Team leading challenges in software development projects

Filip Jovanovic
Project Management Department
Faculty of Project and
Innovation Management
Belgrade, Serbia
filipj85@gmail.com

***Abstract*—Scrum and agile approach in software development eases the way teams are handled and managed. With the right application of this approach into software development projects, Scrum can influence team cohesion, improved efficiency and better control of the final product. Thus, team management in the phase of integration of Scrum values and modifying its "rules" is very important and can have a massive implication on results. Although all standard approaches in team management are important, key for success with Scrum is a personal relationship between Team Leader and the team itself. Maintaining and building human relations and positive atmosphere between team members can help overcome difficult stages in the project. Right motivation and sense of togetherness can, in some circumstances compensate for mistakes in planning or execution. Key in building this circle is Team leader, who needs to guide team through conflicts and team evolution.**

Keywords—project, software, team leading, project management, software development

I. INTRODUCTION

When conducting large or small software development projects, the right team with the right leader makes the most significant difference in achieving the planned deadlines and the quality of the product itself. [1]

Small, quickly organized teams of 5-6 full-stack developers can surpass up to 10 times bigger teams and produce applications and software improvements in a more efficient way. an adding more staff and more developers do not necessarily lead to better results. It's the efficiency that matter and good team cohesion that creates value. Exactly these results and numerous analysis during the years motivated almost all world companies and startups to apply the agile way of guiding and organizing programming teams. For example, in Scrum methods of agile organization, the emphasis is on the division of programmers into lesser subteams, each of which works on a special functionality or software option and is responsible for this functionality and absolutely independent with other teams. Company's architecture team and top management with the CTO oversee all team's work and makes cohesion between them.

Continuous communication between Team Product Manager and top management is critical to the success of this approach. Without the right communication and collaboration within the team and outside of it, the project is doomed to failure, as each information in all types of projects is crucial.

Along with these, one of the main benefits of Scrum is slow and precise iteration and constant change of the product - as opposed to standard waterfall development, where the product is seen only at the end, whereas any change is quite difficult. Also, with its "cutting" of the work in small tasks and correct measuring of the scrum burnout rate, Scrum prevents the overload of the funnel, as each team works

In leading these small agile teams, leadership and capabilities of Team Managers, or Product Managers in software projects, can greatly influence the result of the project itself. A product manager needs to take the team very carefully through all the phases in its development while keeping it on track with project deadlines and desired product quality. The right approach can maximize results, while the wrong approach with even the best of teams can lead to missing deadlines, failing to meet stakeholder expectations and low product quality - which can later have enormous consequences in the app/software market.

Phases in team evolution must be guided with precision and team must be allowed to go through them in order to evolve.

As for the conflicts, in Scrum approach, since changes are often and approach is flexible, team and team leaders must be ready and prepared to constantly manage conflicts and use them to get better results, as they occur much more because of constant changes.

In this article, we will analyze the strategies and impact of team leadership and phases in team development in a larger software development project.

II. PROJECT

A. Project - setting and goals

The project itself was done over an 11-month course, inside a gaming software development team in Mozzartbet company - the biggest betting company in Southeastern Europe. The team consisted of 4 full-stack developers and one Product Manager, who acted as a team leader. Company has decided to build a big software solution internally and given a quite heavy task to a small team, which also had other obligations within the company. Both team members and team leader needed to overcome a lot of personal, operational and technical hurdles in order to complete the project

The project had several goals, but one of them was primary - to create and put in motion a complex Casino

Management System internally and then showcase it at the London ICE Exhibition.

The secondary goal was to unite all the casino software subsystems existing in the company at the time but working independently.

The team wanted to unite all the pieces and allow stakeholders and commercial departments to fully control the casino network from one single place, and with one overall software solution, without the need to manage many interfaces and application.

Beforehand, the team assessed the overall situation with casino software in the company and found that the quality did not match the other departments' solutions and expressed a desire to improve the status. The current system was 8 years old and was built for 300 slot machines that company had at the time. Since number was now reaching close to 4000, the existing system was reaching breaking points in a lot of cases. Additionally, end-users were forced to use 4 interfaces just to create a jackpot in the system. This created massive delays and human errors, which consequently created financial losses for the company.

At first, a visit to yearly London ICE Exhibition was made with an aim to buy an existing Casino Management solution that would give all the benefits of the current system and add the modern options of more advanced European systems. However, during analysis, two problems occurred:

- Systems were far too expensive
- They did not offer a unified dashboard or unified solutions to control machines in many locations in one place.

As the section 2 was quite important from company's standpoint and having in mind that 3000 machines were spread in more than 500 locations, acquisition of the software was abandoned - no viable solution was found, and the team went back to point zero.

B. Project - task/key

After careful assessment and convergent with existing and future company goals, top management decided that a new software system will be created internally, from scratch, called Mozzart CMS. Developers agreed that they would write totally new code following modern standards and that it would be done until the next exhibition in London, which was due in 11 months. The system would answer all of the business needs, change all existing internal slot software solutions, unite them and include:

- Accounting for the machines
- Security overview
- Jackpot control
- Remote machine settings
- Reporting for management

After the team was given the task, they created a development plan that aimed at 10 months for the completion of the software with all the features and full go-live date for one region. Team also obliged to have a fully presentable version of the software, with two test machines in London ICE fair, which could allow potential buyers to acquire the system. The technology used was Java, AngularJX along with Java FX.

PROJECT - MAIN SUBPARTS

Software Projects, in general, have 6 main parts [2]:

- Planning
- Analysis
- Product design
- Development and testing
- Integration with other solutions within the company's software system
- Maintenance.

Planning and analysis

During planning and analysis, the team decided to go with RAD - Rapid Application Development. Rapid application development (RAD) is a software development methodology, which favours iterative development and the rapid construction of prototypes instead of large amounts of up-front planning. The "planning" of software developed using RAD is interleaved with writing the software itself. The lack of extensive pre-planning generally allows the software to be written much faster and makes it easier to change requirements.

The rapid development process starts with the development of preliminary data models and business process models using structured techniques. In the next stage, requirements are verified using prototyping, eventually to refine the data and process models. These stages are repeated iteratively; further, development results in "combined business requirements and technical design statement to be used for constructing new systems". [3]

RAD would give the team the flexibility and would allow fast delivery of each part. Consequently, this approach was the only one that would allow developers to work on their daily tasks other than the CMS project itself.

Of course, these "other" tasks could have created a massive problem during the project, since every developer had daily operational tasks in order to maintain the current system, aside from new features and new development for the CMS. Team and Product Manager foresaw these issues and decided beforehand that the current system should:

- Kept on minimum features and activity
- Receive no new updates (as it was due to be changed soon)
- Create automated tasks that would check and fix bugs that occur daily, with a solid accepted margin of error for these bot-assessed tasks

This decision was made since no options for adding new people were present, and a big focus was needed from the team, in order to complete such a huge task in time. This message was also clearly sent to stakeholders, as to align their expectations and to be aware that the current software will have to "suffer" in a way, in order to have more time to create a new one. As stakeholders approved, the team moved to the product design phase

Product design

In this phase, a product is designed, with all the desired features and a full Software Requirement Specification (SRS) is made. Software Requirements Specification (SRS) is a document or set of documentation that describes the features and behaviour of a system or software application. It includes a variety of elements that attempt to define the intended functionality required by the business (in this case Product Manager and stakeholders) [4]. A database schema is created, and all the features and options are listed along with concrete explanations and use cases for these features. This is the core of the project - a precise plan of what will be developed, how it will behave, how it will look like from a UI/UX perspective, what are the processes and software services needed to support this system and how will software communicate with all the slot machines in the network. A product design document is usually made by the manager or the main stakeholder, who wishes a change in the system. A common practice is that the team is not involved in this as to avoid potential derailing of business wishes. Developers often tend to eliminate some features they deem unimportant, but they are not the ones measuring business impact - that is purely Product Manager's task. In our example, complete documentation was built by Product Manager but with the help of company's CTO who knew the limits of other systems in the network and overall architecture. CTO's role in every development was to assure every new application, service or system will fit into the current architecture smoothly.

Development and testing

During development, the team creates features, backend processes, user interfaces and all the sections provided in the product design plan. Depending on the agile approach, either user testing is made or automated testing. It is usually the hardest phase, and this is where the biggest amount of work and problems are faced. Team evolution and changes impact this phase heavily.

Main phases in team evolution are [5]:

- Forming
- Storming
- Norming
- Performing
- Adjourning

Team has already gone through Forming, Storming and Norming previously, as they knew each other for a few years. But surprisingly, they want back to storming again in this project, as each developer's seniority and experience was called into question. This happened since they have not done a project of this scale never before, nor they faced any kind of deadline of this kind. The pressure and different approaches to product design made them go back to storming phase. The product manager has let the team go through the conflicts and guided them with assertiveness and trust.

Main ways of dealing with the conflict are [6]:

- Forcing
- Win-Win
- Compromising
- Withdrawing
- Smoothing.

The product manager was mostly pushing the team towards compromising and win-win. Key for this was his own position and the goal of the whole project.

The only unifying point for the team at the time was the position and their view on the position of Product Manager. Since Product manager was not a developer, but a business-oriented person with technical and market knowledge, it made the team put aside their differences and accept his own proposal for the product design and all of the changes he made in the development sprints week over week. Since this was not any of the team members - developer's choice, they accepted the proposal with no rejection. This was seen as a Win-Win, as no one from the team itself forced the decision or a choice, which would make other team members oppose this (as this was seen as "against" their own proposal). As a Product Manager had a mediator role, everybody saw this as a perfect solution.

This was also partly because the team had confidence and trust into Product manager's vision for the software that made the team resolve their differences and unite over the idea of building such a big and complex system in such a short time.

An additional reason was also that this decision put all the developers out of any responsibility for the end product and allow them to solely focus on executing tasks. If anything would come on a wrong path, developers knew it was not their own decision or making so they wouldn't be called for.

Types of decision-makers:

The Gut Instinct Follower

The Interviewer

The Exhaustive researcher

The Objective debater

The Random chance submitter

As noted, Product manager in this project can be seen as a Gut decision-maker [7].

Even though the standards for building this kind of software usually goes from 1.5 - 2 years, the team accepted the terms and committed to finishing in less than a year. The idea of presenting the software in front of all the relevant stakeholders in London made the team go over individual differences and do not hold their stances about integration and development. The end goal was bigger than the team itself and was used by the Product Manager as the guiding light in all the phases of the project

This mostly underlines the view on the importance of leadership qualities of the team leader. Although competency might seem influential and crucial at the start, as the project progresses, human relations and trust are the only important things.

Change in leadership approach

Biggest risk and doubt over the success of the project came right towards the end. Even though the team have completed a remarkable development task and managed to build 95% of

the features and software within deadlines, the last couple of steps provided major issues in the project.

The overall team atmosphere and assessment were that the job is done and that this massive task was completed in the desired quality. Internal presentations to top management gave appraisal to the team and confidence that the system will exceed expectations. These appraisals have made the team relax and slow down the pace of feature releases. The last couple of sprints were not efficient, and in the sprint review meetings, problems have arisen. In practice, minor parts were not working as they should, and the interface was full of bugs. This situation was left dragging for few weeks as the team fell into "comfort zone". But even though almost everything was completed, the missing parts could have still ruined the final product.

Product Manager started applying pressure and acted from the point of authority, thinking that it would influence the developer's work in a strict manner. Unfortunately, this tactic had absolute zero success, since the change from assertive to aggressive did not go well, and the team backed out, negatively surprised by this sudden change of approach from the Product Manager. The "laissez-faire" approach that Product Manager has used for 90% of the time, suddenly changed to autocratic, which did not pay dividends.

This proved that even though changes in the leadership tactics and styles is sometimes necessary, big and drastic changes can lead to a negative reaction from the team. Only when the manager backed out to coach/visionary role, team rolled up back again and worked efficiently.

III. CONCLUSION

Software teams in these circumstances tend to be more sensitive than others and react to personal changes. Although competence, motivation and planning have a massive influence on team's cohesion and subsequently to later success, friendly and work environment between the team and product manager can prove to be decisive in building the right product. The team feel more relaxed and less pressure. Since they have a healthy atmosphere, they also do not feel a lot of competition between each other. In every conflict, the sense of trust helps them reach the next stage.

REFERENCES

[1] J. Westland "What is project leadership", https://www.projectmanager.com/blog/what-is-project-leadership, retrieved April 2019.

[2] D. Conde, Dan, "Software Product Management: Managing Software Development from Idea to Product to Marketing to Sales", Aspatore Books, ISBN 1587622025, 2002.

[3] B. W. Tuckman, "Developmental sequence in small groups" Psychological Bulletin, vol 63(6), pp. 384-399, June 1965.

[4] IEEE Guide to Software Requirements Specifications. 1984. doi:10.1109/IEEESTD.1984.119205

[5] L. J. Whitten,B. D. Lonnie, K. C., Dittman. "Systems Analysis and Design Methods". 6th edition. ISBN 0-256-19906-X., 2003.

[6] L. Berger "Forbes: Five Conflict Management Strategies", https://www.forbes.com/sites/forbescoachescouncil/2017/06/07/five-conflict-management-strategies, retrieved april 2019.

[7] T. Carter, "The 5 Types of Decision Makers and How Each Can Thrive in Businesshttps://www.business.com/articles/the-5-types-of-decision-makers-and-how-each-can-thrive/, retrieved April 2019.