Revista de Sistemas de Informação da FSMA
n. 9 (2012) pp. 16-23

SISTEMAS DE INFORMAÇÃO
http://www.fsma.edu.br/si/sistemas.html

# Adapting and Using Scrum in a Software Research and Development Laboratory

Igor Ribeiro Lima, Tiago de Castro Freire, Heitor Augustus Xavier Costa

***Abstract* — Agile software development has gained importance in the industry because of its approach on the issues of human agility and return on investment. This paper shows how Scrum agile software project management methodology has been deployed and adapted to the model of software project management of a research and development laboratory. As a result of this deployment, experiences and lessons learned in seven real projects developed by the authors are reported.**

***Keywords* — Scrum, Project management**

## I. Introduction

Software development organizations have become more interested in agile methodologies, whose focus is client collaboration, individual value and adaptation to change. This interest has grown because these methodologies have shown productivity gains in several different software development project types.

The choice of the most adequate software methodology for software development neither is a trivial task nor guarantees the project's success. Nevertheless, agile methodologies have caught the eye of software companies, given the evidence of the productivity increase they provide [4].

The original movement that helped improve the software development sector introduced the methodology idea, that is, a disciplined approach for software development with the goal of making the process more efficient and predictable [3].

The definition of an agile methodology was created in February 2001 in a meeting of software process methodologists that resulted in what is now known as the Agile Manifesto [8]. This manifesto is a simple and concise declaration that seeks to change the traditional lens that has been used to see software development. Its intention is to value [9]: i) of individuals and interactions over tools and processes, ii) of working software over detailed documentation, iii) of client collaboration over contract negotiation and iv) of change adaptation over plan following. The Agile manifesto is based on 12 principles [8]:

1. Making customer satisfaction a priority though continuous and frequent deliveries;
2. Embrace requisite change, even in an advance project phase;
3. Deliver software frequently, in the smallest possible time frame;
4. Create synergy between the business and development teams in order to allow them to work together daily;
5. Keep a motivated team providing the environment, the support and confidence needed;
6. Allow efficient information spread through face to face conversation;
7. Having a working system is the best progress measurement;
8. Promote sustainable development through agile processes;
9. Continuous attention to technical excellence and to a good project increase agility;
10. Be simple;
11. Allow teams to self organized using the best architectures, requisites and projects.
12. Make a reflection in regular intervals on how to become more efficent and adjust and optimize behavior.

Igor Ribeiro Lima is a graduate student at the Systems Engineering coruse (PPGESIS) at the Departament of Engineering (DEG) of the Federal University of Lavras (UFLA) and a project manager at the Research and Development lab where the study was conducted. E-mail: igorlima@comp.ufla.br.

Tiago de Castro Freire a project manager at the Research and Development lab where the study was conducted. E-mail: kuruma@bsi.ufla.br.

Heitor Augustus Xavier Costa is a full professor at tne Computer Science Department (DCC) of the Federal University of Lavras (UFLA). E-mail: heitor@dcc.ufla.br.

In literature we can find several initiatives for the development of software that incorporate such principles, such as *Extreme Programming* (XP) [19, 20, 22, 23, 24, 25], Scrum [1, 20, 25, 26, 27, 28], Crystal [1, 13, 14], *Feature Driven Development* (FDD) [1, 20, 21, 22], *Dynamic Systems Development Method* (DSDM) [15, 17, 19] e *Adaptative Software Development* (ASD) [15, 16, 17, 18].

In this paper, our goal is to present the experience and the thought process of the development team at a Research and Development Lab on the use of the agile methodology Scrum for software project management that included constant changes and interventions from the client involved in those projects. We will discuss, for instance, the way software architecture proposals and ways to document requisites arise, which is due to the fact that team members can be assigned to different tasks (dynamic allocation) which allows for a macro view of the software. Besides, daily meetings also allow for team members to find direct solutions, because there is involvement of all members, which allows them to become more effective in their work.

The agile methodology Scrum was chosen for out Lab because it is the most used by the companies that adopt the agile principles [10, 11, 12]. This choice was also due to the adaptability of this methodology and to the fact that it can respond quickly to constant changes in software projects.

This work is organizes as follows. Scrum concepts and fundamentals are described briefly in section 2. The consolidation of the agile culture in the Research and Development and the search for a unified, adapted and adequate process are shown in section 3. The lessons learned using the agile methodology are described in section 4 and final considerations are presented in section 5.

## I. Scrum

Scrum was developed by Jeff Sutherland in 1993 [28] and its goal is to be a development and management methodology that follows the principles of the agile methodology. The Scrum team is composed by [5]:

- Team: its the development project team, composed by up to ten developers 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, minimizing the impact of another member's dismissal.
- *Product owner*. 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.
- *Scrum master*. 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. For instance, requests to perform activities not related to the project, problems in the test server, difficulties with the technology and unplanned requisites might be examples of hurdles that might cause problems to the *sprint.*

In our lab, the team is composed of 4 to 7 members, an amount that has been efficient in improving communication. The product owner is a member of the lab that stays constantly at the client, a reversal of the usual strategy that was due to the fact that it was difficult to keep the client constantly in the lab. The Scrum Master for each project is selected by the Lab's IT coordinator.

Scrum is based in practices represented by (i) daily meetings, (ii) sprint planning meetings, (iii) sprint review meeting, (iv) backlog sorting and (v) release presentation [2]. Daily meeting 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 so that the Scrum Master becomes aware of them and may eliminate them [2].

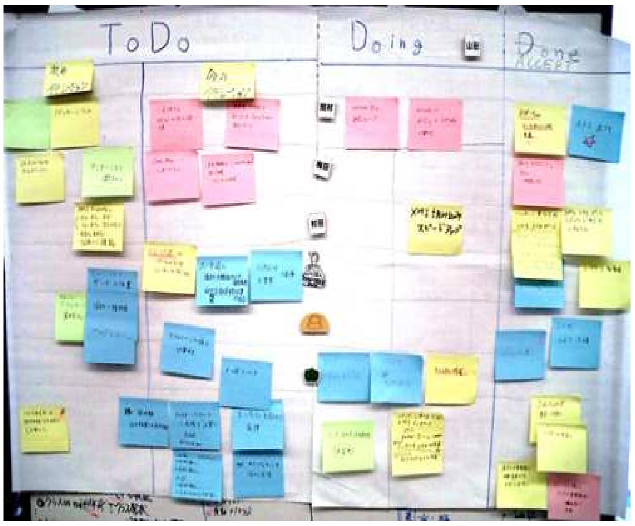

Figure 1 - *Kanban* [6]

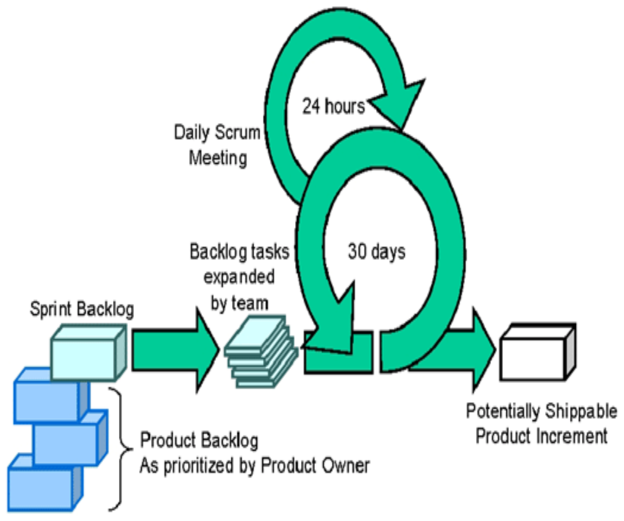
24 hours
Daily Scrum Meeting
30 days
Backlog tasks expanded by team
Sprint Backlog
Product Backlog As prioritized by Product Owner
Potentially Shippable Product Increment

Figure 2 - *Sprint* [7]

A sprint corresponds to a development cycle and must last from two weeks to a month (Figure 2). The goal of the sprint planning meeting is to present the backlog items 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.

One of the methods used to estimate tasks is the *planning poker* (

Figura 3), whose goal is to allow each member to choose a card with the task length estimative. The members that choose the smaller and the bigger estimative discuss the reasons why their estimative differ and then there are up to nine rounds until team members come to a consensus [2].

The sprint review is performed in a meeting where a retrospective of the sprint is analyzed, 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 and improve and evolve all team members [2].

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 [2].

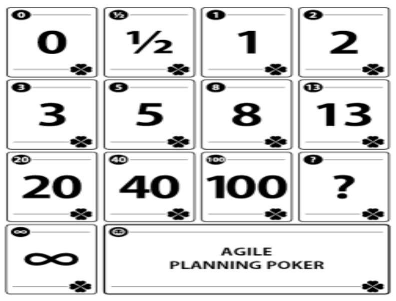

Figura 3 - *Planning Poker* [29]

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.

## II. SCRUM ADAPTATION

Scrum is not a process or a technique for product development, but an iterative and incremental framework [28]. This *framework* may be used with different processes and techniques working well in an environment of constant change [5].

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 [4]. Scrum was adopted after some lab members have heard reports of similar experiences in scientific events that approached the topic of agile software project development and management.

After an informal debate on whether it should be chosen for our Lab, we came to the conclusion that Scrum could be adapted and would answer better to constant change from the client (*Agile Principle 1*). Hence, we could have frequent deliveries with more value to the client, with a focus on the maximization of the return on the investment (*Agile Principle 3*). Besides, it would 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 (*Agile Principle 4*). Nevertheless, we felt the need to adapt it to the scenario where it was adopted (a Research and Development Lab), in order to adequate it to reality and to provide the best return to our clients.

Before the adoption of Scrum there were no well defined and well established development and management processes in the Lab. Project follow up was not done daily – there were only delivery schedules between teams 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 other words, the Lab had no previous solid experience in the process of software development. Probably, one of the reasons for that was the fact that the Lab was quite recent and has no long experience on software development.

During the adaptation of Scrum, we paid attention to identify which changes in the organization culture were necessary to adopt the concepts and principles of agility. For that, we considered the four values we pointed out before on the agile manifesto. For this agile culture to become consolidated, we needed to search for an unique process, adapted and adequate to the team reality.

### A. *Multiple Projects*

Because of corporate secrecy, the location of the Research and Development Lab, the names of the developed projects and our clients will be omitted.

All projects were developed in a Software projects Lab inside a University in the State of Minas Gerais. In this Lab, the teams are organized according to the technical abilities and the availability of its members.

These projects are software systems to help with the environmental management of a governmental organization from the state of Minas Gerais.

### B. *Caracterization of Teams and Used Technologies*

Project teams are, in general, composed by three to five programmers, one or two database administrators, one business analyst, one or two professionals caring for testing and one to three professionals dedicated to system documentation.

The development team in each project is composed by experiences developers and interns and in each project we have at least two professionals with at least three years experience while the database and business analysis teams have at least two years experience. The testing and documentation teams are composed by a mix of experienced professionals and interns.

Each team's composition depends on the complexity of the project. One of the high complexity projects consisted in perform geoprocessing of a specific area of Minas Gerais. In this team, the project was done by five programmers, two database administrators and five persons in software quality.

Another project that was not as complex as the former consisted in filling simple databases for form emission. In this project, we need only three programmers, one database administrator and two persons in software quality.

A third project of average complexity consisted in a virtual environment to support the teaching and learning process in the Minas Gerais economic and ecologic zoning system distance learning. In this project, we required two programmers, one database administrator and a single person in software quality.

All projects were developed in the Java programming language (J2EE - *Java 2 Enterprise Edition*), using MVC frameworks (*Model - View - Controller*), Spring[1] and VRaptor[2] and in the front-end layer we used Flex, JavaScript and JSP (*Java Server Page*). Oracle[3] and Postgres[4] were used as the database management systems.

Communication among team members is constant and iterative, according to the Scrum methodology. That means that meetings are scheduled daily in 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 a single project team works in a single room, in order to increase interaction among its members (*Agile Principle 6*).

Given the integration between development and testing teams, it was not necessary to wait doe sprint functions release for the testing to begin. In a single project the testing team used the TDD (*Test Driven Development*) technique. This project was the least complex one, because we needed to let the testing team understand better the process. In the other projects we used unit and behavioral tests. Hence, as soon as a system function was finished, the test ran and in case of problems, the correction was requested.

### C. *Adapting Scrum*

The project scope is reviewed at each sprint panning so that the team can dedicate itself to the highest priority tasks (*Agile Principle 7*). At each review, the client is free to adjust and review the priority of each function (*Agile Principle 2*).

Activity planning is done in a conference room. Usually, activity planning includes all team members and least close to eight hours, being divided in three parts (*Agile Principle 10*). The first part is the moment when team members decide what is going to be done. The second part is to debate how the activities will be developed and for the development team to list the necessary tasks to implement the planned activities. The third part is to estimate each task length, based on a team consensus on values between 1 and 24 hours for each task at hand.

Estimation is done by team members using cards with the Fibonacci sequence (*planning poker*). 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. Values outside the Fibonacci sequence are chosen when there is no consensus after three rounds of poker. Hence, teams may come to a consensus using intermediary values.

During development the team met daily and each developer reports what he did and how he intends to do the next task. In case a developer reports on a hurdle, technical issues are discussed briefly after the daily meeting. The idea is to steer the developer 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 (*Agile Principle 12*).

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 (*Agile Principle 9*).

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.

[1] http://www.springsource.org/
[2] http://vraptor.caelum.com.br/
[3] http://www.oracle.com/index.html
[4] http://www.postgresql.org/

In spite of the team and the environment being self managed (or self organized), there are some small attributions and task delegation (*Agile Principle 11*). 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.

An example of this collaboration was a situation when a programmer with little experience on the geographic processing library was helped by other team mates (*Agile Principle 5*). A more experienced programmer that knew this library realized that other team members were also not experts in using this library, so he dedicated some time to help them use the functionality it provided.

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 (*Agile Principle 8*).

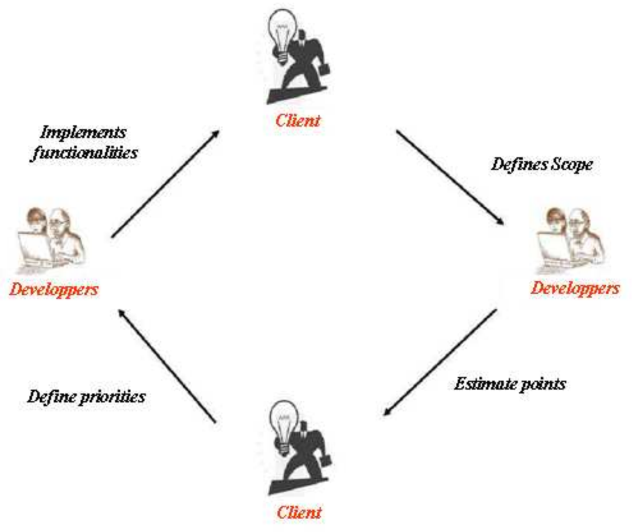

Figure 4 – Life Cycle of the Project Activities (adapted from [2])

## III. LESSONS LEARNED

As we mentioned before, agile practices can bring problems to evidence during its implantation.

Figure 4 illustrates the life cycle of a project and the iteration between developers and clients according to the agile methodology, which has the same iterations: i) the client defines the scope; ii) the developers estimate deadlines; iii) the client defines the priorities; iv) the developers implement the functions and we go back to the beginning of the cycle where the clients redefines the scope until all functionality desired is delivered.

In our practice, it became clear that our lack of experience with the agile methodologies made us misinterpret the client's needs, which reflected on the software development, because development team members had no direct access to the client. The solution was the creation of more artifacts, as the business rule workflow, that required the presence of the client, the Product Owner and all members of the development team.

Another thing that came into evidence was the need to collaborate and the interest from each team members. Their efficiency was directly related to the commitment of each collaborator. If there was a team member that did not share the same idea and philosophy, there was a good amount of friction with the rest of the team, overloading one team member.

Eventually, some individuals that had difficulty working in teams and with no characteristics of pro-activity would refuse to help other team members with the argument that it was not their task. There was also lack of compromise of some team members to perform the tasks that were assigned to them, creating a hurdle to other team members that were assigned dependent tasks.

The teams members are both professionals and interns, the latter being students at the university where the lab is located. In spite of the difficulties found by the students, we could notice that there are a learning curve pointing higher, because they learned to deal with the practices that are involved in software development, such as documentation, testing, implementation and client meetings.

Only two out of the seven projects mentioned in this experiment did not deliver in the agreed schedules, which were renegotiated with the client. We have no measurements in order to compare with software houses outside the academia, but we believe that these delays were caused by the lack of experience of team members with Scrum.

The major issues that caused the delays in those two projects were: (i) lack of commitment from part of the team, (ii) team members that took on more tasks than they were able to deal with and (iii) lack of transparency and communication among team members.

We decreased the delays by improving the criteria for team members selection, allow for a higher success rate.

Lessons learned included the following requirements for the usage of Scrum in an organization: (i) present the framework for all that will use it, so that the team members know how it works and that they will have some additional tasks in their dailiy routine, (ii) create a pilot project for the implantation of the framweork, in order to select personnel, (iii) observe the possible impacts, (iv) structure training and (v) evaluate the benefits of the adopted and implanted Scrum methodology. During the implantation, good practices such as continuous integration and test driven development are opportunities to improve software quality and to make it easier to refactor applications (conclusions to which we came in the project where TDD was used).

It is also important to point out that Scrum usually shows organizational problems, which become hurdles that may at first cause unnecessary friction until the complete understanding of what is a real hurdle.

The biggest problems we found were related to pro-activity and team members' collaboration. There are persons that feel the need of another one to hold them responsible and to solve the problems that arise. This shows lack of commitment, of individuality and of initiative. Persons that think as individuals and not as a team must be excluded from the development team.

From the adaptations in the framework, we must understand that some of its essential characteristics must be preserved, such as: i) the intense communication cycle that guarantees the expectations alignment; ii) the constant scope review that will guarantee project cohesion; iii) the intermediate deliveries that seek to understand the client's needs and to correct any failures before the project comes to an end and iv) the concept of timebox, that forces the team to deliver, avoiding delays.

The benefits come from the principles embodied in the Scrum agile methodology and when adapting it, one must keep the following principles in order to get the best results from the framework: i) strong interaction among people; ii) communication; iii) commitment; and iv) product.

## IV. Final Thoughts

Software development using agile methodologies is becoming a bigger reality in the daily life of software development companies. Agility brings quality to the software development and management process. In order to add value to the final software, one must have a well structures team that follows the methodology and uses correct strategies.

Nowadays in Brazil there is a great interest in the development of new software and patents so that national technological production may be compatible with the scientific production, as measured by national and international papers. The adoption of modern software projects development and management techniques such as Scrum may help lowering this gap, establishing a bridge between quality science and products that effectively solve problems of the national business reality.

Scrum brings an iterative and incremental development process for agile software development and management that stands on four pillars: i) individuals and their iterations are more important than procedures and tools; ii) working software is more important that a complete documentation; iii) collaborating with clients is superior to contact negotiation; and iv) the ability to respond to change is more important that having a pre-established plan.

In the context of our Research and Development Lab the senior researchers ("project managers") supported fully the implementation of Scrum, the basic infrastructure was adequate and the team members understood the new proposal and incorporated it in their tasks.

It is important to point out that using Scrum contributed to the education of team members that were interns (students) and/or autonomous professionals (freelancers).

Teams were usually composed of four to seven members, which makes communication easier. An important adaptation was the inversion of the semantic of the product owner, for in our context he is a member of the lab allocated at the client. This change was made because of the difficulties associated with having a client in the lab.

Based on the analysis of the implantation of Scrum for project development, we could see palpable change in software projects management and development, allowing for easier perception of progress. The involvement and commitment of members of the team with the results increased, allow for more collaborative work.

We also realized that team members were motivated and open to changes in work, which facilitated the process of implementing and adapting Scrum. Hence, it allowed for growth and improvement in the process, in order to answer to the peculiarities of each project.

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

## References

[1] Abrahamsson, P.; Salo, O.; Ronkainen, J.; Warsta, J. Agile Software Development Methods – Review and Analysis. VTT Publication 478. 107 p. 2002. Available at: <http://www.vtt.fi/inf/pdf/publications/2002/P478.pdf>. Last access: 21 jan 2009.

[2] Bona, C. Avaliação de Processos de Software: Um Estudo de Caso em XP e ICONEX. Dissertação (Mestrado em Engenharia de Produção) – Universidade Federal de Santa Catarina, Florianópolis – 2002. Available at: <ftp://www.ufv.br/dpi/mestrado/Gerais/TeseIconix.pdf>. Last access : apr.2011.

[3] Fowler, M. The New Methodology. 2005. Available at: <www.martinfowler.com/articles/newMethodology.html>. Last access: apr.2011.

[4] Leitão, M. V. Aplicação de Scrum em Ambiente de Desenvolvimento de Software Educativo. Monografia (Trabalho de Conclusão de Curso) – Universidade Federal de Pernambuco, Recife – 2010. Available at: <http://dsc.upe.br/~tcc/20101/TCC_final_Michele.pdf>. Last access: apr.2011.

[5] Schwaber, K.; Sutherland, J. The Scrum Guide. 2010. Available at: <http://www.scrum.org/scrumguides/>. Last access: mar. 2011.

[6] Hiranabe, K. Kanban Applied to Software Development: from Agile to Lean. 2008. Available at: <http://www.infoq.com/articles/hiranabe-lean-agile-kanban>. Last access: apr. 2012.

[7] Murphy, C. Adaptive Project Management Using Scrum. In: Methods & Tools - Software Development Magazine -

Programming, Software Testing, Project Management, Jobs. 2004. Available at: <http://www.methodsandtools.com/archive/archive.php?id=18>. Last access: apr. 2012.

[8] 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: 16 abr.2012.

[9] Vijayasarathy, L. R.; Turk, D. Agile Software Development: A Survey of Early Adopters. In: Journal of Information Technology Management, v. XIX, n. 2, p. 1-8. 2008.

[10] Sutherland, J.; Viktorov, A; Blount, A. Distributed Scrum: Agile Project Management with Outsourced Development Teams. In: Proceedings of the 40th Hawaii International Conference on System Sciences, 40, 2007.

[11] Catunda, E.; Nascimento, C.; Cerdeiral, C.; Santos, G.; Nunes, E.; Schots, N. C. L.; Schots, M.; Rocha, A. R. Implementação do Nível F do MR-MPS com Práticas Ágeis do Scrum em uma Fábrica de Software. In: X Simpósio Brasileiro de Qualidade de Software (SBQS). 2011. Available at: <http://www.lbd.dcc.ufmg.br/colecoes/sbqs/2011/SBQS2011-RE10_82940.pdf>. Last access: 23 apr 2012.

[12] Salgado, A.; Melcop, T.; Acchar, J.; Rego, P. A.; Ferreira, A. I. F.; Katsurayama, A. E.; Montoni, M.; Zanetti, D. Aplicação de um Processo Ágil para Implantação de Processos de Software baseado em Scrum na Chemtech. In: IX Simpósio Brasileiro de Qualidade de Software (SBQS). 2010. Available at: <http://www.lbd.dcc.ufmg.br/colecoes/sbqs/2010/RL10_alex_salgado.pdf >. Last access: 23 apr 2012.

[13] Cockburn, A. Agile Software Development. Addison-Wesley Professional. 304 p. 2001.

[14] Cockburn, A. Crystal Clear – A Human-Powered Methodology for Small Teams, including The Seven Properties of Effective Software Projects. 2004. Available at <http://st-www.cs.uiuc.edu/users/johnson/427/2004/crystalclearV5d.pdf>. Last access: 17 feb 2009.

[15] Highsmith, J. (2002); Agile Software Development Ecosystems; Publisher: Addison Wesley; Pub Date: May 26, 2002; ISBN: 0-201-76043-6; Pages: 448.

[16] Paetsch, F., Eberlein, A., and Maurer, F. Requirements Engineering and Agile Software Development. In Proceedings of the Twelfth international Workshop on Enabling Technologies: infrastructure For Collaborative Enterprises (June 09 - 11, 2003). WETICE. IEEE Computer Society, Washington, DC, 308.

[17] Filho, D. L. B. Experiências com desenvolvimento ágil, Instituto de Matemática e Estátistica da Universidade São Paulo, 170p, Masters dissertation.

[18] Portela, C. S. Uma proposta de gerenciamento ágil dos projetos de desenvolvimento de software do CTIC / UFPA, Instituto de Ciências Exatas e Naturais – Faculdade de Computação – Universidade Federal do Pará; 94p, Masters dissertation.

[19] Coram, M; Bohner, S. The Impact of Agile Methods on Software Project Management. 2005.

[20] Awad, M. A. A Comparison between Agile and Traditional Software Development Methodologies. Technical Report. University of Western Australia. 77p. 2005.

[21] Palmer, S. R.; Felsing, J. M. A Practical Guide to Feature-Driven Development. Prentice-Hall. 304 p. 2002.

[22] Hunt, J. Agile Software Construction. Springer. 254 p. 2005.

[23] Lindstrom, L.; Jeffries, R. Extreme Programming and Agile Software Development Methodologies. In: Information Systems Management, v. 21, Issue 3, pp 41-52. 2004.

[24] Beck, K. Embracing Change with Extreme Programming. In: Computer, v. 32, Issue 10, pp 70-77. 1999.

[25] Costa Filho, E.; Penteado, R. A. D.; Silva, J.; Braga, R. Padrões e Métodos Ágeis: Agilidade no Processo de Desenvolvimento de software. In: 5th Latin American Conference on Pattern Languages of Programming. August 16-19, 2005.

[26] Schwaber, K. SCRUM Development Process. In: Object-Oriented Programming, Systems, Languages, and Applications – Workshop on Business Object Design and Implementation. October 15-19, 1995. Available at: <http://www.jeffsutherland.com/oopsla/schwapub.pdf>. Last access: 17 feb 2009.

[27] Schwaber, K.; Beedle, M. Agile Software Development with SCRUM. Prentice-Hall. 158 p. 2001.

[28] Sutherland, J.; Schwaber, K. The Scrum Papers: Nut, Bolts, and Origins of an Agile Framework. 224p. 2011. Available at: <http://jeffsutherland.com/ScrumPapers.pdf>. Last access: 24 apr 2012.

[29] Tekool.net. Printable Agile Planning Poker. Available at: < http://www.tekool.net/blog/2009/07/21/printable-agile-planning-poker/>. Last access: apr. 2012.