Using Agile Methods for Software Development in R&D Scenario

USING AGILE METHODS FOR SOFTWARE DEVELOPMENT IN R&D SCENARIO

Luiz Eduardo Guarino de Vasconcelos[1,2], André Yoshimi Kusumoto[1,2], Nelson Paiva Oliveira Leite[1], Cristina Moniz Araújo Lopes[2,3]
[1]Instituto de Pesquisas e Ensaios em Voo (IPEV)
[2]Instituto Tecnológico de Aeronáutica (ITA)
[3]Instituto de Aeronáutica e Espaço (IAE)
Pça Marechal Eduardo Gomes nº50 - São José dos Campos, SP, BRAZIL
du.guarino@gmail.com; kuzmoto@gmail.com; epd@ipev.cta.br; cmoniz77@gmail.com

ABSTRACT

Due to the quick change of business processes in organizations, software needs to adapt quickly to meet new requirements by implementing new business rules. In Research and Development (R&D) scenario, the research is highly non-linear and changes are inevitable.

In this context, it is known that traditional methodologies (e.g. waterfall) may lead to the detection of failures late, increase the time and cost of development and maintenance of software. On the other hand, agile methodologies are based on Test-Driven Development (TDD), maintain the technical debt under control, maximize the Return on Investment and reduce the risks for customers and companies.

In this paper, we show the use of Scrum and TDD in the development of an experimental tool that aims to make the calibration in real time of the rudder of a fighter aircraft. The preliminary results allowed to increase the coverage testing of the software and hence the quality of the tool.

KEY WORDS

Flight Tests; Agile Methods; Test Driven Development; Research and Development

INTRODUCTION

In Research and Development (R&D) scenario, when it comes to the beginning of a research, often there is the difficulty of performing accurate estimate of the effort to be used in the development of research because the requirements are high level. Besides, according to [1], the research is highly non-linear. As research and product are developed, the requirements become more refined and changes are inevitable.

Besides, sometimes, our meetings became inefficiently and ineffectively. The time required for each Project meeting can vary significantly: a status report can take 30 minutes, while working solutions to a technical problem can take more than one hour.

Another problem in this context is when the team members only have status to report, the meeting slot could be too long, and even if we ended quickly it was hard to make use of the remaining meeting time for work. Furthermore, when the members had a technical issue to explore, there was not enough time, and a follow-up meeting might have to wait some days or even some weeks, slowing progress and increasing overhead. Another problem was to know what every one was really doing in the project. There wasn't a way to centralize the to do list of each member. Thus, a task could be done, at the same time by more than one member.

Many technologies have been created in the field of software development, to accelerate the production and maintenance of software products. In addition to the technologies used, the software development methodology also impacts on productivity and meeting deadlines while developing software. According to [2], a software development methodology is a set of activities that assist in the production of software. The result of these activities is a product that reflects the way the whole process was conducted.

It is known that traditional methodologies (e.g. waterfall) may lead to the detection of failures late, increase the time and cost of development and maintenance of software, once an application is in the testing stage, it is very difficult to go back and change something that was not well-thought out in the concept stage, no working software is produced until late during the life cycle, high amounts of risk and uncertainty, not suitable for the projects where requirements are at a moderate to high risk of changing [3, 4]. On the other hand, agile methodologies are based on Test-Driven Development (TDD), maintains the technical debt under control, maximize the Return on Investment (ROI) and reduce the risks for customers and companies [3]. According to [6], software applications developed through the agile methodologies have higher success rate and lower risk than traditional waterfall methodology.

This paper is organized as follows. Agile methods concepts and fundamentals are described briefly in section 1. The tailoring agile methods to IPEV are described in section 2. The experiment and results are considered in section 3 and the final considerations are presented in last section.

1. AGILE METHODS

The definition of an agile methodology was created in 2001 that defined the Agile Manifesto [16]. This manifesto is a simple and concise declaration that seeks to change the traditional lens that has been used to see software development. The agile manifesto works with some values, that are: i) individuals and interactions over tools and processes; ii) working software over detailed documentation; iii) client collaboration over contract negotiation; and iv) change adaptation over plan following. Furthermore, the agile manifesto has 12 principles [16].

Nowadays, there are dozens of agile software development methodologies (e.g. Scrum [7], XP [8], Crystal [9]), and all these methodologies are based on the Agile Manifesto [10]. The Scrum methodology and their variances (e.g. Scrumban, Scrum with other methodologies) is the agile methodology most used in software development projects [11].

1.1. SCRUM

Scrum is used in projects with uncertain requirements and unpredictable risks resulting from the implementation of new technologies and strategies [3, 12]. This methodology has three key roles:

- Product Owner (PO) that is the customer. He is the one with the responsibility on the software functionality specification and to solve any doubts that might arise during development. He is the client's representative that must watch the project closely and help in the construction of a software that answers completely to the client's needs; He also needs to have financial autonomy for decision making.
- Scrum Master (SM) like project manager in traditional development process. He is the responsible to lead the team and to avoid any hurdles that might arise during the process. A hurdle is something that might impede a member from performing his work; and
- Team Members that are composed by developers, database administrator and testers in which each member has a specific skill. Nevertheless, members are not banned from performing task different from their expertise. Thus, the team will become more integrated and teams members will know better the software.

Scrum is based in practices represented by:

- Daily meetings: are performed with the team members standing in front of the kanban, which is a set of cards (post-it) that indicate the status of a specific task, such as, To Do, Doing or Done (Figure 1). Meetings last approximately 15 minutes, and in them, we discuss questions from team members, what everyone intends to do and what were the hurdles found during that day.

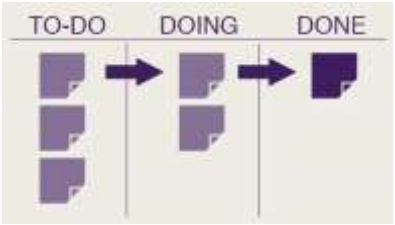


Figure 1 - Kanban example

- Sprint planning meetings: A sprint corresponds to a development cycle and must last from one week and will not take more than 30 days [3] (Figure 2). The goal of the sprint planning meeting is to present the backlog items (i.e. requirements or user stories) and to estimate the tasks. The backlog is a list of software requisites sorted according to their priority, allowing the requisites to be put into development according to their importance. The backlog sorting is performed according to the priority of each item that is calculated from the importance of the functionalities for the client. That way, items with higher priority are implemented before the lower priority ones increasing client satisfaction.

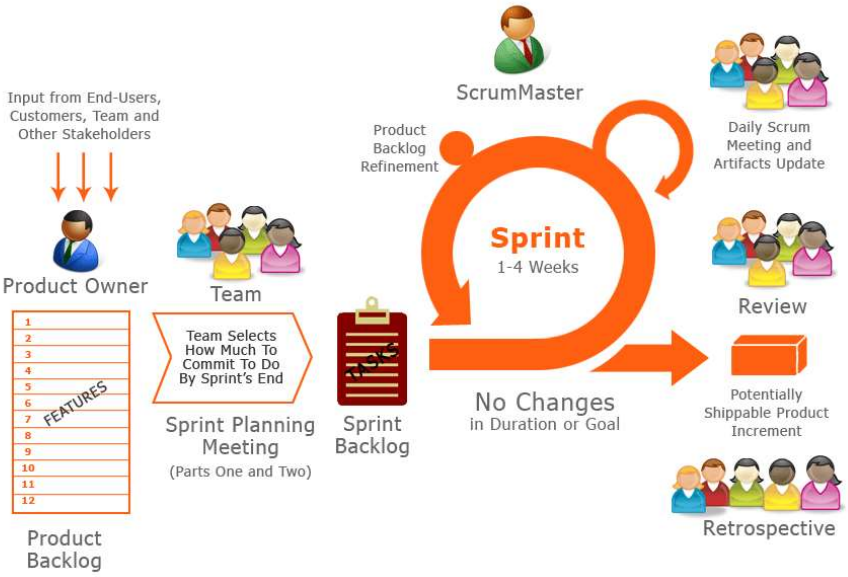


Figure 2 - Scrum methodology. Adapted from [5]

- Estimating product backlog: One of the methods used to estimate tasks is the planning poker (Figure 3), whose goal is to allow each member to choose a card with the task length estimative. The members that choose the smallest and the biggest estimative discuss the reasons why their estimative differ and then there are some rounds until team members come-to a consensus.

Figure 3 - Planning poker cards example

- Sprint review meeting: at the end of the Sprint, the Sprint Review happens, which is a meeting for the delivery of user stories implemented for PO. Thus, the PO gets a software release in which he can evaluate and suggest improvements and changes. A release is a function software version that can be delivered to the client for homologation. For each release, a presentation of the functional part is made to the client. This way, the client can keep up with the project and validate the systems in parts.

- Sprint retrospective meeting: it is a meeting where a retrospective of the sprint is performed, in its positive and negative points identified and analyzed. Hence, it is possible to keep the strong points and create strategies to improve the weaknesses. This way we have feedback from the development team, improve and evolve all team members.

1.2. TEST DRIVEN DEVELOPMENT (TDD)

Currently, there is a recommended technique for software quality assurance that is Test-Driven Development (TDD). TDD is based on short cycles repeated. First, the developer writes an automated test case that defines a desired improvement or a new functionality. So, a code is produced, which can be evaluated by the test. After this, the code should be refactored under acceptable standards [13]. According to [14], TDD encourages simple designs and inspires confidence code. Through TDD, programmers can apply the concept to improving and debugging legacy code developed from ancient techniques [15].

In this case, testing must be an unavoidable aspect of development. From the integration of development and testing, quality is improved [17]. Therefore, testing is a cross-functional activity that involves the whole team and should be continuously done from the beginning of the project [18].

In agile development, two kinds of teams are usually identified: customer team and developer team [3]. The customer team includes business experts, product owners, product managers, and other persons related to the business side of the project. This team communicates and collaborates with the developer team throughout iterations, writing stories, drawing examples, and reviewing finished stories [3].

Everyone involved in delivering code is a developer and is part of the developer team. Agile principles encourage team members to take multiple activities. However, each team needs to decide what expertise its projects require [3]. Testers are integral members of the customer team, gathering requirements, helping the customers to express their needs, and advocating for quality on their behalf. They are also part of the developer team, collaborating with developers to automate tests and assisting them in delivering the maximum business value [3][18].

Nevertheless, testing should not be subservient to development. Testers must have technical and business knowledge, as well as acting autonomously, based on the priorities, complexities, and product needs [17].

2. TAILORING AGILE METHODS

Scrum is not a process or a technique for product development, but an iterative and incremental framework [19]. This framework may be used with different processes and techniques working well in an environment of constant changes [20]. Scrums reveals what might be corrected in the team and its essence is strongly connected to the personality of the team members. This way, one must constantly validate the decisions, practices and process according to the principles and values the team holds dear.

Scrum was adopted because:

- Could be adapted and would answer better to constant change from the client;
- Frequent deliveries could be made with more value to the client, with a focus on the maximization of the return on the investment; and
- Avoid waste and prioritize communication and the visibility of the projects progress, so that team members would always know what needed to be done and what was being done.

Nevertheless, we felt the need to adapt it to IPEV scenario, in order to adequate it to reality and to provide the best return to projects.

Before the Scrum adoption the projects follow up were not done daily. There were only delivery schedules between members and when one deadline was about to expire, the responsible would come and ask for results. In case of danger to the deadlines and subsequent delaying of activities, team members had to do overtime in order to fulfill the deadlines.

In the IPEV, the team is composed of 6 to 12 members, an amount that has been efficient in improving communication. Each project has a PO which is the main researcher of the project. He must have the availability, at least for the meetings. Besides, each project has one SM. The others members make up the team and can be developers, testers, database administrator, professionals dedicated to system documentation and so on. All members in each project are computer science bachelor, engineers or researchers and have at least four years experience in flight tests. One important consideration to improve quality of projects is the amount of person of the software quality. The projects should have one person in software quality to each developer. For example, in small project, the members of the project could be PO, SM, 2 developers, 2 persons in software quality.

The project scope is reviewed at each sprint planning so that the team can dedicate itself to the highest priority tasks. At each review, the client is free to adjust and review the priority of each function. If the main researcher (PO) is in another city or country, the activity planning is done in a conference room. Usually, activity planning includes all team members and least close to four hours, considering a sprint of one month. This activity has three parts:

- The moment when team members decide what is going to be done.
- To debate how the activities will be developed and for the development team to list the necessary tasks to implement the planned activities.
- To estimate each task length, based on a team consensus on values between 1 and 4 hours for each task at hand.

Estimation is done by team members using planning poker cards. Each team member selects a card that he thinks corresponds to the task length and after all cards are chosen, they are exposed. The members that chose the highest and smallest lengths discuss the reasons that took them to make that choice and the cards are played again until the team comes-to a consensus.

During development the team meet daily and each member reports what he did and how he intends to do the next task. In case a member reports on a hurdle, technical issues are discussed briefly after the daily meeting. The idea is to steer the member with the hurdle towards a possible solution.

The place of the daily meeting is in the development environment itself, where the information on project progress is stuck to the walls, such as burndown, product backlog, sprint backlog and error report. Furthermore, all information about the project progress are available in Trello. It allows to manage project and to give visibility for all members of the project. The artifacts and documentation are available in a file server also all members. This plain sight management intends to make available all necessary information in a simple and easy to understand way. This way, work becomes less arduous and the quality of the software created increases.

Daily meetings do not happen at a fixed time, alternating between mornings and afternoons. The time is a consensus between developers that have flexible work hours in order to have all members in all meetings.

In spite of the team and the environment being self managed (or self organized), there are some small attributions and task delegation. Control function belongs to all team members, which choose the best way to work and to fulfill the project goals. In case a member finds a difficulty in a task or encounters a hurdle, he can ask for the help of the team, which can help him if available. The group member that knows about the domain at hand can help him on the specific technical issue.

3. EXPERIMENT AND RESULTS

In this paper, we show the use of Scrum and TDD in the development of an experimental tool that aims to make the calibration in real time, of the rudder of a fighter aircraft. The calibration operations are based on a comparison of standard instruments in order to determine their accuracy and correctness check if this continues according to the specification of the manufacturer. From the calibration, it is possible to know the behavior of the instrument or device, quantifying the systematic errors that it presents, and thus can reduce and achieve more reliable results from the same already calibrated. The figure 4 contains a partial structure of the aircraft.

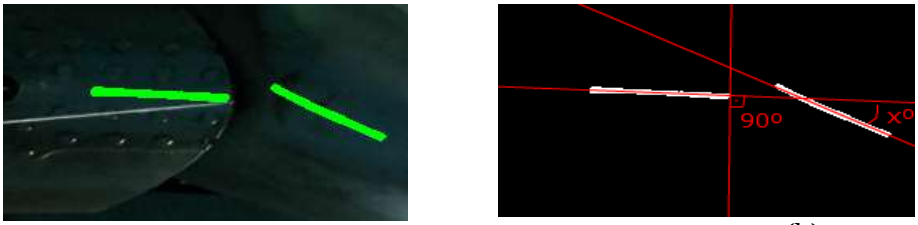
This tool was developed using Matlab environment and C++ programming language with OpenCV [14]. Trello was used to manage of project progress. In addition, we used a version control software (VCS) that enabled collaborative software development and retention of historical changes in files. For this, Subversion was the tool selected. The bugs found during the development cycle of the software were reported in JIRA. At the start of each sprint, the shippable software used the agile testing quadrants [3]. The members automated unit tests using Matlab unit testing framework and Google C++ Testing Framework.

Communication among team members is constant and iterative, according to the Scrum methodology. That means that meetings are scheduled daily in non fixed time slots with all team members in a single room, with all members standing. Communication is performed preferably face to face instead of using written documents and the project team works in a single room, in order to increase interaction among its members.

Given the integration between development and testing teams, it was not necessary to wait does Sprint functions release for the testing to begin. In a single project the testing team used the TDD (Test Driven Development) technique. Hence, as soon as a system function was finished, the test ran and in case of problems, the correction was requested.

Figure 4 - Partial structure of the aircraft

This tool was developed using Linux operating systems and algorithms implemented in OpenCV. This tool makes reading frames of a video camera, using techniques of image processing and pattern recognition to allow calibration in real time of the rudder. The figure 5a contains the markers on the aircraft used to calibrate the rudder and figure 5b contains a frame processed by the application, showing the diagram used to calculate the slope.



(a) (b)

Figure 5 - (a) Original image (b) Frame processed by the application

In this project we had 45 user stories, totaling 186 story points. This product backlog was divided into 3 sprints with 18, 15 and 12 user stories in each sprint, respectively. Each sprint had 64, 60 and 62 story points, respectively. Normally, each sprint should have the same quantity of story points and the same quantity of days to measure the team productivity. The quantity of user stories can be different because each story point can have a value (story point), complexity, effort and uncertainty different. The each sprint had 22 days of work and we estimated 72 story points per sprint. Thus, the first and second sprint had 72 story points and the third had 42 story points. The burndown chart of sprint #1 is shown in Figure 6. The first day of sprint was used to sprint planning and the last day was used to sprint review.

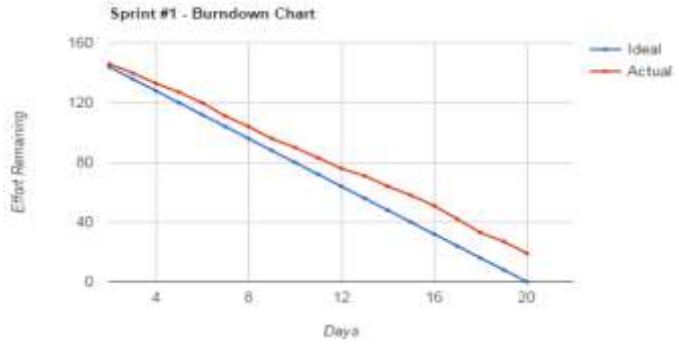


Figure 6 - Burndown Chart of Sprint #1

We can observe in figure 6 that the actual line (red line) is above of the ideal line. It shows that in the first sprint, the team was in debt (i.e. the team failed to deliver all user stories). Three user stories have not been completed in sprint #1. This happened because of the error in the estimation of some user stories. In the second sprint the debt was one user story. In the last sprint, debts of the sprints #1 and #2 were accumulated by the amount of story points had not yet been completed. In the sprint #3, initially, we had 42 story points, however, rose to 64 story points. That was enough to complete all the user stories in three sprints.

CONCLUSION

The use of TDD and Scrum allowed to understand the coverage testing of the software and hence the quality of the tool. This calibration process was never realized by the PO, thus, in research and development activities the changes are frequently and acceptable. Furthermore, we observed a higher customer satisfaction because it allowed his participation next to team members and visualization of the intermediate versions of the tool. This brought great advantage because changes requested by the client were accommodated during the development of the tool.

In conclusion, we can state that the adaptations we performed were flexible schedule for the daily meeting and the integral presence of a team member (the product owner) in the client, as a consequence of having a representative of the client involved in the project.

The next steps include the consolidation of the adopted practices, for adaptations and corrections of the deviations identified during development in other projects. We also need to use metric to evaluate formally the gain achieved by using the agile methodology Scrum.

ACKNOWLEDGMENT

The authors would like to thank the unconditional support given by the Instituto de Pesquisas e Ensaios em Voo (IPEV), Instituto Tecnológico de Aeronáutica (ITA) and Instituto de Aeronáutica e Espaço (IAE). Also, we'd like to thank FINEP under agreement 01.12.0518.00 that funded the development of this work and the presentation trip.

References

[1] T. Stober, U. Hansmann, "Agile Software Development. Best Practices for Large Software Development Projects," Springer-Verlag Berlin Heidelberg, 2010.

[2] T. Gilb, "Evolutionary delivery versus the waterfall model," ACM SIGSOFT Software Engineering Notes,vol.10, no.3,pp.49–61,1985.

[3] L. Crispin and J. Gregory, "Agile Testing: A Practical Guide for Testers and Agile Teams", Addison-Wesley Professional, 2009.

[4] CHAOS. "The CHAOS Manifesto. Agile Succeeds Three Times More Often Than Waterfall", The Standish Group, 2012.

[5] Agile Buddha. http://www.agilebuddha.com/trainings-workshops/scrum-training-workshop/. Last access: 20 jun 2015.

[6] Extreme Programming: A Gentle Introduction. http://www.extremeprogramming.org.

[7] Crystal Methodologies, http://alistair.cockburn.us/Crystal.

[8] Manifesto for Agile Software Development. http://agilemanifesto.org.

[9] VersionOne, “Survey: The 7th Annual State of Agile Development Survey”, http://www.versionone.com/pdf/7th-Annual-State-of-Agile-Development-Survey.pdf, 2012.

[10] M. JAMES, “Scrum Reference Card”, CollabNet Inc., 2010.

[11] W. E. Perry, “Effective Methods for Software Testing”, Wiley Publishing, Inc, 2006.

[12] K. Beck. “Test-Driven Development by Example”, Addison Wesley - Vaseem, 2003.

[13] M. Feathers. “Working Effectively with Legacy Code”, Prentice Hall, 2004.

[14] OpenCV (Open Source Computer Vision). http://opencv.org.

[15] M. Guglielmi, S. Lascar, V. Mastrocola, E. Williams. “A new method for evaluating R&D effectiveness”, 55th International Astronautical Congress, Vancouver, Canada, DOI: 10.2514/6.IAC-04-U.3.B.02, 2004.

[16] Beck, K.; Beedle, M.; Bennekum, A. van; Cockburn, A.; Cunningham, W.; Fowler, M.; Grenning, J.; Highsmith, J.; Hunt, A.; Jeffries, R.; Kern, J.; Marick, B.; Martin, R. C.; Mellor, S.; Schwaber, K.; Sutherland, J.; Thomas, D. Manifesto for Agile Software Development. 2001. Available at: <http://www.agilemanifesto.org/>. Last access: 20 jun.2015.

[17] J. A. Whittaker, J. Arbon, and J. Carollo, “How Google Tests Software,” Addison-Wesley Professional, 2012.

[18] J. Humble and D. Farley, “Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation,” Addison Wesley Professional, 2010.

[19] Sutherland, J.; Schwaber, K. The Scrum Papers: Nut, Bolts, and Origins of an Agile Framework. 224p. 2011

[20] Schwaber, K.; Sutherland, J. The Scrum Guide. 2010.