A Synchronous Agile Framework Proposal Combining Scrum and TDD

Marcia Maria Savoine [1], Vanessa França Rocha [1], Carlos Andrew Costa Bezerra [1],
André Magno Costa de Araújo [2], Joyce Karoline Maciel Matias [1]
ITPAC – President Antonio Carlos Tocantinense Institute Araguaina, Brazil [1], Federal University of Pernambuco Recife, Brazil [2]
e-mail: savoine@gmail.com, vanfranrocha@gmail.com, andrew@r2asistemas.com.br, amcaraujo@gmail.com, joycekarolpaz@gmail.com

***Abstract* — Optimizing processes in software development are becoming increasingly more popular. For that reason, the consuming market demands more efficiency and quality. To achieve that, some methodologies are adopted in order to ensure that real value will be delivered to the customer. This paper relates a series of good practices based on team management features described in Scrum, and the development and source code testing covered in the TDD methodology. By specifying a framework structure with such features, this work allows software factories use a lean model, facing the reality of their projects in several aspects (e.g., team management, code development and testing).**

Keywords- Software Development; Methodologies; Scrum; TDD.

I. INTRODUCTION

The continuous improvement in productivity and organizational processes depend on the use of software as a competitive advantage [1].

Therefore, new methodologies and techniques are adopted to handle processes in software production quality as well as providing training to professionals, aiming to manage and develop products efficiently and in short time.

In harmony with such fast-paced world, Agile methodologies propose structured practices and organized steps throughout the software development cycle. More specifically, Scrum aims to manage all the processes that take place within each project event to obtain a detailed and complete overview of the features developed and their deadlines.

On the other hand, in Test-Driven Development (TDD), the process is apparently simple - tests are written before writing the code itself, not only in order to address shortcomings, but to meet the features in a reliable and predictable manner [2].

Considering the Agile methodologies, it is noticed that small and medium-sized software factories usually find it substantially difficult to fully employ Agile practices, as the size and complexity of the project, possible lack of control, poor staff training and difficult conciliation with existing processes are some of the points that hinder the adoption of these methodologies [3].

This work aims to determine the features offered by both agile methodologies (Scrum and TDD) and asserts when software factories should use each one as a model according to the reality of each project.

This paper presents a theoretical and practical framework for systematic association of Scrum and TDD combining the strengths of both (i.e., responsive management and agile development, respectively) through an exploratory research carried out in the documentation and guidelines of the investigated methodology. It is also dedicated to compare both methodologies to raise similarities and differences towards proposing a method using an association of both to a conceptual framework throughout the software development cycle.

This paper is structured as follows: Section 2 presents the related work; Section 3 presents Agile methodologies concepts, for Scrum and TDD respectively. Section 4 presents a comparative analysis of the studied methodologies. Section 5 shows a framework with the proposed combination and use of both methodologies. Last but not least, Section 6 presents conclusions and future work.

II. RELATED WORK

This research focused mainly on finding works in which the main concern was assessing the use of the Scrum framework to manage activities and the team, and the use of code management in TDD. The main point those works had in common was indicating that the use of TDD with Scrum provided benefits to the software development and design. Sinialto and Abramhamsson [4] show that there is an improvement in the code and application test coverage. However, Janzen and Saiedian [5] show that TDD makes developers more confident during code maintenance, hence leading to higher productivity and possibility of delivery on time.

Puleio [6] points out that for a new project that was meant to replace an existing legacy service with a new one after long meetings, the team opted to use Scrum for project management, TDD and pair programming. Soon after, they decided to fully embrace XP (Extreme Programming) practices. After several difficulties encountered and solved, the project was successful, leading the team to conclude that the code tests could be done following agile methods.

Despite the growing popularity of Agile methodologies, there is a limited amount of literature that combines Agile methodologies and software testing, especially concerning

 ISBN: 978-1-61208-498-5

how to perform the tests and integrate with Scrum. For this reason, Van den Broek et al. [7] performed an analysis based on a case study on the use of tests in a Scrum team. Afterwards, the authors proposed a visual model that integrates testing and some Scrum activities.

Our research relates to the work in Van den Broek et al. [7] since they propose a visual model that integrates testing activities and Scrum. However, we noticed that there are factors affecting the adoption of Scrum and TDD, as the latter is only used to test the practices in Scrum. Our work used practices, characteristics, Scrum roles and artifacts from both Scrum and TDD equally, in order to improve processes where one fails and another completes.

III. AGILE METHODOLOGIES

Agile methodologies aim to accelerate software development and delivery, as a means to make pieces of new software more frequently available to clients and improving the participation of all stakeholders. Therefore, Agile methodologies have specific characteristics, but maintain the same core principles throughout the life cycle of software development [8]. These principles are customer engagement, incremental delivery, people over process, accepting changes, and keeping it simple.

Having the core principles in mind, in subsections A and B the specificities of the Scrum framework and TDD methodology addressed in this work are described in detail.

A. Scrum

Scrum is used for managing Agile software projects iteratively and incrementally. In this sense, Sutherland [9], states that the Scrum framework aims to capture the way the teams really work, giving them the tools to organize themselves and, most importantly, quickly improve the speed and quality of their work.

In Fig.1, one may notice that, initially the Product Owner and Stakeholders set the Product Backlog, which is followed by prioritization of the sprints (Sprint Backlog). At the end of the Sprint a product increment is delivered to the customer.

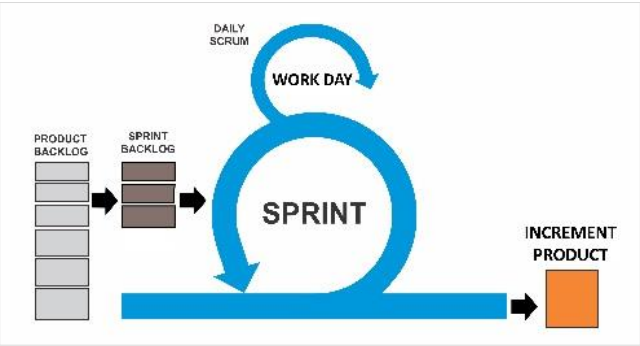

Figure 1. Scrum Cycle. [10]

Scrum consists of three roles that are responsible for executing events and building artifacts. They are:

- Product Owner: responsible for prioritizing the Product Backlog (list of requirements) and getting important information from the stakeholders.
- Scrum Master: ensures that all the work occurs smoothly and in an organized fashion without interruptions, acts as a facilitator or conductor in Scrum meetings and assists the interaction between the Product Owner and the development team.
- Team Members: people who carry out development and testing. The team must be organized and have deep product knowledge.

Scrum implements an iterative and incremental skeleton through fully decentralized roles and responsibilities [11]. In Scrum, Events and Artifacts are:

- Product Backlog: List of requirements, design features, extracted from user history.
- Release Backlog: The product is set to be developed in parts (Sprints). In the end of each Sprint, the delivery of completed increments is called Release Backlog.
- Sprint Backlog: each Sprint aims to add an important part of the Release Backlog.
- Daily Scrum: At the end of each day of development all members in the team come together for the Daily Scrum, a short meeting lasting around 15 minutes.

Despite Scrum's goal being to deliver value to the customer in the form of relevant features in the final product, members of a Scrum project team should use some artifacts to support the decentralized and simple management [12].

The great advantage of Scrum is achieving the delivery of a functional product with higher quality and lower cost, with a team that works in less time. Unlike other methods, it consists in a pre-determined time-box to verify and validate whether what is being done is what was actually established and whether it is adaptable to the format and type of project.

Scrum proposes a new software management framework, which is based on self-organization, motivation, ownership and pride of a team in carrying out their acquisitions. As it is adaptable, Scrum provides the right support for the team, and accommodates phases that are important to the quality of the software production, such as testing activities that are a trend among renowned software factories [13].

B. TDD

TDD is a methodology focused on software quality, in particular the quality of implementation and testing. TDD increases the reliability of the system and raises the assurance that what was executed is in accord with the proposed requirements [14].

According to Aniche [15], TDD is a software development methodology that is based on the repetition of a small cycle of activities. First, the developer writes a test that fails. Then, the developer fixes what is missing and makes the test passes, implementing the desired functionality. Finally, they perform the refactoring of the code to remove any duplication of data or code generated in the process, as shown in Fig. 2.

These three rules of TDD bring immediate benefits when applied: class design with very low linkage; software documentation on tests; flexibility; and debugging time reduction.

Conditions that should be tested are loops, operations and polymorphisms; however, tests should be applied only to those conditions written by the developer himself. This allows

 ISBN: 978-1-61208-498-5

the testing to be compatible with the logic used in the development, covering all the code characteristics [16].

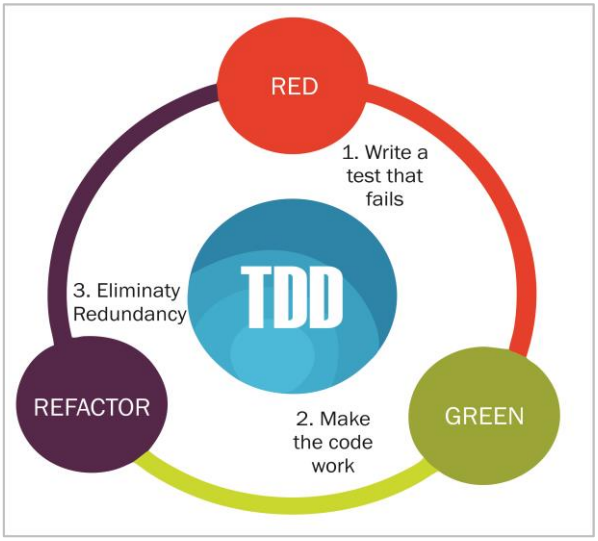

Figure 2. TDD Cycle. [16]

The main reasons for the adoption of TDD according to Kaufmann and Janzen [17] are presented below: the development is set by first considering the objectives, then thinking about the possible solutions; the understanding of the system can be achieved by reading the tests; unnecessary code is not developed; there is no piece of code without test; once a test is working, it is known that it will always work and it also serves as a regression test; tests allow progress to be made in the software, because they ensure that the changes do not alter the operation of the system.

The feedback provided by TDD promotes significant improvement in the design of classes, helping the programmer to encode more cohesive and less coupled classes, reducing the occurrence of errors and software maintenance costs.

IV. COMPARATIVE ANALYSIS BETWEEN THE AGILE METHODOLOGIES SCRUM AND TDD

We held a comparative analysis described aiming to highlight the similarities and weaknesses in both methodologies, and trying to identify where the inconsistencies of a method could be complemented by practices of the other. This was done after realizing that the use of Scrum focuses on management of software projects which had great feedback from self-manageable and self-organized teams, with a strong contribution from the Scrum Master. In this way, Scrum and TDD have different applications, as the latter focuses on software quality by building code based on programming and testing standards.

A. Specific Features

The presented features were selected based on the theoretical framework of Section II, paying attention to their relevance and degree of understanding, and also considered specificities to indicate essential activities found in software projects and executed by a software factory. These characteristics must objectively state individual and similar points of methodologies that make direct reference to the core Agile principles and the structure of the methodologies in the study. Thus, we selected the following:

- Self-organizing team: team organization and work among members who can, for the most part, find the best solution to manage and carry out their work.
- Project supervisor: responsible for monitoring the team giving the necessary support, supervising the work done and what remains to be done;
- Results of the development report: measures the evolution of product development and is presented periodically;
- Setting steps: Breaking the project into phases or stages;
- Quality code: code analysis based on the architecture and coding standards;
- Small steps: Process in which the project has its phases broken into smaller parts;
- Time reduction: Gain of time based on an architecture without design;
- Cost reduction: Savings generated by the use of the methodology;
- Simplicity: Doing only what is necessary;
- Feedback: Returning information to the team and customer;
- Project delivery: All increments are fully tested, guaranteeing they work together and providing a quality delivery.
- Adaptable to change: Product is flexible for adjustments and redesigns;
- Testing responsible: Team person who assumes the function of tester;
- Format test: How the tests are applied to software.

By observing the behavior of each methodology in each pinpointed feature, we analyzed the individual weaknesses and characteristics that are complementary or overlapping, as well specific practices, roles and artifacts (Table 1), which are indicators that help and guide teams through the development process. For example, practices 6 and 7 are complete when executed together; on the other hand, item 8 in "Roles" highlights that such a feature does not exist in the TDD methodology; however, it exists in Scrum, and the Scrum Master benefits if he decides to adopt both methodologies. Likewise, the item 10 in "Artifacts" shows there are no structures in TDD to address progress besides tests, and Scrum complements it.

The final analysis highlights the strongest features, structure and procedures of each method. Thus, although those are meant for different segments – in the software development context where there is a close attention scenario for good practice that values quality, scalability and effective results - the application of both methodologies in a synchronized and adjusted fashion, maintaining their individual characteristics, may contribute for better results.

 ISBN: 978-1-61208-498-5

TABLE I. ESSENTIAL CHARACTERISTICS OF THE AGILE METHODOLOGIES OF SCRUM AND TDD

			AGILE METHODOLOGY	
N°	**INDICATORS**		**SCRUM**	**TDD**
1	**Practices**	Self-Organizing Teams	The team self-organizes activities	Not specified
2		Setting Steps	*Sprints*	Test Cycle
3		Small Steps	Division in sprints for development of releases	Baby Steps guarantee features are broken in as many parts as needed
4		Simplicity	Produce only the necessary	Simplification of procedures and code
5		Feedback	The staff and customer	Continuous feedback to the team
6		Adaptable to Change	Changes in the project and its project type	Test suite ensuring changes without loss of performance and functionality
7		Format test	Parallel-running implementation	Written test before implementing functionality
8	**Roles**	Project Supervisor	Scrum Master	Not specified
9		Testing responsible	Developer	Developer
10	**Artifacts**	Evolution report on the results	Daily Scrum and Burn Down	Automatized tests
11		Project Delivery	Whenever it is all done	Not specified
12		Code Quality	Not specified	Compliance with Object Oriented Programming
13		Time Reduction	Deliver something functional for the customer in a short time	Development and maintenance since it optimizes the process and reduces errors
14		Cost Reduction	The less time, less charges will be applied	Software maintenance error correction or projection errors

V. PROPOSED SYNCHRONOUS USE OF SCRUM AND TDD

We developed a proposal based on the analysis performed with the purpose of exploring characteristics and attaining better comprehension of possible coexisting use of both methods in the form of a synchronous framework that exemplifies the interaction of Scrum and TDD as Agile methodologies and how their integration can be made so they may coexistence in harmony.

According to the framework structure shown in Fig. 3, the integration of Scrum and TDD is applicable to a self-organizing team (item 1) which values the simplicity in the processes and also in the code (item 2) and follows the lead of a Scrum Master. Setting out small steps for each *Sprint* (item 3), the organization achieves in reductions in time and cost (items 4 and 5), as the team produces only the

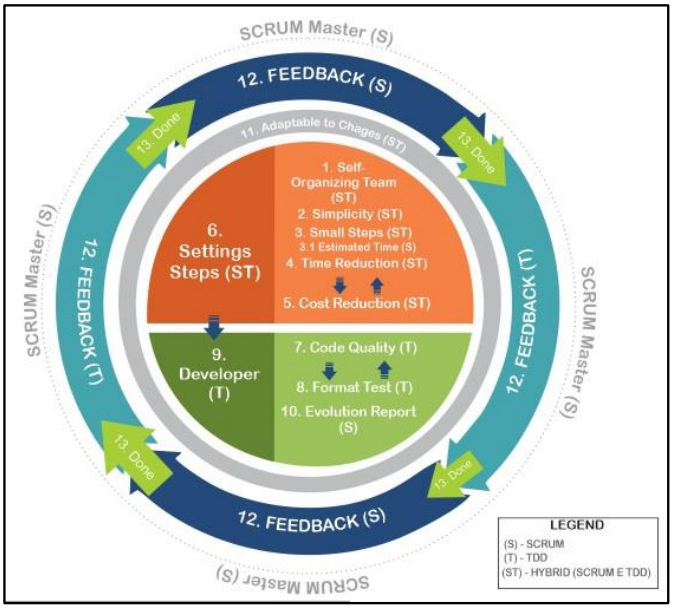

Figure 3. Proposed Framework for combineduse of Scrum and TDD

necessary and minimizes errors and do-overs. The developer (item 9) is responsible for coding and testing following the standards established by TDD (items 7 and 8), and presents its findings to the Scrum Master and the team at every Daily Scrum. The code documentation that comes down to testing as set out in TDD is available to be evaluated and improved by the team, therefore enabling the Scrum Master to measure (item 10), not only what is being done, but how it is being done, ensuring quality to the team and to the end customer. At this point of the framework, we stress the ability to adapt to change (item 11), with documented and tested code and a self-organizing team ending the cycle with continuous feedback with the customer and the team (items 12 and 13); this guarantees that, when the software is considered ready (item 14), it will have high quality levels, ensuring the adaptation and survival of the project even when facing changes to the original delivery.

The presence of the Scrum Master (Scrum methodology) is in evidence because they are present throughout the framework cycle, facilitating and enhancing the work of the entire team, ensuring that the framework is followed, and seeking for continuous improvement.

As stated by Silva and Lemos [18], the role of the Scrum Master is analogous to that of an orchestra conductor. Both should provide guidance and steady leadership to a team of talented professionals who work together to create something that no one can do alone.

Thus, through the proposed framework, we sought to address characteristics of the two methodologies which, by coexisting, may further contribute to the success of the project and the quality of software.

 ISBN: 978-1-61208-498-5

VI. CONCLUSION

This paper presents two aspects within the agile context: the management team which is involved throughout the software process with Scrum and the team that actually develops the lines of code and performs tests using TDD. No methodology by itself includes both strands effectively; therefore, to address this situation we studied both methodologies and proposed a framework which can be used in software factories.

Based on the analysis and the framework presented, it is clear that the differences and similarities found in both methodologies make them more useful when used in tandem, as they guarantee software is produced avoiding problems in the code due to continuous testing, which prevents new errors and corrections during future implementations, eventually reducing time and cost. Thus, the combined use of Scrum and TDD is strongly recommended, as it will bring clear gains to the project by indicating the increase in staff quality, product and codes, and then covering the whole process of development, evolution and maintenance of software grounded on best practices and ensuring full feedback on all processes and practices.

In the future, we intend to develop a tool to assist in software development by proposing a set of coexisting Scrum and TDD methodologies, confirming the efficacy of the proposed framework and performing its validation in a real scenario of a software project.

REFERENCES

[1] J. Herbsleb and D. Moitra, “Global Software Development”, EUA: IEEE Software. 2011.

[2] S. Freeman and N. Pryce, “Growing Object-Oriented Software, Guided by Tests”.Addison-Wesley Signature Series, 2010.

[3] C. Andrade, J. Lopes, W. Barbosa and M. Costa, “Identifying difficulties in the implementation and contract management in agile projects in Belo Horizonte”. DOI - 10.5752/P.2316-9451.2014v3n1p18. Abakós , v. 3, 2014, p. 18-37.

[4] M. Siniaalto, P. Abramhamsson, "A Comparative Case Study on the Impact of Test-Driven Development on Program Design and Test Coverage". Finland: First International Symposium on Empirical Software Engineering and Measurement (ESEM 2007), 2010.

[5] D. Janzen, H. Saiedian, "On the Influence of Test-Driven Development on Software Design". USA: 19th Conference on Software Engineering Education & Training (CSEET'06), 2011.

[6] M. Puleio, "How not to do Agile Testing". USA: AGILE 2006 (AGILE'06), 2006.

[7] R. van den Broek, M. M. Bonsangue, M. Chaudron and H. van Merode. "Integrating testing into Agile software development processes". Lisboa: Model-Driven Engineering and Software Development (MODELSWARD) 2nd International Conference on, 2014.

[8] I. Sommerville, “Software Engineering”, Boston: Pearson. 2007.

[9] J. Sutherland, “Scrum: The Art of Doing Twice the Work in Half the Time”, New York: Crown Business. 2014.

[10] K. Schwaber, “Agile Project Management witch Scrum”, Reedmond: Microsoft Press. 2013.

[11] K. Schwaber, “Scrum Guide”, Harvard Businner Review, Boston, IV, p. 163-179., 2013.

[12] Z. Požgaj, N. Vlahović, V. Bosilj-Vukšić, "Agile Management: A Teaching Model Based on SCRUM". MIPRO-IEEE, 26-30 May, Opatija, Croatia. 2014.

[13] A. Pham and P. Pham, “Scrum in Action: Agile Software Project Management and Development”, Course Technology, 2011.

[14] K. Beck, “Test-Driven Development By Example”. Estados Unidos: Addison Wesley, 2002.

[15] M. Aniche “Real World Test-Driven Development”, São Paulo: Code Crushing, 2013.

[16] K. Beck, “Test-Driven Development”, 1 st. ed. Addison-Wesley Professional, 2002.

[17] R. Kaufmann, D. Janzen., “Implications of Test-Driven Development A Pilot Study”. In: 18th Annual ACM Conference on Object-Oriented Programming System, Languages and Application (OOPSLA 2003). New York: ACM, 2003.

[18] W. Silva and L. Lemos “SCRUM: A New Approach to Software Development Front of Current Competitive Scenario Demand”, Rio de Janeiro: Fluminense Federal University. 2008.

 ISBN: 978-1-61208-498-5