Journal of Theoretical and Applied Information Technology
15th October 2016. Vol.92. No.1
© 2005 - 2016 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

AGILE TESTING PRACTICES IN SOFTWARE QUALITY: STATE OF THE ART REVIEW

[1]CESAR GIL, [2]JORGE DIAZ, [3]MARIO OROZCO, [4]ALEXIS DE LA HOZ· [5]EDUARDO DE LA HOZ, [6]ROBERTO MORALES

[1]Assoc. Prof, Universidad de la Costa
[2, 3,4,5,6] Assoc. Prof. Universidad de la Costa, Department of Systems Engineering

E-mail: [1]cesargil2012@gmail.com , [2]jdiaz5@cuc.edu.co, [3]morozco5@cuc.edu.co, [4]adelahoz6@cuc.edu.co, [5]edelahoz6@cuc.edu.co, [6]romrales1@cuc.edu.co

ABSTRACT

In this paper you can find a review of articles related to agile testing practices in software quality, looking for theoretical information and real cases applied to testing inside a modern context, comparing them with the standard procedures taking into account their advantages and relevant features. As final result, we determine that agile practices in software quality have wide acceptance and many companies have chosen their use for all their benefits and impact on development software processes in several real applications, not necessarily IT governance ones, since other kind of technical applications have shown excellent results on testing.

Keywords: *Agile testing software, Scrum agile testing software, Kanban agile testing software, Test Driven Development agile test software, Behavior Driven Development test software, automation test software*

1. INTRODUCTION

Organizations apply software development methodologies through their growing process, to design computational tools with the best requirements according with the needs of each work unit and its integration as a system, resulting with a product that its quality will depend on many factors in a variable time and cost that can overcome the budget assigned to it.

In the last years, several agile testing practices have appeared that look for the best possible software quality, applying an innovative approach based on decision making of the software projects, through a set of iterative and incremental development activities, where requirements and solutions have an important role and evolve with time according to the needs of the project itself, before their completion.

This way, the work with agile methods is implemented with the collaboration of self-organized teams involved in a decision making shared process, where all members look to build an agile multidisciplinary team, and the tester becomes a multifunction expert, guaranteeing that the business value desired by the client is delivered with a sustainable and continuous rhythm.

In this case, agile methodologies don't consider software testing activities as a separate stage of the process, instead for them, it's integral part of it as coding, which makes a huge difference from conventional development and testing software techniques, all this permits a set of various practices related con agile testing, each of them with their own features, knowledge that motivates to identify the existing scientific literature in specialized databases about this subject in the last five years. This fact, implies the need of a review of the state of the art, according with the agile testing practices context, obtaining a better approach to the relevance of implementing these methods and techniques.

Based on the analysis of the articles found in specialized databases, we describe the results of 20 articles from notorious authors in their study area and finally present a set of conclusions from our review and software testing in general.

2. CONTEXT

Within lifecycle of software development, testing become a relevant mechanism of validation and verification of the software behavior according to the requirements settled by design, which permits to guarantee a level of quality in the developed tool. Agile Testing (AT) currently has the best impact and innovation, standing out among them:

- Test Driven Development (TDD) [1]: This practice is based on testing oriented development and produces simpler, more cohesive and less coupled code that the one created by traditional ways. Its use is growing in many development contexts and its associated to extreme programming.

- Acceptance Test Driven Development (ATDD) [2]: It´s an approach of requirements discovery based on team collaboration activities. In this case, the tests are created by the client, the developer and the tester. This strategy is called triad and it´s executed before the requirements application.

- Story Test Driven Development (STDD) [3], also known Behavior Driven Development (BDD): it´s a practice that communicates all the requirements to all stakeholders through tests. It´s also known as client test.

- Exploration tests [4]: It´s a software test where testers can interact with the system in any way and use the information provided with the objective of exploring all features of the system without restrictions.

- Automation software tests: It means to automate software test activities including development and execution of the tests scripts, test requirements verification and test tools use. One of the main reasons to automate tests is to diminish the time used doing it manually, also increase efficiency when repeating the system functionality, specially regression tests, where tests cases are executed iterative and incrementally after changes made on the software.

3. METHODOLOGY

Based on the context described previously, our first approach was to know studies, research projects and themes developed by authors of scientific community supported by international databases worldwide accepted, these became the need of information and search of scientific articles available in relevant sources such as IEEE, ScienceDirect, ACM and Taylor & Francis, also using additional tools like online translators, specialized texts and other bibliographic references, that guarantee a good information retrieval process.

After identifying the search needs and the information sources, we proceeded with the recovery process through queries in the databases already mentioned, through language controlled by the thesaurus. The keywords that allowed the search and recovery of the information in a faster, more precise, and more secure way, and with a high level of pertinence were:

- Agile Testing,
- Agile Methodology,
- distributed agile,
- Agile testing software,
- Scrum agile testing software,
- Kanban agile testing software,
- Test Driven Development agile test software,
- Behavior Driven Development test software,
- Automation test software
- control engineering computing,
- object-oriented programming,
- production engineering computing,
- program testing,
- quality management,

After searching with the keywords: *Agile testing software, Scrum agile testing software, Kanban agile testing software, Test Driven Development agile test software, Behavior Driven Development test software, automation test software*, the tables 1, 2 and 3 show the number of scientific articles retrieved. This search was made with a restriction of publications in the last 5 years only.

Table 1: Number of Articles retrieved with keywords: agile testing, Software and scrum agile testing software

Database	Agile testing software	Scrum agile testing software
IEEE Explore	798	119
ScienceDirect	4529	67
ACM Digital Library	148535	8936
Taylor & Francis	2431	97

Table 2: Number of Articles retrieved with keywords: Kanban agile testing software and test driven development agile test software

Database	*Kanban agile testing software*	*Test driven development agile test software*
IEEE Explore	9	173
SciencedDirect	140	2735
ACM Digital Library	148557	209937
Taylor & Francis	51	1578

Table 3: Number of Articles retrieved with keywords: Behavior driven development test software and automation test software

Database	*Behavior driven development test software*	*Automation test software*
IEEE Explore	133	88
ScienceDirect	397	66
ACM Digital Library	211644	431945
Taylor & Francis	36225	871

The higher amount of articles reviewed and analyzed come from IEEE and ScienceDirect in the first part of the retrieval. After filtering the results, 30% of them are discarded since they're not related to the objectives of the review and another 20% because they're book chapters or case reports that are not included in our analysis.

Finally, we obtained 20 papers that are strong related to our research theme, and they present concepts, theories and applications of agile methodologies clearly and understandable for its application in real contexts of software testing lead by authors recognized worldwide. For future works, we'll analyze another set of articles, giving priority to enterprise study cases.

Nevertheless, we found several state of the art reviews with the same theme, those are resumed next.

These selected references were chosen based on their strong impact in agile testing, taking into account their citations, their year of publishing and the impact factor of the publisher. We found an exponential growth in year 1998, which it´s a fact that has generated a bigger growth from 2002 to 2016 [21].

We performed a documental review about software agile techniques, which are quite different compared with traditional ones, both have become quite relevant in software development lately. Our conclusion is that they require little planning, since they share the same context than client requirements, looking to prevent delays in the product delivery, which this is why code is built to pass testing according to the needs of the client. [22] [23]

Another important fact to remember is that software development processes in 1990's were heavily criticized as slow, burocratic and had poor adaptation to changes, which generated a shift towards agile development in organizations [24]

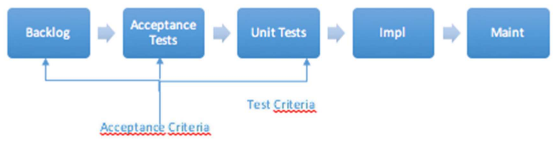

Fig. 1. Agile Testing Approach

Figure 1 represents agile methodologies procedure, specifically as a tool for acceptance and unit testing. They use TDD and ATDD as the most adequate for each unit test. [25]

We can also depict that agile methodologies are currently in a continuous predictive adaptive delevopment, one of the keys for this methodology is that uses a wave displacement approach, which makes them flexible enough in each of their processes, bringing less documentation and complexity management. Nevertheless, testing have a crucial role in lifecycle development software, since if they aren't present, product can't be evaluated for defects preventiong and minimizing risk failure in production. [25]

Assuming tests as an important process in software quality, it's relevant to highlight the role of these present methodologies, which are born from the needs and nature of the project, starting from zero and based on the requirement of the client. In this process, we can find usually operational and management problems such as lack of definition of the software scope, objective alignment inadequate estimation, communication issues between developers, testers and clients, lacking comprehension, etc. [26]

A high level of knowledge is needed to

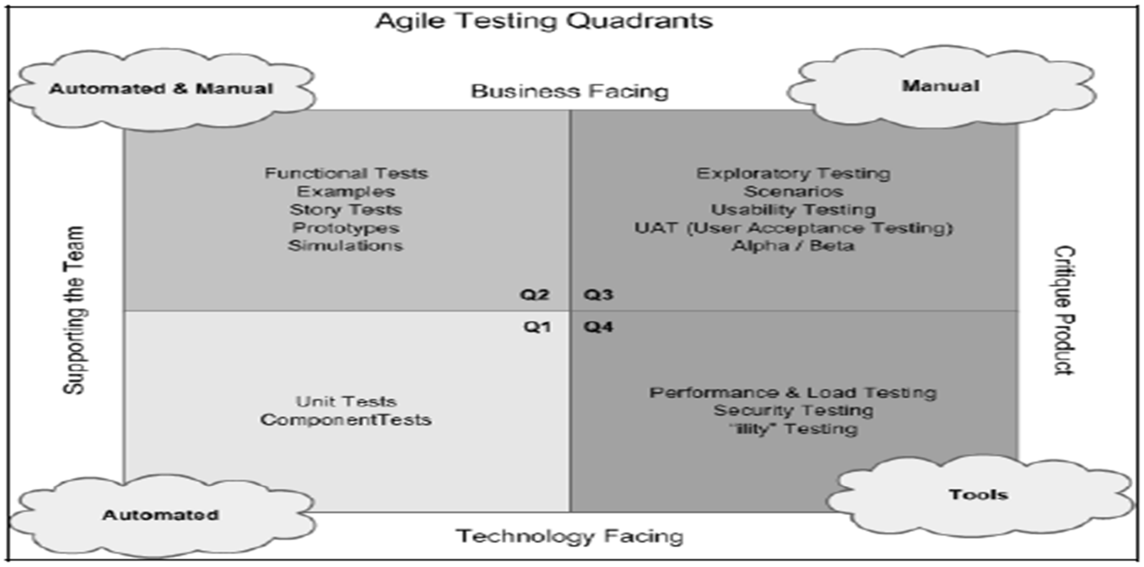

Fig. 2. Agile Methodology Matrix [5]

understand architectural challenges that participate in the adoption of agile approaches and industrial practices, to be able to develop huge and architectural challenging systems. We always will get to the question: Are we creating the right product? This is the moment where validation takes palce, and requires all stockholder participation for requirement specification and development. Validation is the process of making real tests in source code. Through validation, we guarantee that our product is designed covering the needs of the client, monitoring them through a check list using inspection meetings, comments, documents, to obtain a product that is compliant with the initial objectives set by stackholders. [27]

4. RESULTS OF ARTICLES REVIEW

Based on the set of articles found and filtered, we present a list of conclusions:

Robert Korošec [5] describes the procedure of making transition from AVL software development to the application of agile testing, using the construction of a matrix of four quadrants that store unit testing of the components, functional tests, system acceptance tests and quality system tests. Figure 1 shows this concept.

Automation test was implemented, and the tasks were divided among all the team members. A distribution tool was used for time and resources optimization during the execution of the test, obtaining positive results without the need of building a huge amount of requirements as the conventional method dictates.

The project described in [6] is the development of a spectrometer using proprietary technology, it receives a data spectrum and applies a mathematical complex algorithm, sending results to an analysis software through an independent communication link to a display unit. According to this, agile software testing applied to embedded software show several problems, especially the ones designed to test each unit or component because they're related to the communication between hardware and software of the system.

The proposed and applied solution to the process, was based on the construction of tests when hardware changes are made, so algorithms were built in Matlab to be executed in parallel along the test for each module, testing its output and input in the next module. This way, software was tested based on its response of each hardware module obtaining a highly reliable response with the methodology implemented.

The research developed by [7] proposes a project M directed to a commercial system for prepaid cell phones, and the project N with the objective of providing flexible billing for prepaid and postpaid of mobile users, based on business rules for protocols and URL addresses with differences in the billing for data connections.

Several aspects related with applying agile testing are exposed, especially some issues about programmers avoiding their responsibility with software quality, and relying only on testers for it, which it´s a problem since testers have no tools to make changes on the code or either aspects of the software, and also other problems related with team integration, even if there was a structured, organized and rigorous planning of the testing strategy compliant with the SCRUM methodology.

They recommend to set and apply integration strategies for the team that assigns clear and well differenced functions to each member, allowing permanent interaction between them and avoiding overlapping actions and skewing information among others considerations.

Di Bernardo [8], references the activities by software developers that give priority to the normal behavior of applications and won't emphasize to find the exception points in the events flow. Applying agile tests, particularly refactoring, it was possible to fin four errors, two of them unknown for testers in a production system, checking the exceptional behavior of it, reviewing that the right path for exceptions from their starting points to their destinations were correct. This test allowed to complement the proposed approach as an extension of the Junit framework.

Agile methodologies are used to handle the challenges of managing complex projects during the development phase. According to Hale [9] research, statistics results from a survey made to experts in Scrum and Kanban methods to compare their efficiency (means are not significantly different at 95% confidence level), showed little difference, at least to conclude that Scrum would be better than Kanba from the analysis of its effects in management factors of software development. Likewise, it suggest that Scrum and Kanban lead to successful projects and Kanban can be better than Scrum in terms of project management, and both can be used to handle budget management, risk control, project quality, amount of available resources, project scope and schedule control.

The research proposed in [10], sets the original area of product management as the discipline and integral actions that regulates a product, solution or service from start to delivery to the market or client, generating the highest possible value for business. This means that the application of the Scrum methodology to the product management software of a refinery, indicating the most relevant aspects where stand out identification and structuring what will be made based on the vision, theme, concepts and definition of requirements, which are the source of the flow diagram in implemented Scrum.

As a remarkable fact, the sprint was executed with a fixed length of 2 to 6 weeks (varying by company) simultaneously with the development sprint, and this was beneficial to start the next sprint of the following activity. They obtained a set of lists of actions based on improvements of the sprints, alternated by cycles, and the most amount of requirements was processed to their minimal expression, daily meetings were held about Scrum, and a discipline for management of reserve information was implemented, promoting early reuse of information and integration with all processes. With this, they obtained a huge improvement in software quality in controlled times according to each established strategy.

The study proposed by [11] analyzed the conclusions of articles previously published about the effects of TDD on quality software, considering inner and outer factors and the productivity of the company that builds the software, and comparing TDD with Development Testing (TLD). The applied method consisted in a systematic review of literature having articles between 1999 and 2014. The obtained results show that 57% of studies analyzed were validated through experiments approximately, and 32% of them were validated through study cases.

Likewise, analysis concluded that 76% of studies have shown a significant increase in inner software quality, while 88% had an increase of quality in external factors of software quality. Also there was an increment in the number of articles in academic scope, meanwhile in industrial context there has been a huge decrease in it. In general, 44% of the studies show a lower productivity using TDD comparing with the higher level domain, so they can conclude that TDD brings more benefits than TLD with inner and outer software quality, which affects all TLD developers with a lower productivity.

Some of the outstanding results from the research developed by [12], come from the automation of tests applying scrum methodology in three different software: They had a set of problems related with the methodology scrum, highlighting the old habit of doing regression tests manually. They propose three different strategies: unit testing automation for testers and system testing automation for the development of an API embedded in a modem, using a Google Test tool, according to level V application level referenced in [13]. As final result, they obtained a faster application of testing detecting code errors, even if no faults were found.

The second strategy consisted in the automation of a web application for production control in a factory. In this case, a continuous integration test was applied in sprint 1 and 2 using Hudson open tool. The developers executed TDD tests. A relevant result was fast feedback received by each member of the team of developers and testers. Nevertheless, there wasn't an absolute synchronization in automation, since each work team used different automation methods.

The third strategy was the participation of developers and testers in automation of unit and systems tests through 5 sprints. This strategy promotes team harmony, collaboration, knowledge transfer and fast feedback from sprint results.

According to [14], the idea of using agile testing is to promote radical changes in software development inside organizations, and that's clearly evident in software tests. Agile development redefines completely the traditional way of working. The use of these methodologies removes the backbone of the software development cycle in many organizations, including "quality control". This type of agile practices requires full integration of tests and development.

This proposal obtained results from data analysis at great scale with development projects for an Israeli aviation company (IAF), and they provide new evidence that agile testing really works and essentially they improve development quality and productivity. The information system of IAF is critical for daily operations and information security. As a result, the objective system is highly complex and must have the highest quality.

Working with professional testers in agile projects implies the whole team must test their own creation. Professional testers add value, not making more tests, but rewriting some of the ones made by the developers and adding new features. They also can add better test scenarios, even this can vary widely among them.

The use of professional testers proposes two key challenges for organizations adopting agile development: bottleneck tests and coordinate tests between testers and programmers.

Defect management in these kind of projects consists in two huge challenges: to control the workflow and to select and schedule the defects that will be solved. The team members check out their defects, counting and correcting them as a routine.

Workflow management is simpler in agile projects than traditional ones for three reasons:

- Any person can visualize a defect.
- Anyone can close a defect after being fixed and execute the right tests.
- Anybody that finds a defect can assign who will repair it.

In [15], they describe that agile methodologies are born with one goal in software development community, first we would have to talk about eXtreme Programming (XP) methodology, created by Kent Beck [16][|17]. Currently there are different agile methodologies, and we can enumerate some of them:

- XP (Extreme Programming): It proposes a light technique of software development, based on the programmer's discipline.
- Scrum: Its main focus is project management practices on engineering areas. It proposes continuous adaptation of the project planning, using divisions or iterations called sprints, where each of them produces a new version of the produce with new features.
- Crystal Methods: It proposes different processes to apply depending on three basic variables: project size, criticality, and priorities of it. Team members set the process to follow the entire project. It emphasizes team communication.
- DSDM Dynamic Systems Development Method: It focuses on RAD (Rapid Application Development) projects, with single phase of feasibility and then iterative phases for analysis, design and development.
- FDD Feature Driven Development: It proposes to set a series of features that must be contained in the product, hierarchically organized, with a scope short enough to be implemented in a couple of weeks.
- ASD Adaptive Software Development: It focuses on projects with unstable requirements with needs of fast development. It proposes the phases speculate – collaborate – learn to develop projects with these features.
- Xbreed: Combines management project practices from Scrum and XP. It's quite recent and there aren't many references about it.

In [18], the development of big intensive systems in software is a complex task that must be carried out applying a divide and conquer strategy. Companies face with the challenge of coordinating the individual aspects of software development in particular, focusing in two principles: Requirements Engineering (RE) and Software Testing (ST), both implied in agile testing software methodologies. A wrong alignment, not only can lead to a wasted effort, but to faulty software. Nevertheless, before an organization can improve these aspects, it's necessary to know coordination mechanisms. When using REST-bench, its goal is to provide an evaluation tool that illustrates coordination in software development projects and identify concrete improvement opportunities. This tool is developed based on solid foundations of a taxonomy of REST alignment methods, validated in five study cases. The opportunities that were identified indicates that the evaluation was effective and efficient. On the other hand, participants confirmed that their comprehension about coordination between RE and ST improved significantly.

Research developed by [19], wanted to understand how software developers experience continuous performance adaptation in a highly volatile and modern environment, using the software methodology of Lean and Agile. This knowledge can be used as foundation to build and maintain high performance teams, to communicate performance improvement initiatives and also get better work conditions for developers.

As final result, 33 main categories of performance factors and relationships between them were found. A comparison with study cases revealed similarities and differences between types and sizes of the organizations.

In this study, software teams are committed to a permanent cycle of the interpretation of their actions and negotiation of the alignment with other interested parts of the organization. Certainly, there can be difference in size among them, a set of common performance experiences is present despite different context variables.

Improve the performance experiences require integration of soft factors, as communication, team spirit, team identity, and values in development process, which suggest a software development vision and the performance of an innovative software team is centered in social sciences and behavior.

Finally, work proposed in [20], has the objective of analyzing the combination of architecture and agile methods with the goal of exploring and analysis focused in activities and approaches, agile methods and practices, costs, benefits, factors, tools and learned lessons comparing with the combination of the two first ones. As final result, 54 studies were included in this research. Some of the most outstanding aspects found were: (1) There is a meaningful difference in the proportion between activities, the agile methods architecture and practices used in the combination. (2) None of the architecture approaches has been widely used in combination. (3) There is a lack of description and analysis related to costs and the failure stories of the combination. (4) 20 challenges, 29 factors, and 25 learned lessons were identified.

The results of this study help to the community of software engineering to think over the last 13 years of research and the practice of combining architecture and its implications with software agile methodologies.

5. CONCLUSIONS

 Based on the analysis of review articles, we can conclude that search must be refined and filtered with the most number of keywords according to the topics specialized in software agile testing, otherwise the number of articles would be impossible to reference through a research of this kind.
 The lower amount of articles found are related to tools used in automation software testing, which brings an open window of opportunities to develop new research in this topic.
 We advise having in mind other designations for the search topics, in case of not finding related articles directly with the search objective, this is why the context definition is fundamental to refine search scope.

Based on the articles reviewed we can conclude:

- Testing must be executed simultaneously with software development, in other words, there must be a team of experts testing continuously and not when development is finished.
- Agile testing uses continuous feedback, which allows to redirect all the process during software development.
- In a team, all members perform direct tests, or any other tests, including users through alfa o beta versions of the software being developed.
- Time for obtaining mistakes and deciding improvements and corrections is shorter every time and can be done anytime, since this is determined by each iteration in all teams, including the business area. Likewise, the cost of corrections is lower than those made at the end of the process of quality assurance.
- Each error or errors are corrected in each iteration after constant testing, obtaining clean code permanently.
- Testers use checklists to collect essential

information of the test and any details that aren't relevant are discarded.

- Organizations should automate testing for each software layer.
- Test should be easy, reusable and understandable for each team member.

REFERENCES:

[1] D. Janzen and H. Saiedian, "Does Test-Driven Development Really Improve Software Design Quality?" in *IEEE Software*, vol. 25, no. 2, pp. 77-84, March-April 2008. doi: 10.1109/MS.2008.34

[2] L. F. S. Hoffmann, L. E. G. d. Vasconcelos, E. Lamas, A. M. d. Cunha and L. A. V. Dias, "Applying Acceptance Test Driven Development to a Problem Based Learning Academic Real-Time System, *"Information Technology: New Generations (ITNG), 2014 11th International Conference on*, Las Vegas, NV, 2014, pp. 3-8. doi: 10.1109/ITNG.2014.63

[3] S. Park and F. Maurer, "A Network Analysis of Stakeholders in Tool Visioning Process for Story Test Driven Development," *Engineering of Complex Computer Systems (ICECCS), 2010 15th IEEE International Conference on*, Oxford, 2010, pp. 205-214.doi: 10.1109/ICECCS.2010.5

[4] B. Suranto, "Exploratory software testing in agile project," Computer, Communications, and Control Technology (I4CT), 2015 International Conference on, Kuching, 2015, pp. 280-283. doi: 10.1109/I4CT.2015.7219581

[5] R. Korosec and R. Pfarrhofer, "Supporting the Transition to an Agile Test Matrix," *Software Testing, Verification and Validation (ICST), 2015 IEEE 8th International Conference on*, Graz, 2015, pp. 1-2.

[6] N. Van Schooenderwoert and R. Morsicato, "Taming the embedded tiger - agile test techniques for embedded software," *Agile Development Conference, 2004*, 2004, pp. 120-126. doi: 10.1109/ADEVC.2004.21

[7] A. M. dos Santos, B. F. Karlsson, A. M. Cavalcante, I. B. Correia and E. Silva, "Testing in an agile product development environment: An industry experience report," *Test Workshop (LATW), 2011 12th Latin American*, Porto de Galinhas, 2011, pp. 1-6. doi: 10.1109/LATW.2011.5985897

[8] R. Di Bernardo, F. Castor and S. Soares, "Towards Agile Testing of Exceptional Behavior," *Dependable Computing Workshops (LADCW), 2011 Fifth Latin-American Symposium on*, Sao Jose does Campos, 2011, pp. 21-24. doi: 10.1109/LADCW.2011.12

[9] H. Lei, F. Ganjeizadeh, P. K. Jayachandran, and P. Ozcan, "A statistical analysis of the effects of Scrum and Kanban on software development projects," *Robot. Comput. Integr. Manuf.*, pp. 1–9, 2015.

[10] K. Vlaanderen, S. Jansen, S. Brinkkemper, and E. Jaspers, "The agile requirements refinery: Applying SCRUM principles to software product management," *Inf. Softw. Technol.*, vol. 53, no. 1, pp. 58–70, 2011.

[11] W. Bissi, A. G. Serra Seca Neto, and M. C. F. P. Emer, "The effects of test driven development on internal quality, external quality and productivity: A systematic review," *Inf. Softw. Technol.*, vol. 74, pp. 45–54, 2016.

[12] E. Collins, A. Dias-Neto and V. F. d. Lucena Jr., "Strategies for Agile Software Testing Automation: An Industrial Experience," *Computer Software and Applications Conference Workshops (COMPSACW), 2012 IEEE 36th Annual*, Izmir, 2012, pp. 440-445. doi: 10.1109/COMPSACW.2012.84

[13] G. Myers (2004): The Art of Software Testing. Ed. John Wiley & Sons, Inc., Hoboken, New Jersey.

[14] D. Talby, A. Keren, O. Hazzan and Y. Dubinsky, "Agile software testing in a large-scale project," in *IEEE Software*, vol. 23, no. 4, pp. 30-37, July-Aug. 2006. doi: 10.1109/MS.2006.93

[15] K. Beck and M. Fowler, *Planning extreme programming*. Boston: Addison-Wesley, 2001.

[16] Kent Beck. "Extreme Programming Explained: Embrace Change". Reading, Addison Wesley, 1999.

[17] Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, and others. "Agile Manifesto". 2001.http://agilemanifesto.org/

[18] M. Unterkalmsteiner, T. Gorschek, R. Feldt, and E. Klotins, "The Journal of Systems and Software Assessing requirements engineering and software test alignment — Five case studies," *J. Syst. Softw.*, vol. 109, pp. 62–77, 2015.

[19] F. Fagerholm, M. Ikonen, P. Kettunen, J. Münch, V. Roto, and P. Abrahamsson, "Performance Alignment Work: How software developers experience the continuous adaptation of team performance in Lean and Agile environments," *Inf. Softw. Technol.*, vol. 64, pp. 132–147, 2015.
[20] C. Yang, P. Liang, and P. Avgeriou, "The Journal of Systems and Software A systematic mapping study on the combination of software architecture and agile development," *J. Syst. Softw.*, vol. 111, pp. 157–184, 2016.
[21] C. ©. 2. E. B.V, «Scopus» Elsevier, 2016. [Online]. Available: https://www.scopus.com.
[22] K. Sunil y S. Priya, Applying FLOOT Testing to Agile Methodology, Reliability, Infocom Technologies and Optimization (ICRITO) (Trends and Future Directions), 2015.
[23] R. O'Connor. N. Baddoo, J. J. Cuadrado-Gallego, R. Rejas Muslera, K. Smolander y R. Messnarz, Software Process Improvement: 16th European Conference, EuroSPI, Alcala (Madrid), Spain: Springer Science & Business Media, 2009.
[24] « Brief History of Agile Version One, Agile Sherpa by and for the, » 2016. [Online]. Available: http://www.agilesherpa.org/.
[25] S. AMBLER y M. LINES, «Agile Practices Survey Results: July 2009, » 2009. [Online]. Available: http://www.ambysoft.com/surveys/practices2009.html.
[26] «Overcoming Testing Challenges in Project Life Cycle using Risk Based Validation Approach, » K. Nageswara Rao et al. / International Journal on Computer Science and Engineering (IJCSE) , vol. 3, nº 3, pp. 1232-1239, 2011.
[27] D. F. Rico, «v&v lifecycle methodologies, » Software Engineering Terminology, 2014.
[28] C. Tulika, C. Samyadip y J. Nasina, «Analysis of Agile testing attributes for faster time to Market: Context of Manufacturing sector related IT projects, » Procedia Economics and Finance, vol. 11, pp. 536-552, 2014.