CRITICAL ANALYSYS OF THE SCRUM PROJECT MANAGEMENT METHODOLOGY

Năftănăilă Ionel

The Academy of Economic Studies Bucharest, Management Faculty, 6 Romana Square, Sector 1, Bucharest, Management Chair, E-mail: ionel@naftanaila.ro, Phone: +40751080037

Abstract: Considering the fast-paced changes in the economic and technological environment, the organizations are placed under increasing pressure for agility and efficiency. As a consequence, the traditional structures tend to be replaced by newer, flexible structures, adapted to various events occurring inside and outside companies. With regard to Information Technology in particular, the science of project management has become crucially important. The paper critically analyzes one of the most modern project management technologies in this field – SCRUM, by identifying its strengths and weaknesses from both a theoretical and a practical point of view. The paper is relevant from a scientific perspective because it can be considered as a starting point for researchers looking into project management methodologies; the paper is also relevant for practitioners, by presenting the main benefits and disadvantages of using SCRUM in Information Technology companies.

Keywords: SCRUM, Project Management, Software development, Information Technology, AGILE, Methodology, Iterative

Introduction

The importance of research

The scientific literature abounds of examples in which the success of projects drive the success of companies, or, the other way around, the failure of a project puts the company out of business (Charette, 2005), (Voas & Whittaker, 2002), (Jones, 1995). As a consequence, minimizing risk and approaching projects in a structured manner, become critical success factors. By using a proper methodology, the project managers increase the probability to deliver what the client wants, while accommodating the time and budget restrictions.

Therefore, research in Information Technology Project Management becomes increasingly important, any scientific work in this field having the potential to significantly improve the effectiveness and efficiency of the organizations of this field, with regard to the three basic restrictions: time, financial and scope.

Description of SCRUM

The SCRUM approach is used in the top companies in the field of software development, and has a significant success rate. Analysts of the field consider that SCRUM can be appropriate also for other types of software development companies, in order to benefit from using object oriented tools and techniques(Aberdeen Group, 1995).

The SCRUM approach (or methodology) is based on process management; these processes can be defined, or they can be considered as “black boxes” (Takeuchi & Nonaka, 1986). The name SCRUM comes from the English name of a formation of players in the rugby game – a tight formation in which the players are united in a specific position.

SCRUM represents an improvement of the iterative and incremental approaches, in order to deliver object-oriented software. The iterative methodologies have been initially discussed by Pittman (Pittman, 1993) and afterwards developed by Booch (Booch, 1995).

SCRUM is a management methodology, developed in order to improve and maintain an existing system or a production prototype. This methodology assumes the existence of a project and some source code sequences, which almost always exist in the object-oriented software development due to class libraries. SCRUM is not addressing to development efforts for completely new or legacy systems.

The releases in SCRUM are planned taking into consideration the following variables:

- User requests – what are the needed improvements for the current system
- Time pressure – which is the needed timeframe for obtaining a competitive advantage
- Competition – which are the predictable actions of the competition, and what is needed to prevent them most efficiently
- Quality – what is the needed quality considering the above
- Vision – what changes are needed in this phase in order to realize the desired system
- Resources – what human and financial resources are available.

These variables form the initial plan for an Information System improvement. However, these variables are also altered during the project, and a successful development methodology needs to consider these variables, along with their dynamic nature.

The main difference between the traditional methodologies (waterfall approach, spiral approach or iterative approach) and the empirical approaches (such as SCRUM) is that the latter assume that the analyze, design and development processed in the SPRINT phase are unpredictable. A control mechanism is needed to manage this unpredictability and to control risks.

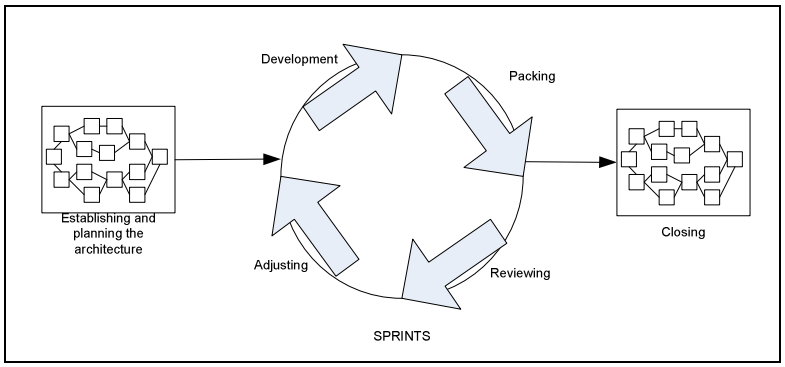

Figure 1 – SCRUM Methodology (Schwaber, 1996)

Some general features of the SCRUM methodology are:

The first and the last phase (planning and closing) consist of defined processes, where all the inputs, outputs and processes are well structured. The knowledge regarding these processes is explicit.

The SPRINT phase is an empirical process. Many of the processes in the SPRINT phase are non-identifiable or impossible to control; as a consequence, the SPRINT phase is treated as a black box which does not need external control. The control, including risk management, is placed on each iteration of the SPRINT, in order to avoid chaos and to maximize flexibility.

SPRINTS are non-linear and flexible. Where possible, explicit knowledge regarding the process is used; if not possible, trial and error methods are used.

The project is open to further development until the closing phase. The deliverable can be changed at any time during Planning and SPRINT phases. Thus, the project remains adaptable to the complexity of the environment, including the competitive pressure, time pressure, quality or financial pressures during these phases.

The deliverable of the project is determined based on the environment, during the project.

In the SCRUM terminology, a SPRINT is a set of development activities which are undertaken during a pre-determined timeframe, usually from one to four weeks. The interval depends on the complexity of the product, on the risk assessments and on the needed degree of skills and expertise. The speed and the intensity of a SPRINT are determined by its agreed duration. The risk is assessed continuously and permanently, and adequate measures are taken for each risk event. Each SPRINT involves one or more teams which perform the following:

Development: Defining the needed changes in order to implement the requests from the journals into packages, opening the packages, analyzing the field, designing, developing, deployment and testing, and also documenting the changes.

Packaging: closing the packages, creating an executable version of the changes and the way in which these fulfill the requests coming from journals. Reviewing: all the teams meet to present the evolution, to evaluate the evolution, to raise and solve problems and to add new entries to the journals. Adjusting: Consolidation of the information discussed in the reviewing meetings.

Each SPRINT is followed by a review, which has the following characteristics:

- The whole team along with the management is present and participates.
- The review may include the clients, the sales team, the marketing team, and others
- The review covers the functional, executable systems, which incorporate the objects associated to the respective team, and includes the changes made to implement the journal entries.
- The way in which the journal entries are implemented can be changed during these revisions
- The moment of the next revision is determined by the complexity and the registered progress. The SPRINTS usually have a duration of 30 days.

Critical analysis of the SCRUM methodology

Strengths of SCRUM

The traditional development methodologies are designed only to answer the unpredictable coming from the internal environment and from the development environment, at the beginning of an improvement cycle. The recent approaches, such as Boehm's spiral methodology (Boehm, Anchoring the Software Process, 1996) and its variations are still limited in their ability to answer to changes in requests once the project has been started.

The SCRUM methodology, on the other hand, is designed to be flexible all along the project life. It provides control mechanisms for planning a release, and then for managing the project's variables as it progresses. This allows organizations to modify the project and its deliverables at any time, delivering therefore the most appropriate release.

The SCRUM methodology frees the developers, so that these can focus on developing innovative solutions during the project, when the learning curve and changes in the environment are already taken into consideration.

Small developer teams can share tacit knowledge referring the development processes; also, the methodology provides a very good learning environment for all involved.

The object oriented technologies provide the basis for the SCRUM methodology. The objects, or the product's characteristics, offer a relatively discrete and easy to manage environment. Procedural code is usually not well-fitted for SCRUM, due to numerous and complex interfaces. SCRUM can be selectively applied to procedural systems, only when the interfaces between the various components of the software are relatively simple, and the product has a strong data orientation.

Team size in SCRUM

When SCRUM is the methodology used in the development process, one of the first aspects which needs to be decided is the size of the development team. The recommended size of a development team when using SCRUM is from one to 10 persons, and it is important that the size of the team does not exceed this number. This is particularly relevant because small teams tend to work more independent and more efficient than the teams involved in large projects (Rising & Janoff, 2000). In smaller projects, communication is usually easier, those involved being more up-to-date with the project's evolution. Also, organizing meetings with the client is easier, and the efficiency of those meetings is increased, because having a small number of members in the project team, each can interact with the client, as opposed to the situations in which the number of members in the project team is large, and they practically cannot interact with the client during the meetings.

However, usually it is not recommended that all the project team participates to the client meetings. Many times, some members of the project team are not directly interested by the information emerging from this interaction, and as a consequence they can be informed by their colleagues about the outcomes of the meeting.

When the project team is too large (larger than 10 persons) the difficulties in communicating and implementing changes are increasing, and sometimes these can overcome the benefits of using the SCRUM methodology (Nichols & Twidale, 2003). The disadvantage which can occur by using SCRUM is given by the following situation: sometimes, in order to reduce the size of the project team, the project is split on separate components, each of those being a SCRUM project (Mountain Goat Software, 2007). The coordination efforts in this case are very big, and also the communication difficulties between the managers involved (and in this situation they are at large numbers).

When using SCRUM, traditional roles in software development projects (programmer, designer, tester, and architect) are not used anymore. In this situation, all the members of the project team are involved in finding the solution to the problem at hand. By being involved, all the members of the project team consider that they have a contribution to each part of the project, and consequently a responsibility towards fulfilling client's demands (Mountain Goat Software, 2007).

Another potential problem which can occur when using SCRUM is, however, due to the lack of specialization between the team members. It is commonly observed that a programmer usually writes better code than a solution architect or a designer, and vice-versa, a solution architect will always have a broader point of view over the system than a programmer. The idea of a strong involvement is essentially a good one, though, and the small number of members in the SCRUM team fully contributes to this.

Daily SCRUM meetings

There are three questions that are addressed to the team members at the beginning of the daily project meeting. These questions are:

- What did you do yesterday?
- What will you do today?
- What obstacles do you anticipate?

By addressing these questions, the most important aspects of development will be covered each day. When these daily meeting exist, the team develops very well from the standpoint of human and professional relationships, and the involvement in the project is growing (Rising & Janoff, 2000). Also, the team members will see the progress of their colleagues, and will be more motivated to involve more and to do better their part of work for the project. There is however a possibility that the team members who repeatedly fail to fulfill their daily objectives to become less motivated, or to be exposed to a higher level of stress than the rest of the team.

Rising (Rising & Janoff, 2000), considers that sharing the problems with the team members will motivate the whole team to use the talent and creativity to solve the problems which occurred. These problems do not have to be solved during the meeting, as this will be time consuming. Those who can help solving the problem only have to state their availability during the meeting, and the problem will be solved later.

When there are daily meetings, the visibility over the progress of the project is highly improved. However, it is very important that the meetings' length to be maintained at a minimum, so that no time is wasted. The literature on the subject (Mountain Goat Software, 2007) recommend that the meetings do not exceed 15 minutes. If the meetings last too much, the tendency of the team members is to become un-interested, which can only damage the project instead of helping it.

Another positive fact related to the daily meetings is the fact that the tasks which will be done during a day are known by all the team members, and those who will have to bring their contribution to the respective tasks are warned since the beginning of the day.

SPRINTS

A typical SPRINT in the SCRUM projects lasts 1 to 4 weeks. After each SPRINT, the deliverable is presented, either internally or to the client.

However, experts recommend that a SPRINT should last approximately 30 days in SCRUM. Thirty days are enough for the whole team to understand various parts of the increment, such as design, development and testing. If the SPRINT is done in less than 30 days, it could become difficult that all the tasks to be accomplished. Considering the fact that SCRUM uses the product environment in each SPRINT, if the

SPRINTS are long, the technology can become obsolete, or the environment will change and the product will not fulfill its initial purpose (Control Chaos, 2007).

Another advantage of using SCRUM and the SPRINTS is the fact that once a SPRINT has been decided and started, there should be no more changes over the features which will be developed over the respective SPRINT. The team can have a meeting before the SPRINT starts, and there they can decide what are the activities that have to be done for a 30 days period (or for the length of the next SPRINT). By holding this meeting, a higher productivity from the team members is obtained, and re-work due to changes practically disappears.

There is, however, a trade-off which has to be done in certain situations. Many times, when the client cannot introduce any changes during a SPRINT, will do the change over the next SPRINT, case in which a large part of the work might be re-done. In this particular situation, altering an already started SPRINT is allowed.

SPRINTS are positive features of the SCRUM, as after each SPRINT the decision to continue or to stop the project is possible; this can be decided over some factors such as the market, the client's need for the product, etc. In a contract project, the client has an early chance to experiment the product, and therefore does not have to pay for something that in final could not fit its needs.

There is also a possibility that the product has to be changed during development, in order to better fit the client's needs. This can be done with SCRUM a lot earlier that with a traditional methodology, where the client has a single delivery of the final product.

The main advantage of the SPRINTS is given by the fact that the cost of change in project is significantly lowered (Highsmith & Cockburn, The Business of Innovation, 2001). By avoiding problems as early as possible in the process, a lot of time and money is saved in the project.

The client

In the SCRUM projects, the client represents a very important development factor. At the end of each SPRINT, the client will receive a deliverable, and will be able to see the incremental growing of the product. The client will also receive feed-back regarding the way the product works.

Bu using SCRUM, a good relationship with the client is usually developed, and his knowledge about the project increases in time. Also, the client gets used to the project and to the (inherent) differences between the way he wishes the product to look and work, and the way it will look and work at the final. It is very important that the client to wish to be capable to participate to all meetings regarding the product.

SCRUM is therefore a methodology developed to be flexible and easy to use, mainly through changes that occur during the development process. SCRUM's characteristics make it very suitable for software development projects, where requirements change rapidly or where they are unclear.

SCRUM's most important characteristic is given by the daily meetings, and the SPRINT review meetings; durint these meetings the team along with the client decides what has to be done and what tasks will be accomplished. The SCRUM methodology is suitable to create multiple types of software, like in-house, under contract or for selling on a market, as long as the client has enough time to give consistent and timely feed-back.

Weaknesses of SCRUM

One potential weaknesses of SCRUM, highlighted in the literature (Highsmith & Cockburn, The Business of Innovation, 2001) is the fact that, when the project is developed for an external client, this has to be involved a lot in the project. The client has to be able and available to test the monthly (or periodical) releases or deliverables, and to suggest new or modified functionalities.

In the projects using SCRUM, the vision of the client highly influences development. Highsmith (Highsmith & Cockburn, The Business of Innovation, 2001) shows that if the client does not have a clear sense of the product's direction, the members of the development team will tend to behave in the same way, and the final product can be significantly different to what is expected. Therefore, one of the main weaknesses of SCRUM is precisely one of its strengths: client involvement in the development process.

Another potential weakness of SCRUM is given by the relatively long period in which the client (internal or external) cannot intervene in the project. Although in principle this is an advantage, there are situations

in software development when the client's intervention has to be done within a SPRINT; if the methodology cannot accommodate these interventions, the risk over the project is significant.

The small size of the project team can also be considered a weakness of SCRUM; the way in which this methodology approaches large projects with large teams is viable, but not very easy to be implemented.

SCRUM has another potential weakness: relatively low visibility over the project outside SPRINTS – in other words, it is very difficult to estimate how long a project will take or how much it will cost; in the project with external clients, where bidding is used to determine the contractor for projects, this can be a major hindrance in using SCRUM.

Conclusions

SCRUM represents a software development methodology, based on the iterative methodologies, part of the AGILE category. SCRUM is known by its ability to accommodate frequent changes (technological and from a requirements standpoint) which occur in the software development projects, and also by offering the possibility to quickly obtain results in projects. Its main feature is flexibility, and in the same time it offers mechanisms of controlling and improving the project's performance. Also, the SCRUM methodology, in the current context of the knowledge-based society, contributes to keeping tacit knowledge inside the organizations.

The methodology is very well fit for projects with small development teams (under 10 developers, but 4-5 developers is recommended). It is preferable that this methodology to be used for projects with internal client, or with highly available external client; the availability of the client directly influences the success of the project.

The methodology is used more and more in software development companies; recently it large software providers (such as Microsoft) started to support it, which shows the increasing importance as well as the recognition of this methodology's advantages.

Future research

One of the most important future research directions based on this paper is the identification of ways to overcome the few weaknesses of SCRUM. More specifically, it would be appropriate that variations of this methodology are developed, in order to allow it to be used in larger projects, with larger teams, and which would offer better and scientifically sound planning tools.

Bibliography

1. Aberdeen Group. (1995). Upgrading to ISV Metholdogy for Enterprise Application Development. Product Viewpoint , 8-17.
2. Boehm, B. (1996). Anchoring the Software Process. IEEE Software , 73-82.
3. Booch, G. (1995). Object Solutions: Managing the Object-Oriented Project. Addison-Wesley.
4. Charette, R. N. (2005, 09). Why Software Fails. Preluat pe 3 14, 2007, de pe IEEE Spectrum: http://www.spectrum.ieee.org/sep05/1685
5. Control Chaos. (2007). SCRUM Principle. Preluat pe 06 18, 2007, de pe Control Chaos: http://www.controlchaos.com/old-site/Case6.htm
6. Highsmith, J., & Cockburn, A. (2001). The Business of Innovation. IEEE Computer , 120-122.
7. Jones, C. (1995, 3). Patterns of large software systems: failure and success. Computer , pg. 86-87.
8. Mountain Goat Software. (2007). The SCRUM Team. Preluat pe 06 18, 2007, de pe Mountain Goat Software: http://www.mountaingoatsoftware.com/SCRUM/SCRUMte
9. Nichols, D. M., & Twidale, M. B. (2003). The Usability of Open-Source Software. First Monday , 1 (8).
10. Pittman, M. (1993). Lessons Learned in Managing Obiect-Oriented Development. IEEE Software , 45-53.
11. Rising, L., & Janoff, N. S. (2000). The SCRUM software development process for small teams. IEEE Software , 26-32.

12. Rising, L., & Janoff, N. S. (2000). The SCRUM Software Development Process for Small Teams. IEEE Software , 26-32.
13. Schwaber, K. (1996, 04 05). SCRUM Development Process. Preluat pe 06 14, 2007, de pe Object Technology: http://jeffsutherland.com/oopsla/schwapub.pdf
14. Takeuchi, H., & Nonaka, I. (1986, Ianuarie). The New Product Development Game. Harvard Business Review .
15. Voas, J. M., & Whittaker, J. A. (2002). 50 years of software: key principles for quality. IT Professional , 28-35.