent explained about linear approach in detail in his article [8] about explaining advantages and disadvantages.
2) Structured: Structured framework is also known as descriptive programming, this allows test automation developer to write the code in supported scripting languages like vb script, typically an editor will be provided to write the business logic using if-else and loops etc. It gives the flexibility to the test automation developer to write his/her own assertions and reusable functions which minimizes the maintenance of the scripts unlike above mentioned linear approach. M. Fewster explained in depth about structured framework [9].
3) Data Driven: Data driven approach uses the common test logic with different sets of test data, in this approach all the variable data will be separated and provided as an input to the test logic and output will be validated. The maintenance cost is relatively low with respect to the record playback. Linda G. Hayes [15] explained advantages and dis advantages in detail.
4) Key word driven: Key word driven approach is also known as "action word based testing", this approach enables the end test developers to prepare the scripts in faster manner. In this approach a set of key words will be defined to represent a different action, all the actions will be placed into a table as a keyword as shown in the below table.

Table 2
An example of key word driven

Test Case1	*Login Test case*			
	Control	*Property*	*Action*	*Data*
	Webedit	Uname	Set	name
	Webedit	Pword	Set	Password
	Webbutton	SignIn	Click	

P. Laukkanen, mentioned details about key word driven in great detail and also compared with data driven approach [16].

1.3.1. Automation Pyramid

From the Fig 3, it appears like UI tests are more lucrative, as most of the test coverage is more by a single UI Test than service, and unit, however Mike Chon's mentioned in "Succeeding with agile" book. about UI tests' pitfalls and suggested to focus more in service layer [19] and unit tests rather UI tests,

Mike explains unit tests are the foundation of any application.

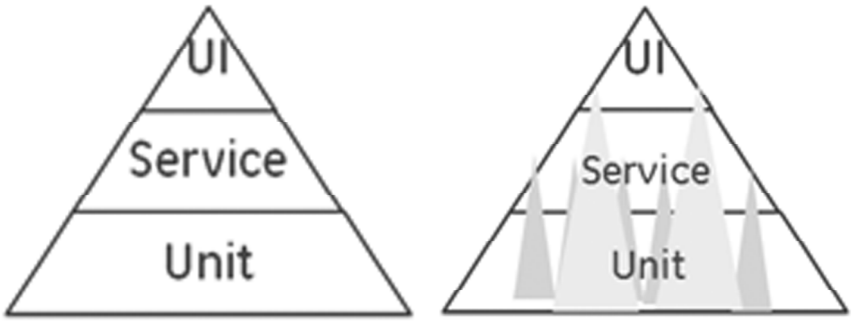

Figure 3: Automation Pyramid

1.4. Agile Development

Broadly, software development models can be classified into two major buckets [3]

1. Predictive
2. Adaptive

For considerable time, majority of the industry players are using waterfall model, a major base for predictive models. However, the said model has its own drawbacks in terms of delivering the business process to the customer. As the name suggests, the waterfall model is based upon a few assumptions

Like clear understanding of the requirements (which is getting nearly impossible in the current situation). The size as well as complexity of business process is drastically increasing in tune with the continuously evolving software industry

1.4.1. Manifesto and Principles of Agile

In 2001, Kent beck and his team introduced agile to software development by publishing agile manifesto and principles [17]

Manifesto for Agile Software Development

- Individuals and interactions over processes and tools,
- Working software over comprehensive documentation,
- Customer collaboration over contract negotiation,
- Responding to change over following a plan

1.5. Agile Principles

- Customer satisfaction is highest priority
- Accommodate change even late

- Deliver working software
- Collaboration among business people, team members
- Building trust around motivated individuals
- Seamless communication among the stake holders
- Deliver in short iterations
- Build self-sustaining teams
- Focus on continuous improvement

1.6. Scrum in Agile

Today's most of the agile projects fall in any of the following flavours.

Scrum is a popular development method the major activities in this method are [12].

1. Sprint Planning
2. Sprint Review
3. Scrum Meeting

'Product backlog' which is a list of features prioritized by the product owner from which the team will be picking up the Activities for each iteration, Scrum master facilitates daily Sprint meetings with Product owner and team, where in each individuals will be explaining the status of their current task and what they are going to do next day A Sprint is usually spans from 2 –4 weeks, at the end of each sprint, team should have working software.

2. MOTIVATION

Transforming Automation from Sequential Development model to agile development model in automation context.

In Sequential model, validation was done as the last stage of the project, where there was no need of manual rerun entire test execution. Automation of test cases was not a beneficial proposition for the teams; accordingly it was not a primary focus for the teams. There existed a need for maintenance of the product in support phases of the project for which targeted test cases were automated essentially is meant to avoid regressions. This was traditionally done with separate budget and has been treated like a separate project, in a more controlled environment like any other sequential development project.

In order to succeed with automation in agile model, teams have to come up with new mindset, techniques and practices and to break the traditional thinking of automation development and usage.

Below are certain challenges teams are facing in this transformation,

1) Need for Quick Feedback: Unlike traditional models automated tests run more often in agile development, with the help of continuous integration and continuous deployment automation is expected to run frequently and uncover the regressions early.
2) Evolving User Interface (UI): Product being incrementally developed, UI also will evolve continuously. This puts pressure on automation being developed or modified for changing UI. Often, test development has to start before UI components are available. To cater these kinds of needs in agile, automation teams has to come with a novel approach which can make the tests ready the moment UI is ready.

3) Automation Priority: In traditional development model, before automating, the automation engineer looks for a certain criteria like,

- Is the scenario technically feasible to automate in the given technical context?
- Is the test repeatedly runs?
- Is the output of the test can be validated?

Once, these questions are answered "Yes" automation starts and gets integrated into the automation suite, in the agile development are these criteria sufficient and qualifies for automation of the test cases is to what other criteria team needs to look to align with agile development model.

In spite of above challenges, agile scrum teams are to do automation in line with development and achieve the in sprint automation to achieve the Return on investment.

3. SCOPE OF THE PAPER

The scope of the paper is to discuss how a team can speed up the automation development process for new features with in a sprint and ensuring the previous sprint stories are working fine, however the regression tests and analysis of the failures are not discussed in this paper. A real time case has been described here and the methods and practices were implemented in the said environment on the project mentioned.

4. BACKGROUND OF THE ENVIRONMENT, TOOLS & GOALS

Our product is intended for large utility company where so many networks are involved, most of the test case preparation involves preparing test network with required assets on it and performing a series of actions and validating the network and properties of the assets again. Our goal is to reduce the test efforts without compromising on the quality of the product which directly impact the reputation of the organization as well the team.

Hence, we need an approach to test the stories in quicker time and we may need to test multiple times, this lead us to look at the automation as a viable solution to address our needs.

For various reasons we have been using our own custom tool called MAX (tool name has been changed) which is an under UI (User interface tool) and looks like a record playback. By using MAX one can record the actions and in step by step manner and can insert required assertion from the available in the tool. MAX has no reusability since it's a linear framework. However it has a few advantages like it allows to importing and exporting of the networks.

We made a decision to automate 80% of the new "automatable" user stories and ensuring delivered stories are not impacted by the new code changes.

5. CHALLENGES SEEN UPFRONT

Though, we have agreed and decided upon the goals, still we do not have the answers for the challenges we see and the challenges are

5.1. No UI developed upfront

Like any other application the new feature UI is not developed upfront, which is a major hurdle[2] for us since MAX being a record playback tool wit out UI it's almost impossible to proceed further, So test scripts cannot made available hence, teams may need to wait till the developer delivers the feature like sequential model.

5.2. Time needed to automate & Focus diversion for the test developer

Since we do not have any separate team to automate only available testers need to take up automation responsibility also. In the team test engineers have multiple responsibilities like ensuring the feature is working fine by testing manually and investing in automation for further testing. This leads the tester to switching between the contexts which is a challenge for test engineers.

5.3. Test engineers development skills

In the current team, we do not have test developers though they are functional experts and capable of developing automation scripts using popular tools as well MAX .So there is a dependency on the developers at least with code related items.

5.4. MAX defects

Since MAX is still under development, it has a few defects like some controls cannot be identified by the MAX tool; this means though we record a few actions on a few controls no code gets generated hence it cannot replay the actions we performed. And the available list of assertions is not catering all or needs and MAX team cannot help us by adding required new assertions immediately.

6. STRATEGY

There are a few approaches available to name a few, creating a user story for the automation of the story, automating previous sprint stories in the current sprint so that teams and another approach is to make automation mandatory for each story and keeping an item in the definition of done, it means in case if the automation is not completed the story is not complete though its working as per the requirements.

We did not see any advantage of creating a separate story in the same sprint and the above said challenges are still remain as challenges other than a separate story points for the automation which is a point to be consider. The second approach addressed the challenge of UI, since team is doing automation for the previous sprint. However, our goal is more about prevention rather than detection and we would like to leverage the automation completely and want to benefit in the same sprint, and more over by the time we move to the next sprint most of the testing is been done and there could be a tendency of keeping the automation on low priority as the stories have been delivered to the customers.

6.1. Review of the Automation Pyramid

As described above automation pyramid has its own advantages and one perspective we have is duplication of the testing, a particular piece of code or feature has been tested more than once, like unit tests , integration tests and then UI tests, we could not deny the advantages of the pyramid, as it acts as a multiple safety nets and we agreed to take balanced risk and tried to avoid the duplication of the tests, and we agreed to strike balance between UI and Integration tests, and unit tests will be intact. To compliment this where ever is applicable we planned to split the acceptance criteria in such a way each item will be accompanied with how this feature is going to be automated like by UI or Integration tests. And considering the advantages like faster development and easy maintenance of integration tests, we planned to have more integration tests and UI will be used only where UI needs to be tested, and scope of the UI tests are more towards UI rather than functionality.

To trade-off between UI and integration tests we do not have a concrete measure other than expert review, though code coverage can be used since it’s a decision to be made before writing code. However we agreed to change the automation method if required, as per the teams’ suggestion.

Mike Chon’s mentioned [19] about the focus should be service layer, and did not mention about the duplication of the tests, we are focusing much more on duplication of the tests along with other factors.

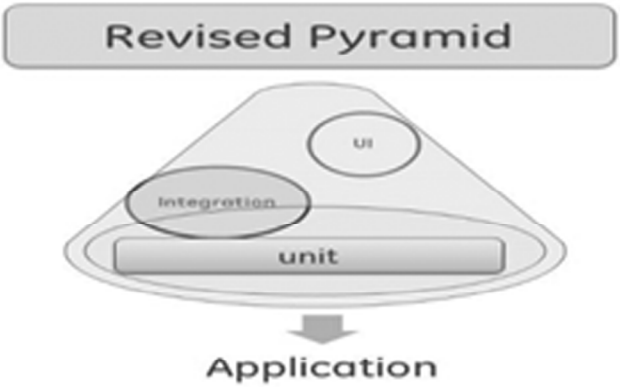

Figure 4: Revised Pyramid

So based on the above thought process we decided to have the automation in the same sprint with spiral approach.

7. PLANNING

We made sure our objectives and thoughts about the in sprint automation with all the team members and made them align to the same thought process as planning is most critical for any task we were cautious from the beginning and made sure we give a thought of the automation strategy at every point of the process.

We embedded our automation in the sprint planning of the project with three steps similar to the development story

As mentioned above, we first defined “Automatable”, for us as we targeted to automate minimum 80% of the automatable we need this definition. And the main

We concluded a particular user story is automatable provided a story is declared as automatable if the below questions are answered YES

- Is it repeatable with in the sprint and after the sprint as a regression?
- MAX support to the feature, what it means here as we discussed earlier MAX is not supporting a few controls if the new story involves un supported controls or not
- As we adopted incremental development of the automation script, in the planning itself we are forced to identify whether it’s a separate individual script or part of the script or having an integration test.

❖ If it’s part of the script whether there will be any definite deliverable? With respect to automation.

❖ If its Integration test then what part or whole

Once we understand automatable, estimated the automation efforts and added to the user story points, so the current user story delivery includes automation script, this approach deviates from agile core principle

Figure 5: Planned process for an automation script

point only those which are required to customers, and in way it inflates the actual story points and in future this reference will always come with automation embedded into it which is not acceptable by our team. So we agreed up on the not changing the points to the story however include the automation is part of definition of done.

We generally would not have one functionality end to end developed in one sprint, there is always addition for the existing functionality, in this case automation can be done up to that extent and keep on adding the new feature as and when it comes. For these kinds of cases, have an overarching MAX script to call these scripts in batch.

Since, MAX supports importing and exporting of the networks, we planned to leverage this feature to the maximum, as part of it we identified a few basic networks which are most commonly used and planned to extend them to cater the needs of current user story requirement

As part of pre-defined networks, team spent some time around 3 days for 2 people and prepared during iteration 0.

In planning we addressed above mentioned challenges up to some extent. However, since test engineers lack of coding language team did not opt for integration level tests to the optimal, whenever accepted they ensured developers help this is a challenge we could not address completely.

Below are a few snapshots from our project, where we marked the stories with basic information regarding the automation. Like are we automating if so whether it is an individual script or part of automation script, if no what's the reason.

In the above snapshot, illustrated three stories where in one story as not automatable since no reusability and rest of the stories qualified for automation and part of the master script.

8. RESULTS AND DISCUSSION

During the execution, we implemented what we discussed above.

While writing the user story acceptance criteria, we mention test approach and maintain detailed acceptance criteria. With this approach we stared adding a new section called "Test Approach" for each story and under this section we mapped how each of the acceptance criteria will be tested manual or automated and how

Figure 6: Notes of stories

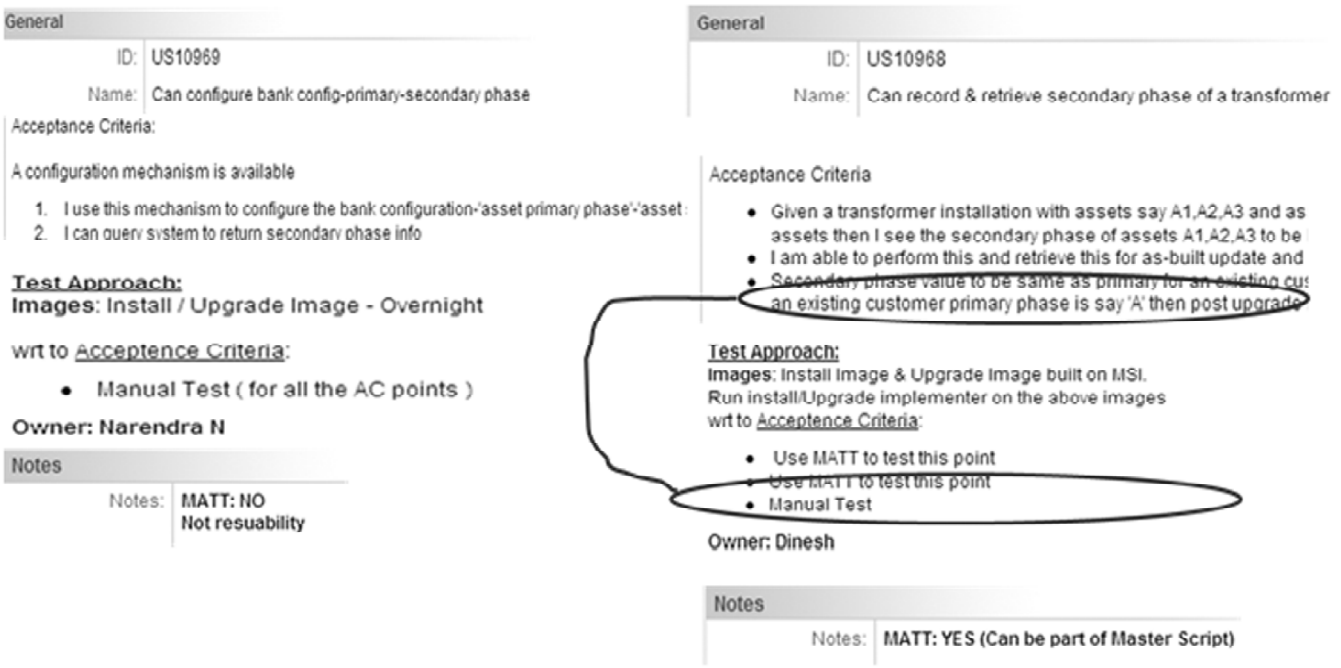

Figure 7: Illustration a few sample user stories

Below is a snapshot of two user stories from the project, one user story marked as "no automation" was planned since it's not meeting the automatable criteria.

During sprints we have a process to track the stories and the status along with the automation. We have maintained stringent review process and it involves the below

Engineer1 writes the test case in detail

Engineer2 reviews the test case and the same will be automated, this enables the quality of the test case as if the same person has to automate, automation task can influence the quality of the test case, since it's been reviewed by and automated by a peer, it keeps the focus of the engineer on the task.

Engineer 1 again reviews the test script and starts using it, we have used tracking tool which was updated promptly as per the status.

Below is a snapshot representing the tracking of the review and automation status

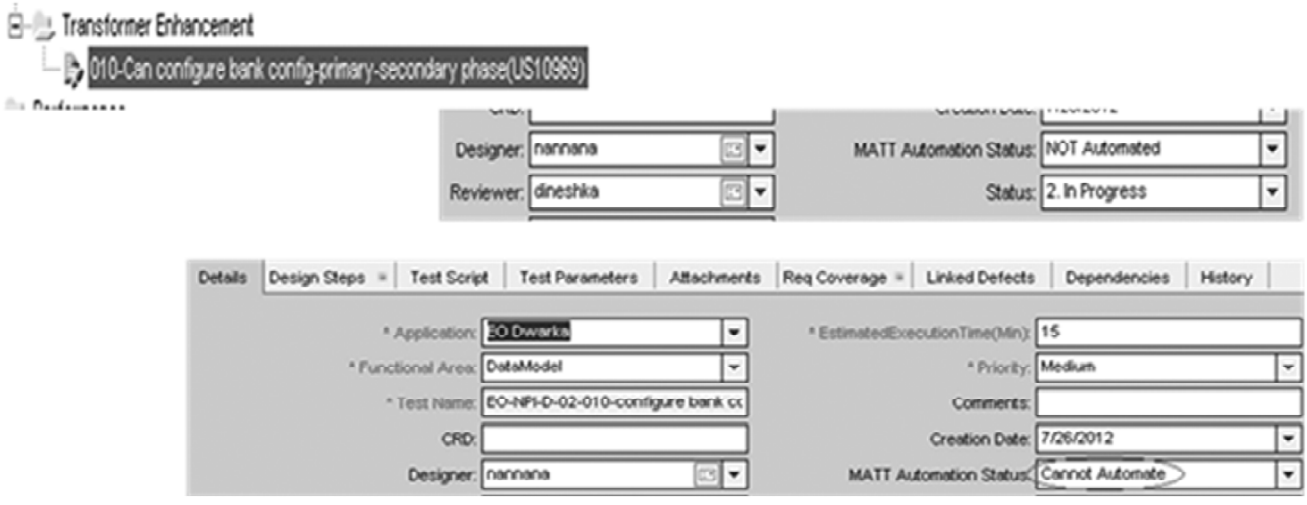

Figure 8: Automation tracking in the project tracking tool

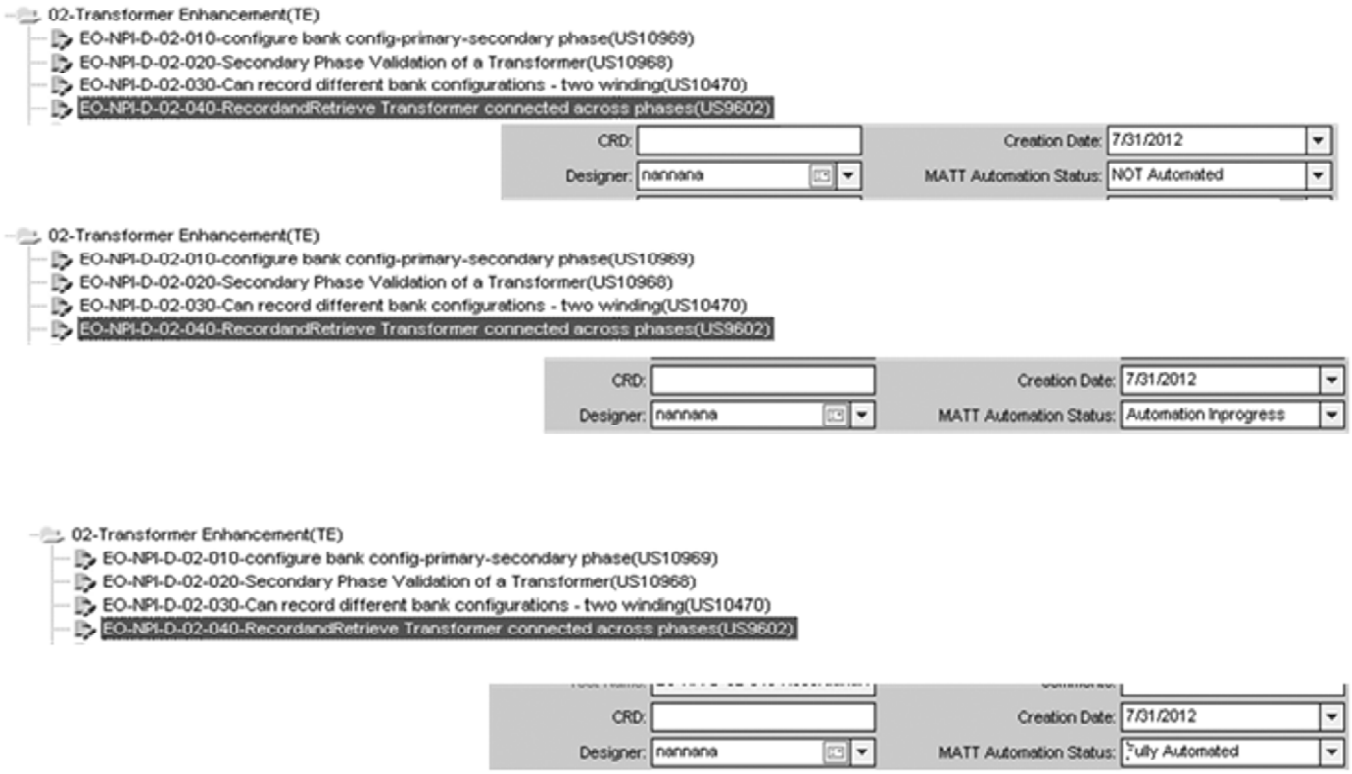

Figure 9: Test case life cycle in the sprint.

During the sprints, the status of the automation script moved from “NOT Automated” to” Automation in progress” and then to “Fully Automated”.

Automation metrics have been gathered for every sprint and published with sprint goals to all the stake holders so that everyone is aware of the status of the automation.

During our script development, we observed no of stories need not match with no. of automation scripts, it can be n to n relation like in some cases a single automation script can serve the purpose of two stories with a little change and vice versa single story can be automated with two or more scripts, the same has been captured in the below snapshot where no of automated scripts are less than no of automatable.

The below table mentioned table consists of first two sprint automation status.

Table 3
Represents the data of automation progress

Sprint #	Start Date	End Date	# of User Stories	# of Automatable	# of Automated	# of MAX scripts
44	25th July	7th August	3	2	2	2
45	8th August	22nd August	5	4	4	3

Complete data the whole sprint is for 8 sprints and data for 8 sprints here the data, first bar chart represents the automatable & accepted scripts versus actual Automated.

The data in the below table represents sprint wise details about accepted, cumulative target and actual automated and cumulative actual automated numbers.

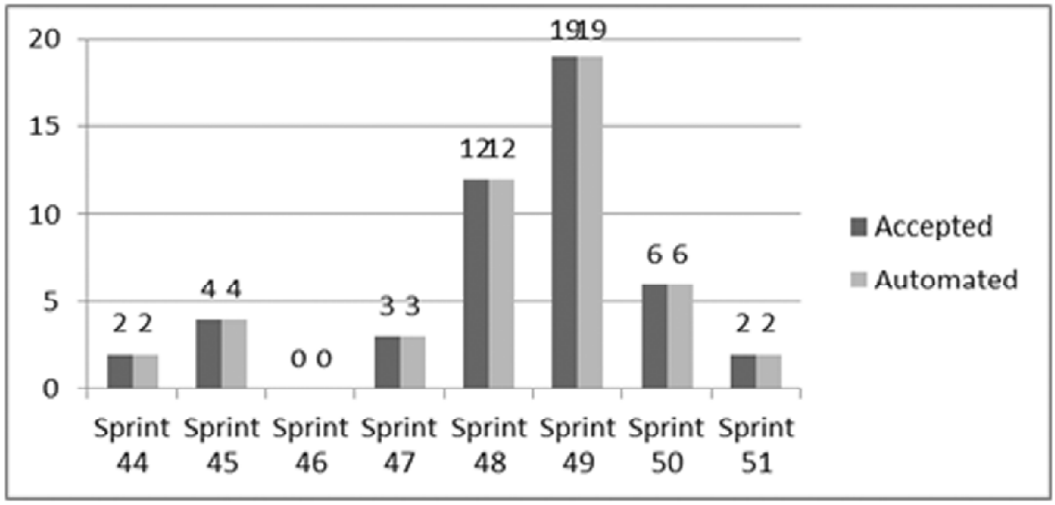

Figure 10: Complete data representation of accepted vs. automated

Table 4
Sprint by sprint progress of accepted and automated user stories

Sprint Number	*Accepted*	*Cumulative Target*	*Automated*	*Cumulative Actual*
Sprint 44	2	2	2	2
Sprint 45	4	6	4	6
Sprint 46	0	6	0	6
Sprint 47	3	9	3	9
Sprint 48	12	21	12	21
Sprint 49	19	40	19	40
Sprint 50	6	46	6	46
Sprint 51	2	48	2	48

We could save around 800 man hours of test efforts on the overall project test efforts, however this cannot be treated as a direct savings as with manual efforts team does not tests all the stories and may test only those are impacted and test high level only. However, with this approach we eliminated the risk based testing and avoided impact analysis as we are running all the tests all the time since most of the tests are automated, a detail data has been provided at the end of this paper un appendix A

The below graph depicts, the trend of cumulative efforts versus cumulative savings.

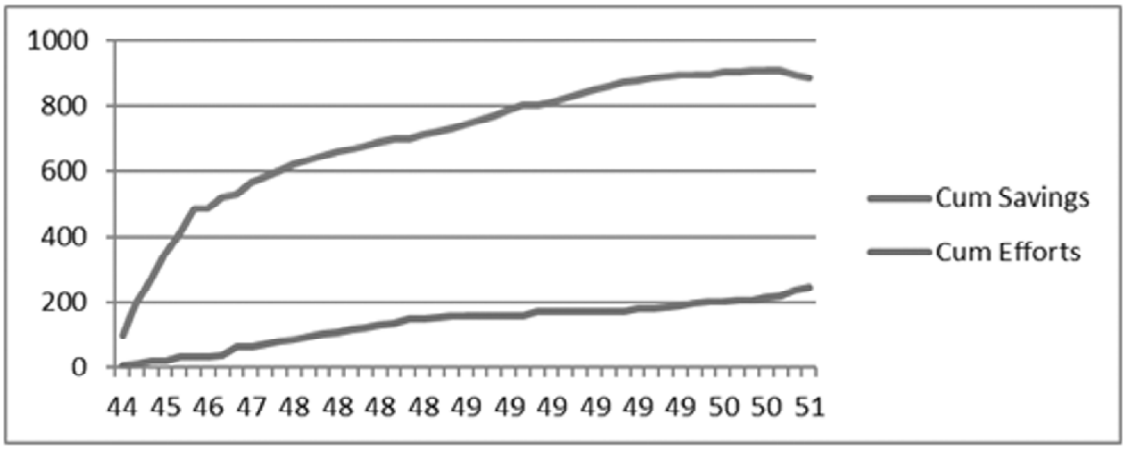

Figure 11: Trend graph of efforts spent versus efforts saved for the whole project

Along with the above tracking, we also track, publish the below mentioned automation metrics for automation progress[10] in order to evaluate our self that how well we are doing with respect to achieving the in sprint automation

$$AP(\%) = \frac{AA}{ATC} = \left(\frac{\#\ of\ \ actual\ \ test\ \ cases\ \ automated}{\#\ of\ \ test\ \ cases\ \ automatable} \right) per\ sprint$$

AP = Automation Progress

AA = # of actual test cases automated

ATC = # of test cases automatable

As sprint goes on we published automation progress against the sprints [10], generated the below graph and informed all the stake holders in order to keep them on top of the information.

$$TP(\%) = \frac{TC}{T} = \left(\frac{\#\ of\ \ actual\ \ test\ \ cases\ \ completed}{time\ (sprints)} \right) per\ sprint.$$

TP = Test Progress

TC = # of test cases completed

T = some unit of time (sprints which is two weeks)

The purpose of this metric is to track the test progress and evaluate our status with respect to the goal which is 100% in sprint automation

9. ACTUAL VS PROJECTED SAVINGS

Actual savings and projected savings are vital metrics to calculate the return on investment or calculating the savings of the test efforts we put on.

In any software project, we will have three categories, first one is regression which includes the previous features which are already there from the beginning of the project we also call as legacy features and

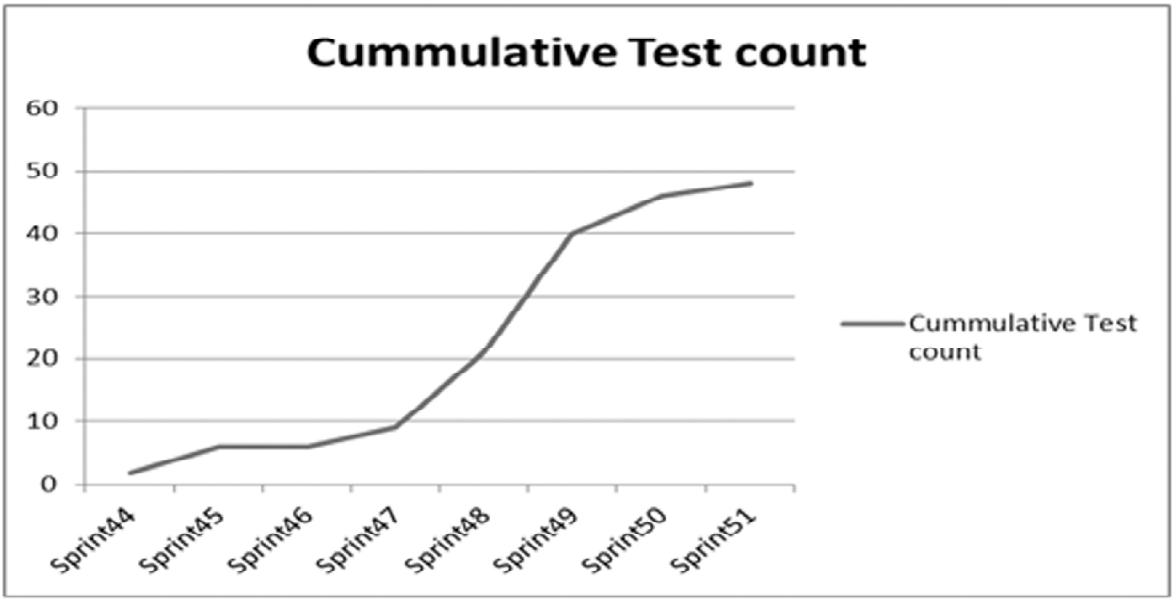

Figure 12: Curve graph of tests automated over the sprints.

second is new release delivered stories this means stories delivered during this release but does not include current sprint stories, as mentioned in the scope we are considering only in play user stories and delivered stories. Let's understand this analogy with an example. In the below example let's assume that we are at Sprint 47, so at this point stories US9602, US10968, US10476, US9604, US10690, US10688 will become the delivered stories for this release, and since we are in the 47th sprint stories US11312, US10671, US10707 will become the in play stories, so at the end of the sprint 47 team needs to ensure delivered stories did not break and in play stories are working as per the requirements.

So, it's checking for any regressions caused due to code change during the new feature implementation, of course need to ensure legacy features are intact which we are not discussing.

As sprint on sprint new features /user stories kept adding and testing need to be done or the new stories and previously delivered stories in the same sprint. The below graph represents how manual test effort goes sprint on sprint because of the no of stories to be tested are more

The below table represents the test efforts for each sprint, as explained above sprint on sprint effort increased and however the total comes around 100 man hours for a 16 week project. And this does not include the automation efforts we kept.

Table 5
Sprint by sprint user stories

Sprint #	Story
44	US9602
44	US10968
45	US10476
45	US9604
45	US10690
45	US10688
46	
47	US11312
47	US10671
47	US10707

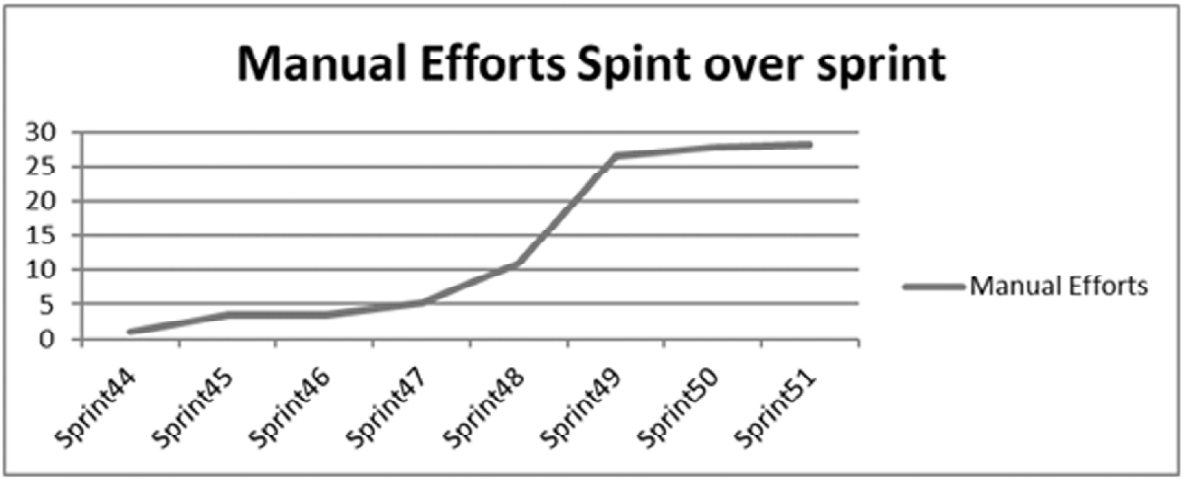

Figure 13: Curve graph of manual test effort for each sprint.

Table 6
Sprint by sprint Manual Efforts.

Sprint No.	Manual Efforts (Hrs.)
Sprint 44	1
Sprint 45	3.5
Sprint 46	3.5
Sprint 47	5.083333333
Sprint 48	11.08333333
Sprint 49	26.58333333
Sprint 50	27.78333333
Sprint 51	28.18333333
Total	106.7166667

107 hours are the actual savings we are achieving by having automation in place, and this 107 man hours does not include automation efforts, if we consider the automation efforts which are around 198 ROI becomes negative and this is true if are just replacing the manual efforts with automation, and at the same time we are ignoring intangible items like quality and confidence. The availability of the automation scripts encourages us to use more number of times as no extra explicit efforts are required since more runs and higher probability of uncovering the defects and also increases the confidence build over build before the delivery.as per the above calculation we run the manual test once in every sprint but with automation we run the same tests every day, and this increased the number of run exponentially and also the savings, along with this makes us ready to take up next set of features and deliver with quality.

10. CHALLENGES IN IMPLEMENTING

We had many challenges during this project and good learning.

- Accommodating application change after script development is time consuming and tedious and lesson learned not to modify the script rather develop a new one.
- Day by day scripts kept growing and difficult to analyze the failures along with regression scripts
- Reviewed the automation strategy to speed up and agreed to develop scripts on development environment and verify them on test environment, this approach will help Script developer to start scripting early.
- As MAX is also in development had to progress with limited features, though we have considered this risk and mitigated with custom codes, developers availability was a challenge which turned out to be significant effort which was not expected.
- Heavily dependent on network files for test bed preparation, issues with network lead to script failure hence maintenance time for the script got increased unexpectedly.
- Could not identify more tests at integration level due to lack of programming skill of test engineers and so could not fully implement our revised test pyramid.

11. CONCLUSION

As discussed, Spiral approach would be a considerable strategy to meet the needs of evolving UI, keeping an overarching script to call in small scripts would help, thus achieving and implementing modular approach would pay off. Though we have no support of reusability, we made script itself as a reusable component and incrementally integrating as we go along.

This also made us realize, in sprint automation can be achieved with basic record and play based tools, and do not need any sophisticated tools. Key element driving the strategy is to identify right automatable tests, and also emphasis of automation in the definition of done.

And enhancing test engineers programming skills or having test developers also adds value and implement revised pyramid strategy successfully

Availability of automation enabled us to use automation more often, which resulted in increased savings when compared to running the manual test execution only once.

REFERENCES

[1] L. Crispin and J. Gregory, Agile Testing. Don Mills: Addison-Wesley, 2009.

[2] Adam Geras, Of Schools and Tools, An Agile Tester's Resource Kit. ACM, 2009

[3] Martin fowler, "The New Methodology" http://www.martinfowler.com/articles/newMethodology.html

[4] F. Elberzhager, A. Rosbach, J. Münch and R. Eschbach, "Reducing test effort: A systematic mapping study on existing approaches," Information and Software Technology 54, pp. 1092–1106, 2012.

[5] K. Karhu, T. Repo and K. Smolander, "Empirical Observations on Software Testing Automation," International Conference on Software Testing Verification and Validation, 2009.

[6] T. Wissink and C. Amaro, "Successful Test Automation for Software Maintenance," in 22nd IEEE International Conference on Software Maintenance (ICSM'06), 2006.

[7] The Art of Software Testing By Glenford J. Myers, Corey Sandler, Tom Badgett

[8] J. Kent, "Test Automation From RecordPlayback to Frameworks," http://www.simplytesting.com/, 2007.

[9] M. Fewster, Software Test Automation: Effective Use of Test Execution Tools, Addison-Wesley Professional, 1999.

[10] Thom Garrett IDT,LLC, "Useful Automated Software Testing Metrics"

[11] McConnell, Steve (2004). Code Complete (2nd ed.). Microsoft Press. p. 29. ISBN 0735619670.

[12] Prof.JamesNoble,Dr.StuartMarshall http://nzcsrsc08.canterbury.ac.nz/site/proceedings/Individual_Papers/pg218_Agile_Project_Management.pdf

[13] Myers, Glenford J. (1979). The Art of Software Testing. John Wiley and Sons. ISBN 0-471-04328-1.

[14] Gelperin, D.; B. Hetzel (1988). "The Growth of Software Testing". CACM 31 (6): 687–695. doi:10.1145/62959.62965. ISSN 0001-0782

[15] L. Hayes, The Automated Testing Handbook, Automated Testing Institute, 2004.

[16] P. Laukkanen, "Data-Driven and Keyword-Driven Test Automation Frameworks," Helsinki University of Technology, Software Business and Engineering Institute, 2007.

[17] The Agile Manifesto is online at http://www.agilemanifesto.org/

[18] Cohn, M., & Safari Books Online (Firm). (2010). succeeding with agile: Software development using Scrum. Upper Saddle River, N.J.: Addison-Wesley