Software Test Automation Practices in Agile Development Environment: An Industry Experience Report

Eliane Figueiredo Collins
Nokia Institute of Technology
Manaus, AM, Brazil
eliane.collins@indt.org.br

Vicente Ferreira de Lucena Jr.
Federal University of Amazonas
Manaus, AM, Brazil
Vicente@ufam.edu.br

***Abstract*— The increased importance of Test Automation in software engineering is very evident considering the number of companies investing in automated testing tools nowadays, with the main aim of preventing defects during the development process. Test Automation is considered an essential activity for agile methodologies being the key to speed up the quality assurance process. This paper presents empirical observations and the challenges of a test team new to agile practices and Test Automation using open source testing tools integrated in software projects that use the Scrum methodology. The results obtained showed some important issues to be discussed and the Test Automation practices collected based on the experiences and lessons learned.**

***Keywords*-test automation; agile testing; testing tools; software testing**

I. INTRODUCTION

In recent years, Software Engineering has been adapting to the evolutions occurring in the global software development scenario in order to alleviate high cost, high complexity, market time, maintenance difficulties and end-user satisfaction. In this context, two evolutions could be highlighted and they are the focus of this paper: the Test Automation and the Agile Methods.

Depending on the system phase, software testing can be executed manually or automatically. According to Dustin et al. [1] the definition for Software Test Automation means to automate software testing activities including the development and execution of test scripts, verification of testing requirements, and the use of automated testing tools. There are many reasons why software tests should be automated. When compared, manual testing is time consuming while Test Automation increases efficiency for repetitive steps to reproduce system functionality, especially in regression testing, where test cases are executed iteratively and sometimes incrementally after making changes to the software [2]. Automate all test activities during development phase can take more than 50% of the overall project effort [3].

The agile software development process stands out mainly because of its ability to rapidly accommodate changes in the original requirements and prioritize the development of functionalities through executable codes instead of extensively written documentation. It also provides quick responses to changes and collaboration with the customer rather than following rigid plans and contract negotiations. The Scrum, as agile methods, is an important practice in software development, with characteristics of relatively flexible organizational system, timely and interactive feedback, objective-oriented flat management and effective participation of members [4]. It can solve the difficulties faced by traditional open methods, some to a large extent and it can also be used to manage and control software and product development.

Agile processes are iterative, so the test activities have to follow the iterations which need to be executed fast and efficiently leading to the use of automated testing tools, quick responses to the requirement changes and interact with the development team [5].

Crispin and Gregory [6] identified the automation problems in projects and according to their experiences in agile teams, the issues found that influence the success of Test Automation are:

- Programmers' attitude, some programmers don't bother to test much because they have the QA team as a safety net to catch bugs;
- The "Hump of Pain", means the learning curve for testers to learn about tools and code;
- Initial investment;
- Code that's always in flux, interface changes turn the GUI automation not viable;
- Legacy systems, legacy code isn't designed for testability;
- Fear, testers that don't have programing background and programmers that don't have experience in testing;
- Old habits, manual regression tests instead of automating the regression suite;

This paper presents experiences and results of the implementation of Test Automation using open source testing tools in projects which used the agile development process Scrum. For confidentiality reasons, the software projects described in this paper will be called Project 1 and Project 2 respectively.

In section 2, the agile development with Scrum and a brief on Agile Testing is explained. Section 3 shows details of case studies, the projects characteristics, testing process and common problems. Section 4 shows the results obtained with the experience collected and the lessons learned. Finally, section 5 presents the conclusions and further steps.

II. BACKGROUND

A. Agile Development Scrum

The Scrum is illustrated in Figure 1, the functionalities are to be implemented through a list called Product Backlog. Each sprint has two week duration (iteration). There is a Sprint Planning Meeting where items on this list are prioritized by the customer representative (Product Owner). The team defines the features which may be accomplished within the iteration. This list planned for the next iteration is called Sprint Backlog [3]. During the Sprint, the activities are broken down into tasks that are observed by the team through a daily meeting which lasts a maximum of fifteen minutes where tasks impediments are identified.

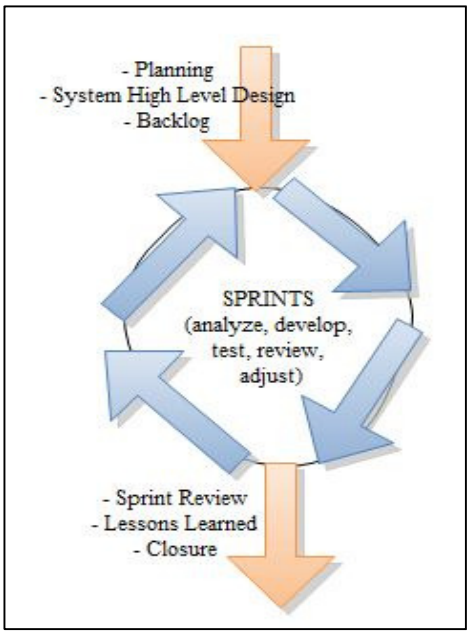

Figure 1- Scrum methodology

The Scrum methodology has three key roles to run the process [7]:

- Product Owner (PO): This role is responsible for communicating the product vision to the scrum team and must also represent the customer interests through requirements, business rules and priorities;
- Scrum Master (SM): Link between the team and PO, but does not manage the team. Scrum master works to remove the obstacles that are blocking the team tasks from achieving the sprint aim;
- Team Member: Team liable for completing the work, cross-functional and empowered to make plan, effort estimates and take technical choices.

B. Agile Testing

As Crispin and Gregory claim [6], agile testing does not just mean testing on agile projects but testing an application with a plan to learn about it, let the customer information guide the testing in line with agile values working software which responding to change. According to the authors, for a successful agile testing approach the following are the key factors:

- Look at the Big Picture;
- Collaborate with the Customer;
- Build the Foundation for Agile Core Practices;
- Provide and Obtain Feedback;
- Automate Regression Testing;
- Adopt an Agile Testing Mindset; and
- Use the Whole Team Approach.

Test Automation is considered the key of successfully agile development and the core of agile testing. Agreeing that [6], different tests have different purposes, they show a diagram of testing quadrants describing how each quadrant reflects different reasons to test, as presented in Figure 2.

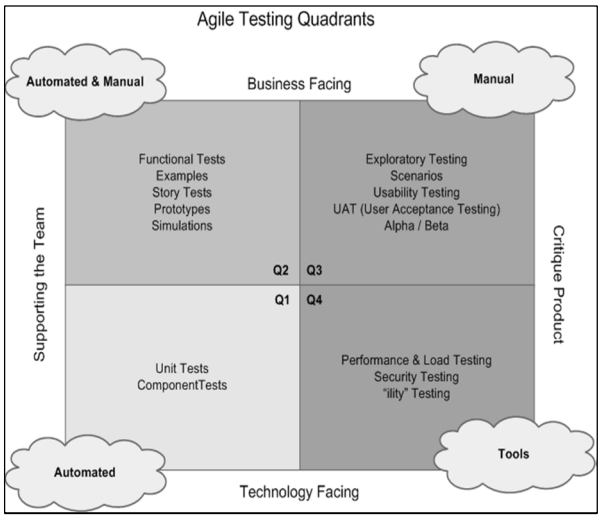

Figure 2. Agile Test Quadrants[6]

- **(Q2)** Business Facing tests in Support of Programming (Business requirements testing – Does the code do what it should?)
- **(Q3)** Business Facing tests to Criticize the Product (Business defect testing – Does the code do something it shouldn't? Are there missing requirements?)
- **(Q1)** Technology Facing tests in Support of Programming (Technical requirement testing – Does this method do what the developer intended?)
- **(Q4)** Technology Facing tests to Criticize the Product (Technical defect testing – Are there leaks? Can it handle a load? Is it fast enough?)

Traditional software testing focuses almost exclusively on the right side, the functional tests are made to criticize the product but not playing a productive part in supporting the creation of the product. Moreover, it is involved late in the development to detect failures but not to prevent them.

In agile testing, the team is not only involved in identifying, but also in preventing failures. Thus, agile testing is a challenge for testers that used to work in traditional waterfall projects mainly because they do not have to wait for system delivery at the end of the project to start the test activities but they need to be proactive and start the test tasks from the very beginning of the project together with the developers as one team while bearing in mind that success in this challenge requires experience, overcoming barriers and mind set.

III. ENVIRONMENT

A. Software Projects Chacracteristics

During a period of one and a half year two software projects had been conducted, all using Scrum as the development methodology for management. In the context of a research institute in Manaus/Amazonas/Brazil they served as study cases to apply Software Test Automation and collect the practices and lessons learned.

First, in Project 1, there was focus on usability. The customer was able to use a web dynamic interface easily to register campaigns, users and ads attaching images, videos and texts without difficulty and with good performance. This project had a development team composed of one Scrum Master and 3 full-time developers.

The test team had one Test Leader half-time and two Testers full time. Project 1 was web platform developed using PHP programming language, MySQL database, Apache server and IDE Eclipse. The system was simple and composed of 8 screens (4 forms to register campaigns and users and 4 screens to search and generate reports).

Project 2 was a web project to attend the factory. It automates tasks according to procedures to control materials and production costs. The main goal was the usability of interface for specific users, performance of system responses and the accuracy of the data to be manipulated during the system calculation.

This system had high complex rules in two web interfaces: one counter user through mobile browser and an administrator user through computer browser. Java language was used for development environment, JBoss server, Oracle database and Eclipse IDE. The scrum team for Project 2 was composed of 4 developers, one Scrum Master, one Test Leader half-time and two Testers full time.

These projects initially followed the same test process, but at specific times during their execution, the Test Automation structure must be modified to improve the test coverage and the goal of the projects.

B. Testing Process and Tools for the Scrum Projects

The Test team in accordance with agile principles was integrated to the project team. There wasn't any formal separation between the Testers and the Developers. The Testers were participating in Scrum ceremonies (sprint review, daily, retrospective and planning meetings). Their responsibilities basically was to plan test cases through the stories described in the Product Backlog, specify the acceptance criteria, and use automated test tools to speed up the execution activities of each sprint. The main tools recommended by the test team to be used to automate test tasks in the projects were:

- ***TestLink*[1]:** used to manage test plans, write test cases, and report test executions. This tool acted as an editor and organizer of test cases, storing all information allowing updating and creating test builds easily. It facilitated the documentation of reports, and also the controlling of the test execution versions [8];
- ***Mantis Bug Tracker*[2]:** through this tool, which was already used in the institute, the tester registered the defects found, sent them to the developers and controlled the lifecycle of each defect. The friendly interface facilitated communication among teams.
- ***Subversion*[3]:** it was used to share and manage the system code, documentation and information among the project team's members.
- ***Jmeter*[4]*:*** Open Source Tool for web performance and stress testing generating result reports.

The tasks of test processing during a sprint were incremental and iterative, according to the agile testing, they began with the development and did not wait for system delivery. The Test team was new to scrum and agile testing. They had experiences testing waterfall projects. They may be proactive doing verification tasks for the period of the development using automated tools. To avoid the Cascade effect, the sprint could not be delivered for tests at the end of sprint. The Test Process flow is represented in Figure 3.

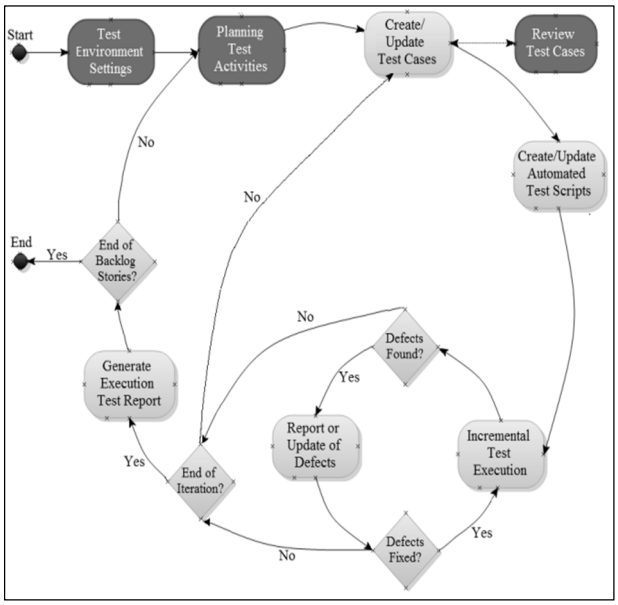

Figure 3. Testing Process

[1] TestLink URL: http://www.teamst.org/
[2] Mantis URL: http://www.mantisbt.org/
[3] Subversion URL: http://subversion.tigris.org/
[4] Jmeter URL: http://jmeter.apache.org/

The dark rectangles represent tasks to be done by the Test Leader and the lighter rectangles were tasks to be done by the Testers. The Test Leader was also responsible for monitoring all tasks in the process. The task Incremental Test Execution included execution of manual exploratory tests and automated regression tests.

For web projects 1 and 2 described, developers used for unit testing the frameworks: *PHPUnit*[5] and *JUnit*[6] respectively. The tool of choice for functional GUI test automation was Selenium *RC and IDE*[7].

Selenium uses the approach record-and-play support web applications for the browser Mozilla Firefox. *Selenium IDE*[8] can record the user actions in the browser, creating test scripts in several programming languages and executing them later [9]. The *Selenium Java API* used in the project allows running the test scripts in other browsers such as the versions of Internet Explorer.

IV. Study Cases Applying Test Automation

Different Test Automation practices were used during each project to achieve their goal. The scrum iterations allow the project team to use different approaches until it finds the right way to work. So, when some tool or method didn't work well in the sprint, the team can try something different in the next iteration to improve the process. Practices adopted for test automation and the common problems found for each project are outlined below.

A. Project 1

During the first stage of Project 1 (Sprints 1 to 3), the team decided that developers should perform stories of new functionalities and unit testing using PHPUnit throughout the development process. Continuous integration (CI) process for infrastructure reasons and developers' attitude could not be implemented. Developers were focused on new features and Testers should automatize functional tests through GUI for regression tests using Selenium RC and explore the system through manual tests. The Stress tests should be realized after all the functionalities were implemented at the end of the project. To apply agile test automation in an environment without continuous integration was a big challenge.

This was the first attempt by the team to automate unit tests and functional tests in an agile project. Internally the project team was divided into Developers and Testers. The Developers weren't performing effective unit tests because they were focused on new features and confident that the Testers will find all the system defects. So, many code defects as null pointer exceptions, screen crashes and illegal exceptions were found during functional system tests.

In consequence, the greater part of the code needed to be modified for the next sprint causing technical debt.

Moreover, the task to automate functional GUI tests in the current story of sprint presented problems because the GUI interface of system wasn't stable enough, Testers had no strong java programing background to improve the automated suite, they were trying to automate using the record-and-play approach for all test cases planned, including for interface issues, but the system interface changed during the sprint and the automated test scripts had to be recorded and written again. The lack of time in the sprint to finish the automation tests and non-synchronization tasks with development resulted in back to manual execution of regression tests. Testers had technical debt too.

After identifying problems, the project team started refining the strategy and changing the automation activities to the second stage of the project (Sprints 6 to 9). The Unit testing performed had to be reviewed by Testers and achieve 100% coverage of business rules. It was an item for story acceptance criteria in the backlog stories.

To alleviate automation test problems, Testers decided to perform the automation with the support of Developers, automatizing just the acceptance tests of past sprint which interface was considered stable. The exploratory manual tests should be run for current sprint stories. Developers were helping with test activities programing the automated test suite. The team was working together supporting each other in test activities and programing.

Other improvements were implemented at the end of the project. The automated regression test suite with Selenium was integrated with *TestLink* using the *Testlink API Client*. This API allows the automated test suite in java to access the *Testlink* and register the Selenium execution results automatically. *Testlink* was integrared with *Mantis* also, allowing traceability of defects and test cases.

Thus, when Testers were executing regression tests automatically, results were registered in *Testlink*, facilitating the generation of test reports, providing quick feedback of tests and spending the time more efficiently to run manual exploratory tests to find usability and performance defects.

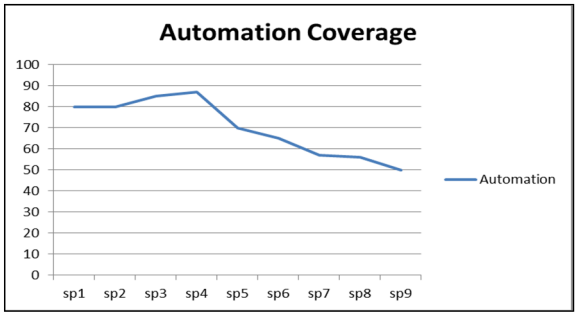

Figure 4. Project 1 Automation Test Coverage

[5] PHPUnit URL: www.phpunit.de
[6] JUnit URL: www.junit.org
[7] Selenium URL:www.seleniumhq.org

Figure 4 shows *Selenium* GUI test automation coverage of functional test cases. Initially Testers tried to automate all test cases planned for the sprint and web interface achieving 87% of coverage. During the sprints with interface changes and new test approach the automation cover decreased by 50%. This happened because the team automated just the acceptance tests for stable interface. Compared to Figure 5, it is possible to realize that defects found along the sprints increased until down to the end of project. It proves that high automation coverage for GUI didn't mean finding many defects, because Testers were very busy trying to automate a lot of GUI unstable tests without value for the project. When Testers decided to automate just what was necessary for feature acceptance, the automation coverage decreased, but with time for exploratory tests, they found more defects and did tests that added value to the project.

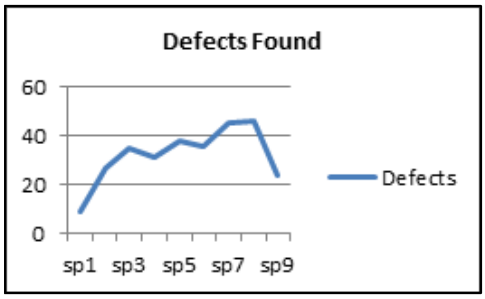

Figure 5. Project 1 Defects per Sprint

Stress Tests using *Jmeter* tool and security tests as planned were executed at the end of the project (Sprints 8 and 9). The problems identified during this phase impacted the entire system and many parts of the code and even test scripts had to be reworked. The functionalities and GUI were according to backlog stories and acceptance criteria but the non-functional system requirements and vulnerability tests had failed. So, non-scheduled tasks of reworking appeared close to delivery of the project.

B. *Project 2*

The structure of Project 2 had some improvements compared to Project 1. Continuous Integration (CI) environment was configured in the first sprints (1 to 3) using the open source tool *Hudson* [9]. Automatically the CI should be able to compile the code, execute unit tests, accept GUI tests (*Selenium*) and deploy the versions of the system to the servers: Development, Test and Production.

Due to the experience of Project 1, Testers had adopted the strategy of startup system basic load tests and vulnerability tests as soon as possible using *Jmeter* test tool integrated in the CI environment. Thereby, it ensured that each release generated for Project 2 was covered by unit tests, GUI acceptance tests, basic load tests and security tests according to the customer requirements specification in order to minimize risks of problems and rework in the system at the end of project as occurred in the previous experiment. The reliable system version could be available to customers at any time in the production environment.

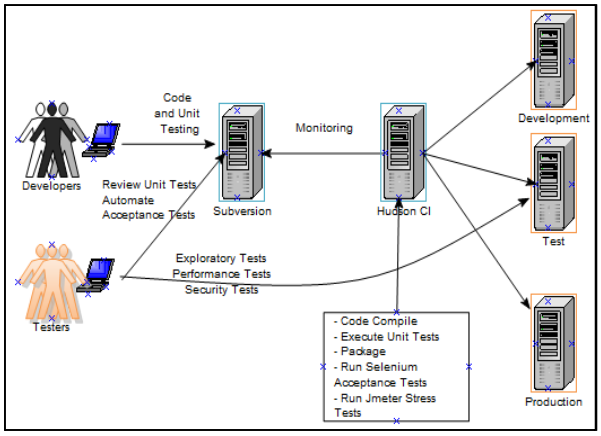

Figure 6. Automation Structure with CI

Figure 6 represents the automation structure using CI. Testers and Developers committed to sending the daily code, unit tests and automated acceptance tests (*Selenium* and *Jmeter*) through *Subversion*. *Hudson* monitored the *Subversion* changes and started the CI process. Developers could easily run acceptance tests ensuring a functional system and Testers could execute manual exploratory tests, security tests and usability tests on Test environment.

Test problems were found during sprints (sprint 5 to 10). The customer realized that interface, performance and usability were working well, but the data accuracy wasn't right. This happened because the tests were performed for the code layer and GUI layer, but functional tests for complex business roles may be executed for the layer behind the GUI.

Note that for this project the base of automation was unit tests and acceptance tests, but to ensure data accuracy for complex business rules, it wasn't enough, so integration tests had to be done to validate it. To solve this problem, Testers and Developers implemented tests using *Fitnesse Test Tool*[10].This tool can be used as wiki for input data and fixtures implemented in java. It was also integrated with *Hudson*, team members easily implemented tests stubbing database data, the test execution were behind GUI identifying the integration problems earlier.

Quick feedback in this project were observed because *Hudson* displays real time reports of the tests and an e-mail was sent to the project team to do the necessary corrections in the project. The CI environment made it possible to take automation testing to another level of efficiency and endorsed the team's integration.

Besides, the regression tests suite of Selenium tests were large and were run in order of creation of every sprint, after sometime it was observed that the first tests didn't reveal defects. This happened because the first features were stressed with test cases several times, so they were stable.

[9] Hudson URL: http://hudson-ci.org/

[10] Fitnesse URL: http://fitnesse.org/

But the tests for new features reveal defects caused by code changes. So, in conclusion, prioritizing the regression test for a large test suite reveals defects earlier.

Project 2 had 20 sprints, were the automation for unit tests, integration tests, non-functional tests (performance and security) and GUI acceptance tests covered respectively 70% of code, 100% of business rules, 100% of non-functional requirements and 40% of GUI acceptance tests. The defects found per sprint before integration tests varied between 45 and 48, after integration tests the defects found per sprint decreased for 29.

V. Results and Lessons Learned

In this experience on conducting Test Automation in agile software projects, we could observe it can work very well when the project team has the patience and persistence to find the right way to work. However, some issues need to be managed to avoid the risks introduced by the difficulty of apply agile practices in an inexperienced project team. Thus, we have identified some key lessons that we have learned, minimizing the impact of automation risks:

A. *Collaboration is an Essential Factor for the Success of Test Automation in Agile Projects*

All main research works in the agile methods field indicate that team collaboration is an important aspect to achieve success in an agile project, and we could also confirm these claims for Test Automation. Some practices were essential to reach this success:

- Involve Testers in backlog review, design interfaces, configuration environment and architecture system discussions are very important to avoid separation between developers and testers. Maintaining the unity of the project team encourages cooperation.
- Ask for help when a task requires knowledge from another area. For example, in Project 1 Developers helped Testers to improve the *Selenium* automation test suite and Testers supported Developers reviewing unit testing to achieve test coverage.
- Test Automation environment is not a responsibility just for Testers. All Team members have responsibility in setting a test automation environment. Pair programming can be a good exercise to integrate the project team. Pairs of Tester and Developers can be useful for some tasks like setting environments, code inspection, review and program scripts.

B. *Fit Testing Tools with Test Strategy and Agile Method*

Choosing the right tools to be used to automate tests is very important. Sophisticated tools are not always the best choices for Agile Test Automation and the project team has to know the limitations of free test tools well and consider the learning curve and time spent on configuring the test environment.

It is important to keep in mind that if one tool or technique is not working or adding value to the project, in the next iteration, the project team can start trying something new or fit the tools with the project needs as happened in Project 1.

The tools to be chosen in the agile environment should be flexible to be adequate in the project to facilitate the test process, scrum iterations, incremental addition of test scripts, foster collaboration, while supporting real programing language allowing customizations [10].

C. *Automate Each Layer of Software When It Is Possible and Just for Acceptance Tests in the Sprint*

As was observed in Project 1, when the team spent too much time on first sprints trying to automate all functional GUI possible tests, they wasted time that should have been spent on tests that could have added value to the project.

It was noted in Project 2 that even after doing effective unit testing, good CI environment and GUI test automation, the business rules behind the interface couldn't have been ignored because they detected problems of integration and data accuracy important to the security of the system.

D. *Test Automation in the Agile Project Should Be Simple*

Test Automation cannot be a subject just for Testers. All team members should be able to use the automated test tools, run, write tests and see reports quickly. Concentrate the knowledge of tools in just one or two people, and adoption of complex automation tools, represent risks for agile projects and separate the team. The combination of skills to finish a task achieves good and quick results.

In Project 2 for example, the simplicity of Fitnesse tool allowed developers and testers to contribute together to the creation and execution of integration tests.

E. *Prioritize the Regression Test Execution*

Along the sprints of Project 2, the first automated test scripts executed many times didn't reveal defects because the functionality was stable, so they didn't need to be executed first. It is important run tests that have more probability to reveal faults speeding up the defect detection.

F. *Automate and Run Security Tests and Stress Tests Early Reduces Risks and Rework*

Note in Project 1, security and stress tests were executed at the end of the project, it caused extra tasks on the system and delay of project delivery chronogram. The defects could be detected before when intermediate versions were released avoiding risks and rework in the project.

Spending time to do this kind of tests at the beginning of the project aggregate value, knowledge and confidence for the agile team.

G. *Use Test Automation for Documentation and Information Feedback*

Test Tools are also highlighted in the technical literature as an important element for documenting software artifacts. In the context of these agile projects, some special tools should be cited:

- A test management tool controlled all information regarding the software testing activities, test cases specification, creation of new releases, test environment, test running, number of detected failures, and bug tracking (creation, correction, and validation).

In our software projects we used the *TestLink* and *Mantis* test tools. The integration of *Selenium* and Testlink allows the automatic execution results feedback facilitating test reports.

- The *Hudson* CI has an interface available to all team displaying release test status providing execution history and reports of all kinds of integrated tests. It gave to Project 2 the general vision and metrics on how the release status was and what problems were occurring in time to correct them. In Project 1, without CI, the quick feedback to team about the system status and test execution wasn't possible.

VI. CONCLUSION

This paper presented the experience in performing Test Automation in agile software development environments that used Scrum. We could observe that it is feasible to adapt agile practices to attend to the project goal, work to solve problems, search for integrated open source testing tools while maintain the management and organization of a software testing. The automation was a resource to document software, reduce cost, and allocate tasks in smaller parts. On the other hand, the difference in these scenarios regarding the testing activities could be avoided with some technological solutions already reported in the technical literature.

Some lessons learned could be extracted from these experiences and they could support other software engineers when performing Test Automation in a similar environment. In our next steps, we would be evaluating the effectiveness of the test automation practices in new software projects when compared to the scenario where testing are performed in other development platforms to extract the testing effort, number of detected failures pre and post-delivery, and schedule adherence.

ACKNOWLEDGMENT

The authors thank to Federal University of Amazonas and Nokia Institute of Technology for its contribution to this work.

REFERENCES

[1] E. Dustin, J. Rashka, and J. Paul, Automated software testing: introduction, management, and performance. Addison-Wesley, 2008.J.

[2] Karhu, K.; Repo, T.; Taipale, O.; Smolander, K.; , "Empirical Observations on Software Test Automation," Software Testing Verification and Validation, 2009. ICST '09. International Conference on , vol., no., pp.201-209, 1-4 April 2009 doi: 10.1109/ICST.2009.1

[3] E. Kit, Software Testing in the Real World: Improving the Process. Reading, MA: Addison-Wesley, 1995.J.

[4] Hu Zhi-gen; Yuan Quan; Zhang Xi; , "Research on Agile Project Management with Scrum Method," Services Science, Management and Engineering, 2009. SSME '09. IITA International Conference on , vol., no., pp.26-29, 11-12 July 2009 doi: 10.1109/SSME.2009.136

[5] Collins Eliane, Lucena Vicente. 2010. Iterative Software Testing Process for Scrum and Waterfall Projects with Open Source Testing Tools Experience. In Proceedings of the 22nd IFIP International Conference on Testing Software and Systems2010(ICTSS'10). CRIM, 2010. 115-120 p. [ISBN : 978-2-89522-136-4] .

[6] Crispin, L.; Gregory, J.; Agile Testing: A Practical Guide for Testers and Agile Teams, Addison-Wesley, 2009, ISBN 0-321-53446-8.

[7] Beck, Kent; et al. (2001). "Manifesto for Agile Software Development". Agile Alliance. Retrieved February 2012..

[8] Siebra, Clauirton A; Lino, Nancy L; Silva, Fabio Q B; Santos, Andre L M; , "On the specification of a test management solution for evaluation of handsets network operations," Automation Quality and Testing Robotics (AQTR), 2010 IEEE International Conference on , vol.2, no., pp.1-6, 28-30 May 2010. DOI= 10.1109/AQTR.2010.5520833.

[9] Antawan Holmes and Marc Kellogg. 2006. Automating Functional Tests Using Selenium. In *Proceedings of the conference on AGILE 2006* (AGILE '06). IEEE Computer Society, Washington, DC, USA, 270-275. DOI= http://dx.doi.org/10.1109/AGILE.2006.19.

[10] Hendrickson, Elisabeth. "Agile-Friendly Test Automation Tools/Frameworks," http://testobsessed.com/2008/04/29/agile-friendly-test-automationtoolsframeworks, 2008.