Software Development Methodology for Fast Changing Environment

Zuzana Šochová
CertiCon a.s., Prague, Czech Republic
sochova@certicon.cz
http://certicon.cz

Abstract. This paper presents the usage of the agile and Scrum methods for SW projects in extremely dynamic environments. It describes the benefits of using Scrum process in fast changing environments and addressing the most crucial problems described by IT project surveys. The paper is focused on extremely short projects and delivery times counted on days and hours and changing the Scrum process accordingly. The key focus practices include team collaboration, team responsibility for the quality delivery on time and with right functionality, and customer integration into the process.

Keywords. agile methodology, Scrum process, software development, team performance, iterative development, efficiency, dynamic environment, team collaboration

Introduction

Statistical analyses of IT project failures provide a good overview of the rate of success of IT projects. There are many surveys querying all different environments from different perspectives, but the output is quite poor everywhere: The Robbins-Gioia Survey (2001) identified 51% of the ERP implementation as unsuccessful, the Conference Board Survey (2001) identified that 40% of the projects failed to achieve their business case, the KPMG Canada Survey (1997) identified over 61% of the projects that were analyzed were deemed to have failed [6]. This suggests the conclusion that an IT project is more likely to be unsuccessful than successful [6].

According to the Chaos Report from The Standish Group from 2006, 35% of software projects can be categorized as successful, meaning they were completed on time, on budget and met user requirements [3]. This is indeed not great result, but with respect to the first report from 1994 which labeled only 16.2% of projects as successful [3], there is a substantial improvement.

This progress can be attributed to the acceptance of new methodologies that specifically provide better project management, customer focused orientation and iterative development [2], [10], [3]. The chairman of The Standish Group Jim Johnson said with respect to the extensively better results announces in the 2006 report that there is better project management expertise and technique. Managers have a better
understanding of the dynamics of a project and iterative development, which makes it easier for people to get what they want [3]. One of the methods specifically addressing the issues are agile methods. The key benefit is that agile methods are originally designed for software development, so there is no need to adjust general methodology into a specific ICT environment.

I. Agile Methods

Agile methods have been in practice for some time, but as in any other industry; it takes time to make agile part of life and everyday work. The core document Agile Manifesto\(^2\) actually addresses the most painful issues of the software development projects as were stated in the Chaos Report from The Standish Group.

The 2007 Agile adoption survey shows that agile techniques have been successfully adopted within a majority of organizations [10]. Agile methods promote a true team atmosphere with close daily cooperation between business people and developers, continuous attention to technical excellence and good design, and regular adaptation to changing circumstances [1]. The growing popularity of agile methodologies can be gauged from a recent survey of software developers, where 69% of respondents indicated that their organizations are using agile methods and another 7.3% hinted that they should be going agile in the next year [2],[10].

What is different about agile methods in comparison with other methodologies? Agile is built upon several basic principles, which are in fact common sense for both developers and project managers, but on the other hand hard to keep and follow as well. To be truly agile you have to learn to sense and respond, emphasize improvement and innovation and distribute and coordinate authority [5]. Being agile means following the best practices, listening to the customers, improving existing capabilities, and providing individuals, groups and units the autonomy to improvise and act on local knowledge [5].

On the negative side, agile can be misused for disorderly development with no rules and limited processes. As an example, people might say "We don't have to write documentation; we're doing agile"[4], but that’s not the core idea of agile. Agile methods are less rigid and less tight in a processed way, but still, there is a clear order and processes to follow. There could be even more documentation required, but this documentation and design creation is distributed over the whole project life cycle. Agile could be described as a contradictory process to Structured Analysis i.e. Waterfall method, and is more likely fit to the dynamically changing requirements, environment and customer needs. To some extent agile actually could be described as a huge number of tiny waterfall processes.

\(^2\) Manifesto for Agile Software Development (http://agilemanifesto.org/)
1.1. Efficiency of Agile Processes

So are agile methods more efficient? And if so, why? Agile methods bring several new approaches to software development. Agile addresses practically all the identified software issues – poor estimates and unpredictable due dates bring higher development cost, low code quality, different functionality than expected and thin customer connection causing the customer dissatisfaction. But all of the good ideas in the original vision could possibly be misused and implemented in a wrong way. All new methods and changes needs time to be accepted and cost considerable energy and money as well to set it in a right way. Roughly speaking if a company has switched to agile programming and they haven’t quadrupled their developer productivity in six months, they aren’t doing it right. After two years, they should be five to 10 times more productive than before [4]. Being agile you should “float like a butterfly, sting like a bee” [12].

1.2. Agile Culture

Using agile is not just an internal process within the software development group. Agility must be an integral part of the whole company. The companies must actively maintain a culture of agility [12]. This aspect starting to gain more and more of value - as a recent McKinsey & Company survey found, nine out of ten executives ranked organizational agility as both critical to business success and growing in importance over time [12]. Company agility and ability to adapt faster than competitors must be carefully designed into all processes and culture; otherwise the possibility to succeed in agile software development is limited.

Agile culture is based on iteration process; starting with some process, continuing with reflection, and then implementation of the change to the original design process. It needs active and creative people, who are willing to spend energy improving the way of doing things. Nonetheless, agile methods are not just a formal set of rules you have to follow. It is a set of recommendations and best practices you are advised to use and adjust to your environment. You are not forced to follow all of them, there is no need to deploy agile methods as such, but rather share some of agile practices and values [9].

The principles of agile came from lean manufacturing, which started in the 1960s [4] and can be found in many foreign companies. There is a good quote from the chairman of Toyota. Someone asked him, "What is Toyota's advantage?" He replied, "It is our ability to get normal people to do outstanding things. Other companies try to hire outstanding people to do normal things." [4]. A similar statement would be true if agile and non-agile companies are compared. Agile teams create environments that allow them to be more efficient and provide better quality, just by being agile.

1.3. eXtreme Programming (XP)

One of the first and most commonly known agile methodologies is eXtreme Programming (XP) [8]. It combines all the techniques and practices which should help the developers to create the reliable and high-quality results.

Even though the results from using XP are in general positive, there is still something missing. XP implementation is usually too strict in using some of the methods such as pair programming and test driven development, etc. Some of these ideas may be useful, but it they don’t bring the most important aspect - the team
collaboration and active customer cooperation. XP is, in this context, too inside oriented and doesn’t really reply to the fast changing outside world.

1.4. Scrum Process

A slightly newer agile method is called Scrum. It’s not as widely used as XP, but it is beginning to be more and more common. In contrast to XP, Scrum brings an agile approach for managing software projects by increasing the probability of successful development of software [7]. In general, Scrum is an agile process characterized by self-directed teams, and projects broken into small chunks [4]. Scrum takes the best XP practices and appends the team responsibility, collaboration and brings the customer into the process. Scrum is actually bringing psychology into the software development process.

It’s all about an iteration cycle. The team works in sprints, defined time iterations, in which all the work is first planned and then finished. At the end of sprint iteration, there is a results review. Every other sprint there is a reflection meeting to discuss the process, new ideas, improvements or issues. The length of the iteration is variable, the Scrum process is pretty adaptable to fast changing environments.

1.4.1. Scrum Iterations

When starting Scrum, one of the first questions would be how long the iteration should be. The general rule is, it must be long enough to enable the completion of planned tasks and short enough to prevent changing of the already planned sprint backlog and priorities, which is often not enough. The best agile practice would be to start with any iteration and then change it according to the reflection meeting results. The figure (Figure 1) shows the survey results [10] asking about average iteration length on agile projects. It’s not surprising the majority of agile teams do iterations in between of one and four weeks.

![Average Iteration Length on Agile Projects](image.png)

**Figure 2.** Average Iteration Length on Agile Projects [10]

This raises the question of whether it ever makes sense to make the iteration shorter than a week or longer than a month. Regarding the Scrum methodology, there should be no restriction given from the outside. All the process should be agreed on by the team and it’s again the team who is taking the responsibility for delivering right product on time, with required functionality and within the budget.
If the project has difficult and complex features, such that the functionality cannot be delivered even in one month, then the sprint must be longer. But it must be decided by the team itself. The team must consider whether the environment is really stable enough that they can handle such a long sprint without any serious negative consequences. On the other hand, if there are some special circumstances that call for a sprint iteration shorter than a week then the team should be allowed to set it that way.

1.5. Using Tools

As it was already mentioned, the agile methods are primarily designed for the software industry, with software engineers as the most common users of the Scrum process framework. Software engineers are designing the process, defining the methods, and selecting tools, (which may be, at the beginning, quite difficult).

When starting agile, it is always better to setup an easy environment with almost no supporting tools. If some tools are needed, there will be plenty of time to select them carefully as soon as the process itself is well designed and proved by a couple of iterations. In general, tools are dangerous for teams new to Scrum, likewise to teach math to a 10-year-old, you give him a piece of paper, not a calculator [4].

The overwhelming majority of people starting agile or Scrum, begin with selecting extensive professional tools which will restrict their ability to adapt the process according to the project, customer, and external environment needs. Therefore, Scrum gurus recommend using just cork board and paper cards. They’re not online tools, indeed, they don’t support any automation but they are actually great to understand the Scrum process and adjust it accordingly to the environment and customer specifics without any limitations.

2. Agile and Scrum Methods in Special Conditions – Lot of Idea (LOI) Case Study

2.1. Extreme Short Sprint Iteration

As most of the Scrum process surveys said, more than 93% projects sprint iteration cycles are longer than one week (Figure 2). Actually there is no widely available study describing the possibility of handling sprint iterations of one day or shorter. Most of the agile community would say it’s not possible and it’s going against the methodology.

Such a special setup needs a very good reason to do it that way. But as it was already mentioned, agile methods are just an open set of recommendations and best practices which have to be adjusted to the project specific environment. For example, if the project lifetime could be counted in days, it’s still quite a good idea to use Scrum process to get more predictable results, improving traceability and team collaboration and focus on customer satisfaction. Looking at it all around, it’s even more important than on regular length projects where some small inefficiencies and delays are just hidden in the scope.

2.2. LOI - Business Introduction

One of the very special Scrum processes was setup as a part of consultancy services in Lot of Idea Company (LOI). The process was based on experiences gained within
CertiCon and adjusted to the specific environment of a small company, considering the extremely short product delivery. The key business of LOI is based on two standardized products – web portal and e-shop. The products are designed for every customer, but using standard components as building blocks in maximized scale. Therefore their average delivery time is about 3-5 days.

Considering the extremely short delivery, there is no space to hide any inefficiency or delay. Therefore there was a need to identify strict measurements which would preserve extremely high flexibility and predict the problems in the very beginning as well. It was necessary to make the team really feels as a team; helping and supporting each other as problems arise. In general it was necessary to turn around the whole company culture, to be able to keep all dates and deliver the high quality products.

2.3. LOI – Scrum Process Specifics

Therefore the Scrum process was adjusted to cover the specific conditions. The Sprint cycle was set to half a day. The key reason was driven by the need of extremely good predictability. It was discovered that the average task can be done within 2-4 hours, so the half day sprint makes sense from this point of view.

From the other side, there was extremely high pressure to deliver on time. This was actually nothing new, except there was not much time to think about being late. A clear metric, identifying schedule slip early enough in the project, was a must. So the half a day Sprint would make the project done in about 6-10 iterations, which already provides some time to identify problems and delays and still be able to manage it.

2.4. LOI – Customer Demo

The customer demo in such a short project doesn’t make much sense after every Sprint iteration, but there are still phases where you may like to show the customer your actual progress (i.e. once the first version is ready on the testing server, the customer may like to see it, even if there are just testing texts). In such arrangements there is no need to set up a formal demo, just send an actual online test version every other Sprint.

Nevertheless the customer should be involved in the process. Some projects may require direct customer confirmation. For example, the graphics phase must be confirmed by the customer before the team can continue with the other implementation. This can result in problems with parallel projects planning, where the environment is so dynamic and changing so fast that it’s almost impossible to keep the week plan unchanged.

2.5. LOI – Planning

The planning process has two phases. The first is based on Critical Chain and covering planning of the projects or the project phases, providing sustainable schedule, delivering the output on time. The biggest identified issue was related to the extremely dynamic environment, where the delivery times are changing fast based on the on-time customer confirmation or input data delivery.

The second phase of planning is covering the particular project backlog (Figure 4). Every project has a planned velocity, and from the burndown it’s easy to see if the backlog was extended or not due to changed requirements or new tasks. Accordingly,
the overall project plan and sprint plan is reconsidered every sprint as a follow up of the current sprint results.

2.6. LOI – Backlog and Burndown

Going one by one over the Scrum practices, the product backlog was defined strictly on a point basis. There was no time remaining on tasks, the only pertinent parameters were effort (points) and velocity. As the environment is extremely volatile, and the whole team was not co-located, the best idea for tracking progress was online, using GoogleDocs services. The product backlog for every project was shared online with the whole team, so everybody could track the progress including the project manager, external process consultants and any other shareholders.

In addition to that, the project backlog was extended for the sprint progress tracking table (Figure 3) where the team updated sprint results every sprint. Having that, project managers could directly see the progress and compare it every half a day with planned effort. In the case there was some delay or new work package, there was still some time to adjust the project priority and find some backup to enhance the progress. If that was not possible, or the customer desired extensive changes a new discussion can be opened with the customer about the due date or delivery.

With the product burndown online at GoogleDocs, it was necessary to change the graphs to be compatible with the GoogleDocs. Finally, the most efficient set of the graphs is defined on the following figure (Figure 4). The idea was to create a dashboard including all information project managers and consultants may need for the decision making process, including velocity graph, burndown graph and final date projection.

If the actual line (blue) and the projection (red) is in between of the unbuffered and buffered lines (orange and blue), the project is going well and need no attention. If it goes out of the red line, either the effort must be replanned (i.e added some additional resources to increase velocity), or the discussions with the customer about new delivery must be started.
Finally there is one quite useful graph based on backlog and burndown data. It covers the plan for particular teams and tracks if the reality matched the weekly plan or not (Figure 5). The left graph is checking the team velocity and the right graph is simply showing whether the team had been working on the planned projects or not.4

4The graph shows that during the second sprint, there was additional work done on the ‘purple’ project, and at the end of the week the priority had changed, and the team started to work on ‘blue’ project instead of ‘red’ project.
To summarize the experiences of using the Scrum process in extremely short timeframes, there were significant improvements observed. Before the Scrum methodology was used about 80% of projects were late, without the team really noticing. The new delivery dates were therefore not communicated to the customer on time. Now, about 20% of projects are delivered later than was expected, but in 70% of the cases it’s caused by late customer requirements or confirmation, so such projects are re-planned together with customer and don’t affect customer satisfaction.

In addition to that, setting the weekly plan helped a lot with the overall workload distribution. At the beginning, there was not even single day according to the plan, which is now significantly better, and the changes are correct with respect to the highly dynamic and flexible environment.

It was discovered that until there was a possibility to perfectly track the history and to visualize it, there were just a huge number of exceptions with proper reasons for doing them, but no one really noticed there are just the exceptions. The first visualization was extremely important, as it brought all these small exceptions together and it was clear that there is almost nothing as it was originally planned.

To summarize it, after 5 months of usage, Scrum was evaluated as very successful in the LOI environment. There is still some space for improvements, but the steps already taken, provided higher quality results and significant progress.

3. Conclusion

Considering the mentioned surveys, agile methods are likely to be successful. One of the key aspects is, agile is not just a method used for software development, but it must be used within the whole company and it must be an integral part of the culture and IT governance. The most valuable attribute of agile methods is the flexibility and adjustability. So don’t be afraid to adjust it to a particular business, customer, or environment specifics. Effective IT governance improves your ability to deliver software that meets your stakeholders’ needs; without it, your team may go rogue and ignore management’s directives [11].

The case study described above approved, that agile and Scrum methods address the project success criteria [13], even under specific environments and conditions. Although the Scrum process is originally designed for longer projects (weeks), it was confirmed in the case study, that Scrum process brings benefits even when the delivery times are counted on days and hours, which is not generally acknowledged.

As a final point, the following table (Table 1) maps the project success criteria as defined at [13] to the agile and Scrum practices as it was observed on many different projects using Scrum.

<table>
<thead>
<tr>
<th>Project Success Factors</th>
<th>Scrum Process</th>
</tr>
</thead>
<tbody>
<tr>
<td>User Involvement</td>
<td>The customer sees results every increment the result and he is often brought back into the process deciding the priorities or confirming the functionality, his involvement increased.</td>
</tr>
<tr>
<td>Executive Management Support</td>
<td>Someone from the management is part of pre-planning meeting and customer demo.</td>
</tr>
<tr>
<td>Clear Statement of Requirements</td>
<td>At all times it is clear what we are going to do. It doesn’t mean the</td>
</tr>
</tbody>
</table>
requirements are fixed over the project, but they are clear at any time.

<table>
<thead>
<tr>
<th>Requirements</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proper Planning</td>
<td>Using Scrum, the overall plan is checked and adjusted every sprint.</td>
</tr>
<tr>
<td>Realistic Expectations</td>
<td>As the progress is checked every Sprint and the results are demonstrated every increment, then everybody including internal or external customers has better understanding of what’s going to be developed.</td>
</tr>
<tr>
<td>Smaller Project Milestones</td>
<td>As the project milestones (sprints, increments) are smaller, the reflection can be done more often, providing traceability and more realistic expectations and bringing customer satisfaction in the end.</td>
</tr>
<tr>
<td>Competent Staff</td>
<td>People are forced to work as a team, helping each other and learning from each other, so the skills and efficiency grow faster.</td>
</tr>
<tr>
<td>Ownership</td>
<td>As the Scrum methods are based on the team work, there is no particular developer ownership. Every code or document is shared and there are several people ready to update it. This goes aligned with continuous integration.</td>
</tr>
<tr>
<td>Clear Vision &amp; Objectives</td>
<td>Using Backlog and Burndown, everybody has a clear vision of where we are going and at which phase we are right now. The customer demo brings the user voice and reflection meetings bringing the team into the decision making process.</td>
</tr>
<tr>
<td>Hard-Working, Focused Staff</td>
<td>Team is responsible for the result and team is making the promise to deliver the planned tasks. Every developer is updating the burndown with his progress, so every developer is part of the project and can influence it during reflection meetings, which brings higher motivation.</td>
</tr>
</tbody>
</table>

**References**


