St. Petersburg University
Master in Management Program

A QUANTITATIVE ANALYSIS OF THE SUCCESS OF AGILE IT PROJECT

Master's Thesis by the 2nd year student
Concentration – MIM
Grigorii Liubachev

Research advisor:
Senior Lecturer, Elvira V. Strakhovich

Saint-Petersburg
2017

ЗАЯВЛЕНИЕ О САМОСТОЯТЕЛЬНОМ ХАРАКТЕРЕ ВЫПОЛНЕНИЯ ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ

Я, Любачев Григорий Михайлович, студент второго курса магистратуры направления «Менеджмент», заявляю, что в моей ВКР на тему «Количественный анализ успешности гибких проектов в сфере информационных технологий», представленной в службу обеспечения программ магистратуры для последующей передачи в государственную аттестационную комиссию для публичной защиты, не содержится элементов плагиата.

Все прямые заимствования из печатных и электронных источников, а также из защищенных ранее выпускных квалификационных работ, кандидатских и докторских диссертаций имеют соответствующие ссылки.

Мне известно содержание п. 9.7.1 Правил обучения по основным образовательным программам высшего и среднего профессионального образования в СПбГУ о том, что «ВКР выполняется индивидуально каждым студентом под руководством назначенного ему научного руководителя», и п. 51 Устава федерального государственного бюджетного образовательного учреждения высшего профессионального образования «Санкт-Петербургский государственный университет» о том, что «студент подлежит отчислению из Санкт-Петербургского университета за представление курсовой или выпускной квалификационной работы, выполненной другим лицом (лицами)».

[illegible] (Подпись студента)

29.05.2017 (Дата)

STATEMENT ABOUT THE INDEPENDENT CHARACTER OF THE MASTER THESIS

I, Liubachev Grigorii Mikhailovich, (second) year master student, program «Management», state that my master thesis on the topic «A Quantitative Analysis of the Success of Agile IT Project», which is presented to the Master Office to be submitted to the Official Defense Committee for the public defense, does not contain any elements of plagiarism.

All direct borrowings from printed and electronic sources, as well as from master theses, PhD and doctorate theses which were defended earlier, have appropriate references.

I am aware that according to paragraph 9.7.1. of Guidelines for instruction in major curriculum programs of higher and secondary professional education at St.Petersburg University «A master thesis must be completed by each of the degree candidates individually under the supervision of his or her advisor», and according to paragraph 51 of Charter of the Federal State Institution of Higher Professional Education Saint-Petersburg State University «a student can be expelled from St. Petersburg University for submitting of the course or graduation qualification work developed by other person (persons)».

[illegible] (Student's signature)

29.05.2017 (Date)

АННОТАЦИЯ

Автор	Григорий Михайлович Любачев
Название магистерской диссертации	Количественный анализ успешности гибких проектов в сфере информационных технологий
Направление подготовки	Менеджмент
Год	2017
Научный Руководитель	Эльвира Витаутасовна Страхович
Описание цели, задач и основных результатов	Целью данного исследования является разработка объединённого, приоритизированного количественного подхода измерения успешности Agile проектов в сфере информационных технологий. Он позволит менеджерам проектов измерять и контролировать процесс разработки на разных этапах жизненного цикла проекта, что приведет к максимизации преимуществ от внедрения Agile. Для достижения цели исследования был применён трехступенчатый подход. Сначала современная литература была проанализирована, чтобы выработать структурированное мнение об успешности проектов и определить аспекты их успешности. На втором этапе было оценено покрытие выявленных аспектов успешности количественными метриками, которые уже используются в организациях. Для этого метрики были собраны и сопоставлены с критериями успешности проекта и факторами, которые на них влияют. Ввиду их недостаточности, дополнительные техники оценки были предложены. В конце мнение экспертов было использовано для приоритизации параметров оценки. В результате был разработан подход для оценки успешности Agile проектов в сфере информационных технологий. Созданные список метрик и чек-лист лежат в его основе и могут быть использованы для целостной оценки проектов. Индустриальная специфика в разработанном подходе имеет второстепенную роль. Это означает, что в будущем он может быть использован как основа для разработки схожих подходов в других индустриях.
Ключевые слова	Гибкие методологии, Успешность, Оценка, Управление проектами, Разработка ПО

ABSTRACT

Master Student's Name	Grigorii Liubachev
Master Thesis Title	A quantitative analysis of the success of agile IT project
Main Field of Study	Management
Year	2017
Academic Advisor's Name	Elvira V. Strakhovich
Description of the goal, tasks, and main results	The goal of this study is to develop a unified, prioritized quantitative approach to measuring the success of Agile IT projects. It would allow project managers to track and control the development process at different stages of the project lifecycle, leading to the maximization of Agile adoption benefits. In order to achieve the goal a three step approach was adopted. Firstly, contemporary literature was analyzed to develop a structured view on the project success and define aspects of their successfulness. Secondly, quantitative metrics, already used by organizations were assigned to each of the success criteria and factors to see if they cover all aspects of project success. Since they didn't, other assessment techniques were proposed. Finally, expert opinion was used to prioritize evaluation parameters, when necessary. As a result, an approach for Agile IT project success evaluation was created. A list of metrics and a checklist are a core of it. They can be used to holistically evaluate Agile IT project success. Industry specifics in the developed approach appeared to play a secondary role. It means, that in the future it can be used as a base for the development of similar approaches for other industries.
Keywords	Agile, IT Project, Success, Evaluation, Project management, Software development

TABLE OF CONTENTS

INTRODUCTION

Nowadays software significantly affects almost all aspects of our lives. Because the applications of software are very diverse, there is no single best way to do software development. New methodologies always emerge, and old get transformed.

In the early days, old “heavy” project managers applied processes that evolved during the era of manufacturing to the IT projects. Later it became apparent, that with such a high pace of technology development and ever-changing requirements of the business environment a different approach should be taken. It wasn’t someone's decision, but rather the natural evolution of the working process. In 2001 seventeen independent practitioners from the so-called «Agile Alliance» officially formulated new ideas and published Agile Manifesto.

Today, agile projects are almost four times more likely to succeed than traditional, sequential Waterfall projects - 39% vs. 11% (The Standish Group, 2015). However, Agile just recently became a dominant approach in IT with 67% of organizations using pure agile or leaning towards agile. A major uptick in Agile adoption began in 2009-10 (HP, 2016), with over half of organizations switching to agile during the last five years.

That might be a reason why a lot of the existing research on the topic focuses on guiding decision-making process of Agile adoption. Hummel M. identified that “Evaluation of Agile Practices”, “Adaption / Combination / Extension”, “Tool Support”, “Team Characteristics” and “Adoption” account for more than 60% of all studies on Agile. (Hummel, M., 2014).

As Agile becomes the new norm, tracking and control of methodology application start to play a vital role. One of the latest “State of Scrum” reports showed that main challenges organizations faced in achieving their goals with scrum were a lack of clearly determined metrics to identify and measure the success of Scrum projects and delivery (The 2015 State of Scrum Report, 2015).

The goal of this thesis is to develop a unified, prioritized quantitative approach to measuring the success of Agile IT projects. It would allow project managers to track and control the development process at different stages of the project lifecycle. It will lead to the maximization of Agile adoption benefits.

As a means to achieve this goal, a three-step process is utilized. Firstly, contemporary literature will be analyzed to develop a structured view on the project success. It will include success criteria and factors affecting these criteria. Secondly, quantitative metrics, already used by organizations will be assigned to each of the success criteria and factors to see if they cover all aspects of project success if not, other assessment techniques will be proposed. Finally, expert opinion will be used to prioritize evaluation parameters, when necessary and conclusions will be made.

As agile is spreading to new industries and geographies, it is just a question of time, when sophisticated tracking and control would be needed outside of information technology sector. This thesis can act as a base for the future research, since parameters are mostly based on the unified criteria of project success, not related to a particular industry, and could be adopted to new use-cases.

The rest of the thesis is organized in the following manner:

In the chapter one Agile project management will be introduced and compared with alternatives. Different Agile methods will be presented. The literature on agile success will be reviewed, and research gap will be identified. Research questions will be formulated.

In the chapter two review of existing literature on the project success criteria and factors will be made. An overall view of project success will be developed. Then metrics used by contemporary organizations will be found and structured. The full list will be checked for duplication and assigned to different success criteria and factors.

In chapter three Fuzzy Analytic Hierarchy Process (AHP) analysis method for incorporation of expert opinion will be introduced. The final questionnaire for the empirical part will be presented. Results of the empirical study will be summarized and discussed.

Conclusion and suggested implementation procedure will follow.

CHAPTER 1: AGILE PROJECT MANAGEMENT AND PROJECT SUCCESS

Before the 1970s, mostly old project management processes that evolved during the era of manufacturing were applied to the IT-projects. An example of such traditional sequential approach to IT project management is Waterfall software development model. It was first formally introduced in 1956 by Herbert D. Benington. In waterfall development process is perceived as a sequence of steps. Transition to the next step can be made only when the previous step is completed (Benington, 1983). This kind of project life cycle model is called predictable model (PMBOK 5th edition).

It works well if all the requirements are presented upfront, but is inflexible (Rahmanian, 2014). With such a high pace of technology development and ever-changing needs of the business environment, it became apparent that in some cases a different approach should be taken. It wasn’t someone's decision, but rather the natural evolution of the working process.

At the same time, raising costs of IT-projects made failures even less tolerable. However, that wasn’t reflected by a decrease in the project failure rate. In contrast, IT-projects were failing distressingly frequently. While there is no precise statistics, due to the fact that accuracy of data in early CHAOS reports is now being challenged (Jørgensen et Moløkken-Østvold, 2006), some sources indicate that at least 50% of IT-projects failed to meet the expectations of stakeholders (Keil et Mark, 1995, p. 442). One trend can be clearly seen when analyzing types of project failures. Projects start to "take on a life of its own" and get terminated after they went significantly over the budget and timeframe. Commonly the reason for that is seen in poor project management.

One of the attempts to distinguish good project management from the poor one was undertaken by Boehm et al., (1989, p. 903), they argue that there are three principles that a proper methodology for managing software projects should simultaneously follow. It should be general, simple and specific.

However, further formalization of IT-project management was seen as a solution. Capability Maturity Model (CMM) and Capability Maturity Model Integration (CMMI) from Software Engineering Institute can be mentioned as examples. The creation of documentation and strict following of the process defined by CMM/CMMI became superior to the actual software development. Developers opposed such formalization and eventually it was one of the reasons behind the development of new approaches built upon new, agile principles.

1.1 Introduction to Agile methodology and methods

Back in 1974 alternative ideas regarding the project management started to be discussed. (Edmonds, 1974). However, only in 2001, a new methodology called Agile was officially formulated. It proposed new ideas, quite different from those in the conventional way of project management (See Table 1). The methodology is based on 12 principles that rely on four primary. values. According to the Agile Manifesto, these values are:

*"**Individuals and interactions** over processes and tools*
***Working software** over comprehensive documentation*
***Customer collaboration** over contract negotiation*
***Responding to change** over following a plan"*

As it can be seen from the principal values, agile methodology is designed to simplify and accelerate the process of delivering working software, by promoting communication, collaboration among all stakeholders and eliminating risks of being unable to respond to the challenges over the course of the project. (Agile Manifesto, 2001)

Table 1 Main differences between traditional development and agile development (Dybå and Dingsøyr, 2008)

	Traditional development	Agile development
Fundamental assumption	Systems are fully specifiable, predictable, and are built through meticulous and	High-quality adaptive software is developed by small teams using the

	Traditional development	Agile development
	extensive planning	principles of continuous design improvement and testing based on rapid feedback and change
Management style	Command and control	Leadership and collaboration
Knowledge management	Explicit	Tacit
Communication	Formal	Informal
Development model	Life-cycle model (waterfall, spiral or some variation)	The evolutionary-delivery model
Desired organizational form/structure	Mechanistic (bureaucratic with high formalization), aimed at large organizations	Organic (flexible and participative encouraging cooperative social action), aimed at small and medium- sized organizations
Quality control	Heavy planning and strict control. Late, heavy testing	Continuous control of requirements, design, and solutions. Continuous testing

A main distinguishable feature of agile is the fact that it is iterative (See Figure 1). The process starts with the creation of the product backlog. Stakeholders collaboratively decide what should and what shouldn't be included in the product. Usually, the product owner who has the vision of what product should deliver so-called "epics". These epics are broad user stories that should be broken down into smaller "stories". These stories constitute the backlog. After stories are prioritized, the backlog for future iteration is created.

Each iteration combines all the processes that are involved in the software development: Planning, actual development, testing and receiving feedback, reacting to feedback and releasing. All processes are interconnected, meaning that the outcome of one process influences the whole loop *(*Huo, Ming, et al., 2004).

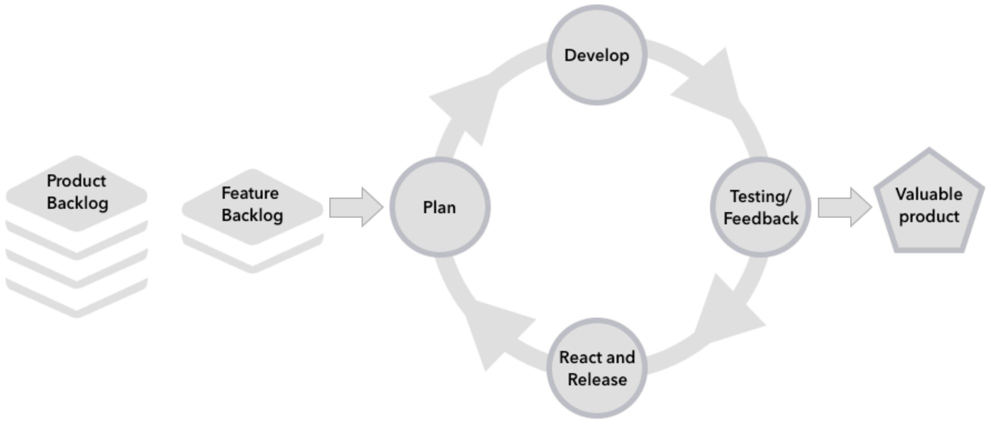


Figure 1 Agile project methodology product lifecycle (Huo, Ming, et al., 2004)

Agility in this methodology should be understood as an ability of an enterprise to react in advance to the changes happening in internal and external environments. This implies that agile organization strives for feedbacks since as they are incorporated into the product, it becomes more valuable for the customers which in turn guarantees their satisfaction. (Kumar and Bhatia, 2012)

The goal of Agile methods is to spend as little time and effort on planning as possible since they were developed in response to plan-dominated methodologies. Planning is not considered evil as such. The procedures required by planning are thought to take too much time, slowing down the development process. That is why software features are broken down into smaller parts, eliminating the long-term planning from being directly involved in the process of features development. This means that agile approach to software development is also adaptive. (Beck, 1999, pp. 70-73)

Since one of the main risks of projects is related to inaccurate estimations and plans, managing software projects in this way promotes adjusting to the changing needs on every phase of the process. As main risks are eliminated, this approach is expected to lead to fewer failures. It is important to mention that according to the agile methodology a result of any single iteration should be ready for release. However, it often needed to do several iterations before a particular product, or a feature adds significant value.

The way how traditional way of project management evolved into the modern methods can be graphically seen in the Figure 2.

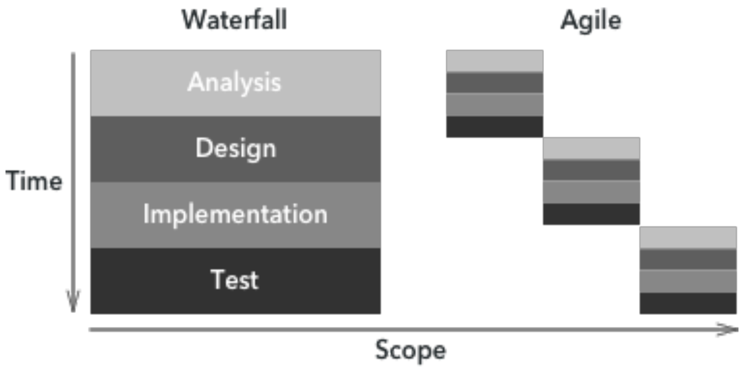


Figure 2 Evolution from Waterfall to Agile (Source: Beck, 1999, p. 71)

For the past two decades, more and more IT companies were implementing Agile. Actually, every company implemented agile a bit differently (Racheva, Daneva et al., 2010, p. 131). As several prevailing ways to implement agile are currently in use, it is important to distinguish between them, in order to be able to analyze their success factors.

Table 2 Summary of key Agile frameworks

Name of the framework	Key features	References
Extreme Programming (XP)	**Simple Design:** Development is started with a simple design in mind. As more features are added, the design is made to fit perfectly the current state of the product. **Pair Programming:** Production code is then written	(Lindstrom and Jeffries, 2004, p. 47-49);

Name of the framework	Key features	References
	by two developers sitting at the same machine. **Test-Driven Development:** developers first write a test and then try to make it work. So any iteration is a combination of smaller test driven cycles. **Design Improvement:** Every iteration can bring value to customers and businesses only when the product is designed well. This is achieved by engaging in quick design sessions and design revisions through refactoring.	
Scrum	**Product owner**: a person who communicates with customers and users and represents their opinion in the form of vision. **The Scrum development team:** a multi-functional team, meaning that there is no distinguishing between developers, testers or designers. It acts as a single self-organized entity. **Scrum Master:** It is a person without direct managerial power, who is responsible for enhancing the communication between the team and product owner and making it easier to apply the Scrum techniques. **Sprint:** An iteration, which usually is about 1-4 weeks. A team working on those iterations are usually small. However, it can easily scale for large teams with hundreds of members.	(Schwaber, K., 1997)
Feature Driven Development (FDD)	FDD can be split into five main activities: 1. the desirable outcome is documented as a model and notes. 2. Full list of features is created 3. The individual plan of feature is created 4. The individual design of model is created 5. The individual feature is developed.	(Beck and Beedle et al. 2001, p. 4-9)
Crystal Method	- Focus on people and not on tools and processes. - A constructor, rather than a group of methods. - The necessary methodology is developed for each project, meaning no extra procedures will take place. Nowadays it is mostly used in non-critical projects of small size, where just a couple of people are involved.	(Cockburn, 2004, p. 4-5)

1.2 Agile IT project success in contemporary literature

Considering the popularity of Agile methodology, it is not surprising that a lot of research was done on the performance of Agile IT projects. However, a majority of research on the topic is based on casual observations. Just recently a major cross-industry international study of 1002 projects was conducted in order to see whether Agile methods can improve the likelihood of project success. It was found that implementing agile methods, compared to doing so with those of other approaches, does have a positive influence on project success. Interestingly, the quality of the vision/goals turned out to be a marginally significant moderator of the effect. (Serrador & Pinto, 2015).

There is a belief that agile methodologies that help small projects to succeed may fail to scale for larger projects or projects with multiple teams (Rosser et al., 2014). A recent empirical study, using a conceptual model (see Figure 3) suggests that project size cannot be used to explain Agile project failure or success. The model was tested on 40 Dutch projects, and it revealed that agile might be suitable for larger projects if there is "high value congruence, agility and transformational leadership" (van Kelle et al., 2015).

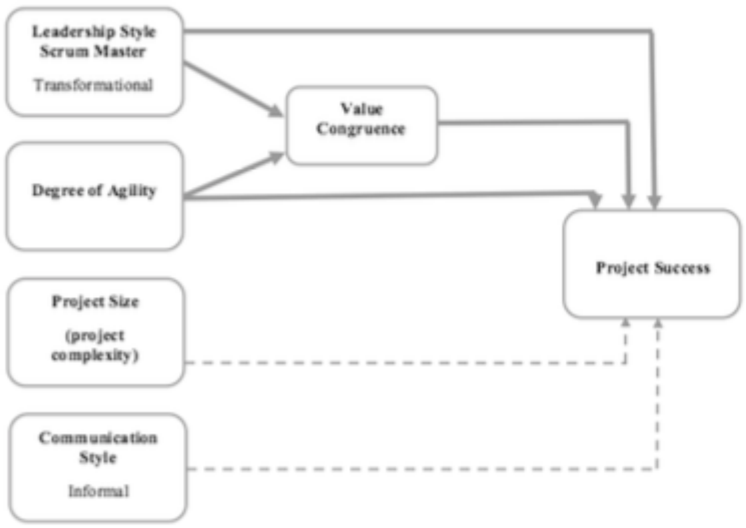


Figure 3 A conceptual model of social factors that may be of influence on the success of software development projects in general, and of Agile projects in particular (van Kelle et al., 2015)

While there is evidence that Agile projects can succeed in different environments, the evaluation of success is often subjective. For example, in work by Serrador and Turner introduced earlier, authors used "a self-reporting (subjective) assessment of project success and use of Agile methods." Therefore, in their paper, they are looking at perceived project success as reported by participants. (Serrador & Pinto, 2015).

Less formal nature of Agile project management requires even better measurement. Not only Agile methods require different measurement techniques, but due to focus on simplification risk associated with inappropriate success measurements are higher (Javdani et al.,2013).

When looked from the different perspective, in agile, involved parties can come to disagreement more easily, due to the lack of documentation. A recent study revealed, that even though agile promotes process transparency and closer communication, team members, team leaders, and product owners have different views on team performance. This difference is even larger than the one in traditional waterfall projects (Lindsjørn, 2016).

As agile moves towards larger enterprise projects, it faces the risk-averse public, that needs to know whether implemented methodology really works. One way to get an objective evaluation is through the use of metrics. Currently, there are studies on metrics usage in project management (they will be presented in the next chapter) but research is mostly based on case studies, and there is no unified set that promises to cover all aspects of project success.

That is why the research questions of this thesis are:

RQ1: What parameters should be measured in order to differentiate a successful Agile IT project, from the one that is not?

RQ2: What measuring techniques can be used to assess these parameters?

1.3 Summary of Chapter 1

A brief history of project management in software development industry was presented. The drawbacks of the stepwise development process in IT were investigated, including the context of such drawbacks. The transition from traditional waterfall project management approach to agile approaches was discussed. Differences between approaches were summarized. After that key concepts of all different agile frameworks were listed.

Several research areas related to agile project success were identified. The majority of these papers focused on comparative studies of successfulness of projects developed following an agile methodology, and those following traditional methodology. Other works investigated whether agile could be successfully applied in different contexts, such as projects of different sizes.

The majority of reviewed papers relied on perceived successfulness of the project as a key identification method of project success. Works that discussed the quantitative assessment of success were scarce. The importance of objective success measurement was further emphasized by findings from recent publications. So the research gap was identified.

To sum up, contemporary research lacks studies that would focus on providing a unified approach for quantitative evaluation of an agile IT-project. The need for such research leads to the formulation of two research questions that will act as a basis for this thesis.

CHAPTER 2: VERSATILE ASSESSMENT OF AGILE IT-PROJECT SUCCESS

Assessment of any project success is not as straightforward as it may seem. It can be best illustrated with an example. Sydney opera house construction took 15 years and project was 14 times over the budget. However, it became one of the main Australian landmarks and is now considered as a successful project (Jugdev & Müller, 2015). Opposite happened with the Motorola’s Iridium project. Even though it was well planned and managed it is now viewed as a failure (Shenhar & Dvir, 2007).

2.1 Holistic view on the Agile project success in the software industry

View on the project success assessment changed significantly over the last decade. Before, it was mostly focused on the Triple constraint, meaning that if the project met time, cost and scope requirements, it was generally considered successful. Recently, an idea that stakeholder satisfaction plays a major role in project success became widely recognized (Serrador & Turner, 2015). Even the 5th edition of PMBOK, released in 2013 defines project success beyond the triple constraint, which is not even mentioned there (PMBOK 5th edition).

The broader definition of project success became a topic of interest for many researchers. According to Serrador and Turner, there are two competing measures of projects success:

> "**Project efficiency**: meeting cost, time, and scope goals; and
> **Project success:** meeting wider business and enterprise goals as defined by key stakeholders."

In their research, they wanted to see how "Project efficiency" and broader "Project success" are related. After an extensive survey of 1,386 projects, authors found that project efficiency correlates moderately strongly (correlation of 0.6 with R-squared equal to 0.36) to overall project success. This means that other factors significantly contribute to the overall project success as well. Possible other factors, introduced by authors, can be best characterized as meeting stakeholder expectations in terms of product, performance, and risks. (Serrador & Turner, 2015).

While measuring cost, schedule and scope is a relatively straightforward operation, measuring meeting of stakeholder expectations is a more complex process. Turner and Zolin believe that different stakeholders evaluate project success on different timescale. They define three such time frames: End of the project, End plus months, End plus years. (Turner and Zolin, 2012).

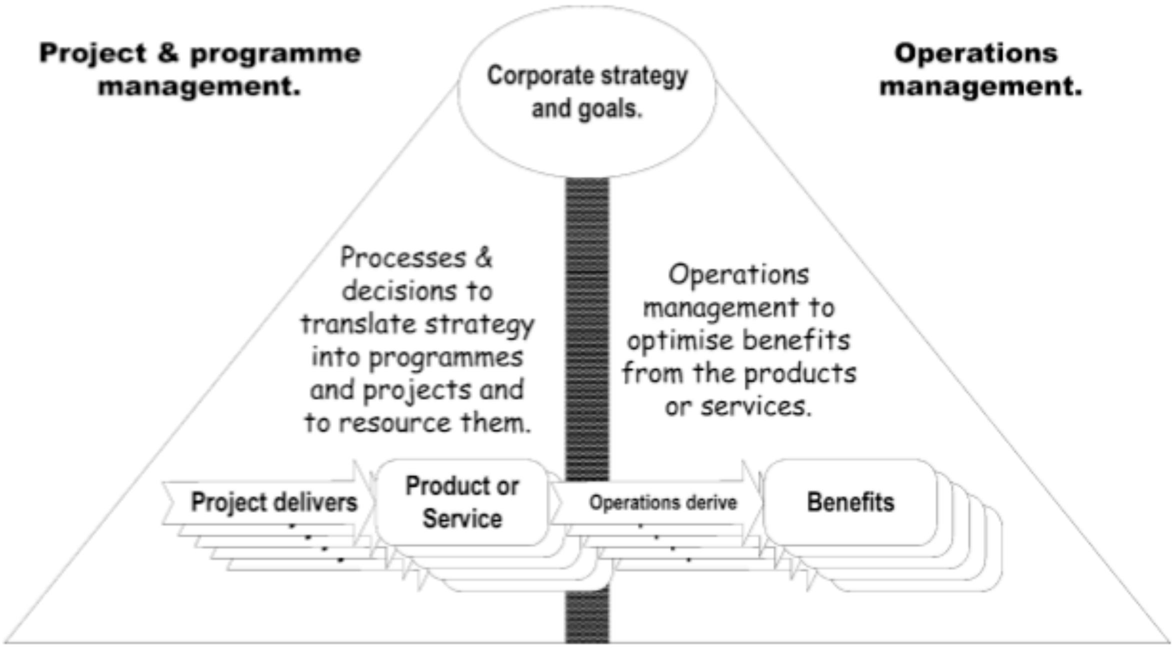
In this work, we focus on the first time frame till, including, the point "End of the project" for three reasons. First, it covers all stakeholders, including project manager and project team. Second, as soon the project is finished and handed to the customer, the project management ends. As we are mostly interested in fine tuning the project management process, we need to focus on the stage, when it happens. Third, a study of other timeframes would require longitude studies, which are out of the scope of this work and could be done in future research.

Based on the previously mentioned studies, we define 5 categories of success: Time, cost, scope, quality & stakeholder expectations. Each category includes a set of factors.

While categories are universal and don't depend on the industry, specific factors are industry specific. That is why it is needed to focus on more specialized studies, where it is applicable.

In an international study of 109 agile projects, Chow and Cao wanted to define critical success factors (CSFs) of agile software projects. First, based on existing literature, a preliminary list of success factors was created. Then, using reliability and factor analyses, it was reduced to 12 items, and assigned to four success categories: time, cost, quality, and scope. Later, based on survey data, using multiple regression techniques, six success factors appeared to be significant. (Chow & Cao, 2008)

Since overall project success doesn't depend only on project efficiency, it is important to consider additional factors. It is needed to define factors for stakeholder expectations. Terry Cooke-Davies (2002) aims to answer this question. After analyzing six "bodies of knowledge" he identified 60 central topics in eleven topic areas. "Anticipated benefits" turned out to be a core of formal and informal assessment of project success, by stakeholders. He points out that those, who started the project (stakeholders) hope to achieve certain goals (benefits). Since benefits delivery happens outside of the project group (see Figure 3), effective benefit delivery and cooperative management process are needed.



Project & programme management.
Corporate strategy and goals.
Operations management.
Processes & decisions to translate strategy into programmes and projects and to resource them.
Operations management to optimise benefits from the products or services.
Project delivers
Product or Service
Operations derive
Benefits

Figure 3 The importance of project management and operations management working together to deliver beneficial change from projects (Source: Cooke-Davies, 2002).

Using works of Chow, Cao and Terry Cooke-Davies, we can combine different categories of success and corresponding factors into a matrix (See table 3). Having a structured view on the project success would allow us to explore the full range of project success variables and be sure that nothing is missed or misinterpreted.

Table 3 Critical success factors

	Quality	Scope	Timeliness	Cost	Stakeholder satisfaction
1. Team environment	C	-	-	-	N/A
2. Team capability	-	-	C	C	N/A
3. Customer involvement	-	C	-	-	N/A
4. Project management process	C	-	-	-	N/A
5. Agile software engineering techniques	C	C	-	-	N/A
6. Delivery strategy	-	C	C	C	N/A
7. Effective benefits delivery and co-operative management process	-	-	-	-	C

2.2 Accessing the completeness of existing metrics portfolio

It is common to see metrics used as a primary tool for quantification of observations, including those in success assessment. General reasons for the use of metrics are quite broad. Kupiainen et al. (2014) conducted a study, where they determined real use-cases for metrics in agile context. According to authors, there are seven of them: “Iteration planning, Iteration tracking, Motivating and improving, Identifying process problems, Pre-release quality, Post-release quality and Changes in processes or tools.” It was found that majority of metrics is focused on planning and tracking. Authors note that applicable quality metrics and metrics to motivate and enforce improvement are potentially interesting for future research. (Kupiainen et al., 2014).

Having a holistic view of the project success allows us to see that there are two dimensions of success assessment. There are Success Criteria. By measuring them, it is possible to see whether the project or a single iteration is perceived successful post-factum (lagging indicators). Additionally, there are Critical Success Factors, the things that must go right (leading indicators) for the project to succeed in terms of success criteria. This cause-effect relationship between two means that measuring CSFs can be used to evaluate the general context of the project or iteration. As success is relative, it is important not only to quantify whether success criteria were met but also understand whether the general context was favorable or not.

For example, project perceived as mildly successful might have been completed in the context, where certain CSFs were not met at all. In this situation, other mildly successful projects, where all CSFs seem to be met would raise a question. Are these two project equally successful or team behind the project with less favorable context did a better job? As project stakeholders might have limited control over the context or even no control over certain CSFs, this questions becomes even more important for the adequate success evaluation.

This being said, knowing general context would help project managers to evaluate projects more objectively, to compare projects between each other and to identify areas for future improvements in a straightforward and structured way.

The metrics portfolio should cover both leading and lagging indicators in order to be suitable for all-inclusive project success assessment. Interestingly, a single metric can be both leading and lagging indicator (See figure 4). For example, employee engagement is a lagging indicator for management process/ability but a leading indicator for development process improvement, financial performance, etc.

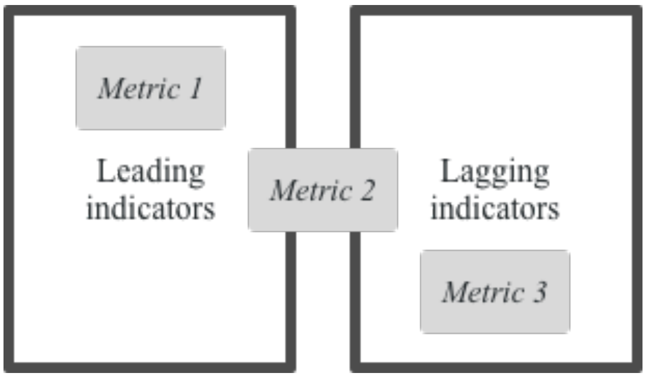


Figure 4 The possible status of metrics from the perspective of leading and lagging indicators.

The unequal distribution of metrics amongst use-cases calls for a systematic approach. In order to guarantee that different aspects of success assessment are covered, success criteria and CSFs were derived previously. To track changes, quantifiable metrics are needed. Since metrics are already widely used by agile teams in the software industry, it is first needed to gain an understanding of what areas of the success assessment matrix are already covered with existing metrics.

A number of studies relating to different aspects of the use of Agile metrics in contemporary organizations have been done. High-quality discussions on this topic can be found in Abbas, N. et al. (2010), Petersen, K. et al. (2010), Petersen, K. et al. (2011), Petersen, K. (2012), Staron, M. et al. (2010).

In their work Kupiainen et al. (2015) conducted a systematic review of studies on the usage of agile metrics in software projects, including those from the previous paragraph. The systematic review covered 30 papers in total that were published till recently. Even though works were of different scope and used different methodologies, they all presented metrics used by contemporary organizations. Authors of the review analyzed those metrics and synthesized a list of high influence metrics based on a number of occurrences and perceived importance factor (See Table 4). This list will act as a basis for the future analysis. In order to keep the work brief and focused, presentation of the detailed review of all the above papers will be avoided. Therefore, interested readers are referred to Kupiainen et al. (2015), where they will be able to get acquainted with a detailed overview of all the studies.

Table 4 High influence metrics based on number of occurrences and perceived importance factor (Kupiainen et al., 2015)

Metric	Number of occurrences	Importance factor	Sum of ranks / 2
Velocity	15	3	1
Effort estimate	12	3	1.5
Customer satisfaction	6	3	2.5
Defect count	8	3	5
Technical debt	2	3	5
Build status	2	3	5
Progress as working code	1	3	6.5
Lead time	4	2	7
Story flow percentage	1	2	9.5
Velocity of elaborating features	1	2	9.5
Story percent complete	1	2	9.5
Number of test cases	1	2	9.5
Queue time	1	2	9.5
Processing time	1	2	9.5
Defect trend indicator	1	2	9.5
Work in progress	6	1	10
Number of unit tests	5	1	11
Cost types	1	1	14
Variance in handovers	1	1	14
Deferred defects	1	1	14
Predicted number of defects in backlog	1	1	14
Test coverage	1	1	14
Test-growth ratio	1	1	14
Check-ins per day	3	N/A	16
Cycle time	2	N/A	16.5

However, it is not possible to rely only on the list derived from the systematic review. It was published a couple of years ago, meaning that the most recently adopted metrics won't be presented there. Since agile methodology is rapidly changing and evolving, more recent studies should also be taken into account.

Greening, D. R. (2015) believes that behavioral metrics that are typically used by teams for self-assessment have some flaws in larger settings. They can't easily scale in larger enterprises, where several independent small teams work simultaneously. That is why he looks on the metrics from the different perspective and outlines scalable metrics that help to avoid dysfunctions related to meeting metrics-based targets.

Table 5 List of metrics (Source: Greening, D. R., 2015)

Metric	Description
True Sprint Length	Number of time needed to really roll out a feature, might be several sprints
Velocity	Number of BGIs or Story points per sprint
Velocity deviation	Deviation of velocity from sprint to sprint
Forecast Horizon	The point in the Product Backlog just before the first unestimated Product Backlog Item
Lead time	Shortest sprint time or a minimum time for consecutive sprints of several involved teams
Downstream impact	In case of team interdependencies, the number of downstream teams affected

Padmini, K. J. et al. (2015) look at the overhead costs associated with the use of metrics in agile software development. Moreover, authors believe, that due to the iterative and incremental nature of the agile software development metrics used in traditional approaches might be inappropriate. By analyzing practices of 24 development companies, they came up with a list of 10 metrics that can be beneficial to the agile software development process. Meaning that benefits from using these metrics outweigh the cost of measuring, tracking and other processes involved.

Table 6 List of metrics (Source: Padmini, K. J. et al., 2015)

Metric	Description
Delivery on time	Ratio of features done in planned release schedule
Work capacity	The sum of all work reported during the Sprint, whether the SBI toward which the work was applied finished or not.
Unit test coverage for the developed code	% of code covered by unit tests
Percentage of adopted work	∑(Original Estimates of Adopted Work) ÷ (Original Forecast for the Sprint)
Bug correction time from new-to-closed state	Bug correction time from new-to-closed state
Sprint-level effort burndown	Sprint burn-down chart for tracking of team level work progress during a sprint, identifying the work remaining.
Velocity	∑ of original estimates of all accepted work
Percentage of found work	∑(Original Estimates of Found Work) ÷ (Original Forecast for the Sprint)
Open defect severity index	Number and severity of open defects

Focus factor	Velocity ÷ Work Capacity

The problematic of reporting to stakeholders was taken further by Boeman M.P. (2015). They developed a practical model. The model was then checked for reliability, usefulness, and feasibility using a case-study approach. The practical value of separate metrics and monitoring method was confirmed.

Table 7 List of metrics (Source: Boeman M.P., 2015)

Metric	Description
Added PBIs	% of PBIs that are new on the backlog of a given iteration
Changed PBIs	% of Changed PBI in a iteration
Enhancement Rate	% of items from the last backlog included to the production in the next backlog
Estimation Shift	Positive or negative % change in estimation of an iteration
Expenses Prognosis	Extrapolation of expected product costs
Effort At Risk	Sum of rejection impact ratios per PBI
Priority Shift	Sum of difference (shifts) in priorities
Project Size	Sum of size estimates or quantity of PBIs
Project Size Remaining	Number of PBIs remaining
Rejected PBIs	Number of rejected PBIs
Scope Prognosis	Expected level of functional completness
Software Quality	Measured, using quality properties described in ISO/IEC 25010
Time Prognosis	Expected end iteration

Till now a list of metrics should be quite extensive. However, since it may take more than a year for research articles to be published and available it is also needed to look for data beyond the academic literature. Annual “State of Agile” report is a largest and longest-running agile survey in the world. It is being conducted since 2006. 3880 respondents participated in the 10th annual report, that was released in 2016. Respondents were asked “How is success measured on a day-to-day basis?”, bases on the answers, a list of metrics used in organizations was comprised:

- Velocity
- Iteration burndown
- Release burndown
- Planned vs. actual stories per iteration
- Burn-up chart
- Work-in-Process (WIP)
- Planned vs. actual release dates
- Customer/user satisfaction
- Defects in to production
- Defects over time
- Budget vs. actual cost
- Business value delivered
- Defect resolution
- Cycle time Estimation accuracy
- Individual hours per iteration/week

- Test pass/fail over time
- Scope change in a release
- Cumulative flow chart
- Earned value Customer retention
- Revenue/sales impact
- Product utilization

After metrics from all the sources had been merged, the list included 75 metrics. Since original definitions of those metrics were available in the papers, it was possible to eliminate duplications not only by looking for the direct matches of names but also by analyzing the exact parameters that those metrics were made to measure. After accounting for duplicates, the list comprised 55 items. The second check of the list was made based on the time scale parameter. As it was mentioned earlier, we are interested in the project evaluation happening the latest at the end of the project. 4 metrics that do not comply with the requirements mentioned above were removed from the list. The final list included 51 metrics.

As the goal of this thesis is to develop a unified, prioritized set of quantitative metrics of success of Agile IT projects it is needed to check whether all aspects of the success assessment matrix are covered by the list of metrics brought together in the previous chapter.

Firstly, we will see, whether these metrics can act as lagging indicators for evaluating the project based on the success criteria. In the following paragraphs metrics from the list that was developed in the previous chapter will be assigned to five success criteria.

Metrics by success criteria

Quality

Based on the work by Chow & Cao (2008) quality means delivering good product or project outcome. However, measuring the quality of a software product is not a straightforward process. Software products are very diverse and are usually extremely customized, meaning that it is challenging to create up-to-date, objective and universal quality test, so companies have to rely on rules of thumb in many cases.

All else equal, product with less defects can be considered as a product of a higher quality. It is important to mention, that an automated testing is a popular technique utilized by agile teams in order to find defects throughout all products faster. Automated testing might be perceived as something more related with faster development, rather than product quality. It is not the case, since writing tests requires substantial time and effort investments that outweigh gains in speed.

That is why metrics that allow to measure an absolute number or changes in defects and that show how good product is covered with tests should be assigned to the “Quality” success criteria.

After reviewing the list of metrics that was comprised earlier, it turned out, that 16 metrics, or almost ⅓ of all metrics is related to quality.

Scope

Chow & Cao (2008) define scope category as meeting all requirements and objectives. As it was mentioned, agile is an iterative framework, that welcomes new requirements based on the feedback from end users and/or customers, those who in the end evaluate whether all requirements and objectives were met.

In scrum working process is divided into relatively short iterations, during each of which a certain functionality should be developed. Team decides on their own about the number of backlog items (usually measured in story points) that will be included to the iteration. This means that scope can be positively affected by productivity of the team. More story points it includes into the sprint, means more features it builds in the limited period of time.

Since usually after each iteration product is evaluated and scope is refined, it is crucial for the team to have a working piece of software in the end of each sprint.

That is why metrics that determine the productivity of the team and its ability to deliver working software in the end of iterations will be assigned to the scope criteria of success. After analyzing the list, it turned out that there are 10 (19.6%) such metrics there.

Timeliness

Time category means delivering product on time, as planned (Chow & Cao, 2008). It is one of the most fundamental success criteria. In agile, in particular, it plays an interesting role, as projects are split into limited number of iterations, meaning that instead of one big deadline, there are several small ones throughout the whole project.

This opens more possibilities for evaluating the delays, interdependencies and makes it easier to change and adjust the process.

That is why metrics associated with process time and end date estimations were included into this category. There were 13 of them, or a bit over a ¼ of all metrics. Meaning that this category is perceived as really important.

Cost

Cost category means delivering product within estimated cost and effort. Interestingly, in the software industry, time spent on the project and it's cost are usually highly correlated, due to the high proportion of variable costs in the development process. Basically, salaries of the developers comprise a significant share of above mentioned variable costs.

For this reason, in addition to classical financial observations, hours spent on a project may be used to judge, whether the project is under control in terms of costs.

Using the principles described above, metrics from the list were assigned. In total, just 4 (7.8%) of them fell into this success category.

Stakeholders satisfaction

Stakeholder satisfaction is the least obvious success category. Agile practitioners, by promoting stakeholder collaboration are trying to ensure the engagement of all parties involved in the process. Engagement then makes it possible to find out the expectations that stakeholders have for a certain project.

An additional aspect that should be taken into account is predictability. It can be used to assure that expectations that were already set by stakeholders are regularly met.

In the list, there were 8 (15.6%) metrics that could help to measure either predictability of development or engagement of stakeholders

Summary

Metrics were assigned to certain success categories, meaning that assigned metrics may be used as an indicator of project success in a certain category. It is already clear that metrics used by contemporary organizations are predisposed towards success criteria, as no metrics were left unassigned.

Their distribution is not equal between different criteria (see Figure 5). The three leading criteria are Quality (31%), Timeliness (25%), Scope (20%). This can be partially explained by the nature of the relationship between time and cost criteria, mentioned in a corresponding section of this chapter. Interestingly, Stakeholder Satisfaction is measured quite extensively.

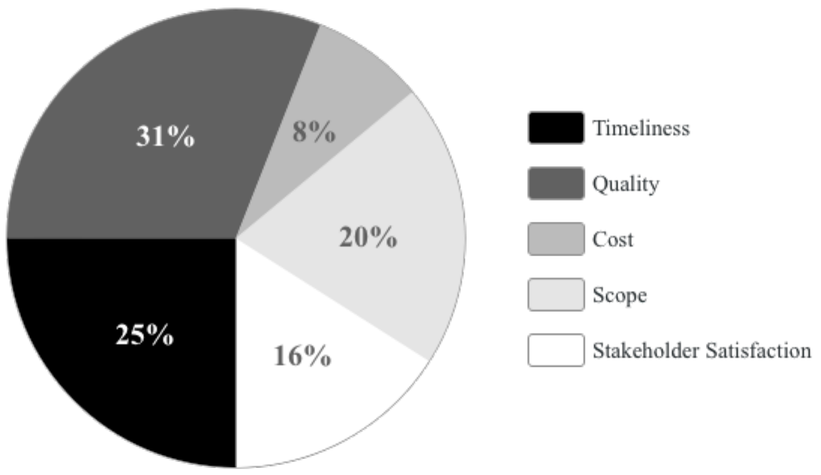


Figure 5: Distribution of metrics amongst different success criteria

Metrics by Critical Success Factors

Attributes of the majority of CSFs was presented in work by Chow & Cao (2008)**.** For this reason, the structure of the following paragraphs will be as follows. For each CSF its attributes will be briefly discussed, and metrics will be assigned, if possible. The formulations will be interpreted using the information from the same source as used for those from the original paper: "Agile software development" by Leon & Koch (2004).

Team environment

- Collocation of the whole team

  It is an organizational attribute which basically intends to clarify, whether the whole team is located in the same physical location or not. Today it is not uncommon to see teams working remotely, thanks to the development of teleconferencing software. However Agile has a prejudgment against such setups, since communication assumptions mentioned in the agile principles are much easier to meet when the whole team is together.

  There are no metrics that allow detecting whether the team is collocated in the list.
- Coherent, self-organizing teamwork

  The agile principle states that working in self-organizing, i.e. self-managed teams is a way to achieve technological excellence. Additionally, in this Philosophy team is perceived as an independent unit, that can influence decisions and provide insight to management and clients. This positively affects the motivation of team members as well, as they see the impact of their work.

  In the list, there are no metrics that make it possible to distinguish self-organizing team from a traditional command and control team.
- Projects with small team

  Emphasis on face to face communication over extensive documentation puts a "natural" limit on the team size. This means that in agile team size is assumed to be somewhat small, while there are no direct restrictions. Of course agile can be applied to projects where large teams might be needed, but in this case, the team is usually split into smaller sub-teams that operate more or less autonomously.

  Metrics that could allow tracking the team size are not present on the list.
- Projects with no multiple independent teams

  Agile does not offer any straightforward approaches to managing multiple teams. As mentioned earlier, if a larger team is needed it is possible to split it into independent sub-teams. It is challenging, and project management has to be flexible and innovative to make it work, but it is possible.

There is 1 metric that indirectly addresses issues of working with multiple teams.

In general, this CSF is quite specific, as it focuses on the team design technicalities. The total number of metrics from the list that allow measuring attributes of this CSF = 1.

Team capability

- Team members with high competence and expertise

  In the core of agile Philosophy, there is an assumption that teams are built around competent and motivated people. The competent team is a team that feels that it is able to complete the tasks of the project. Availability of all individuals with all necessary expertise within the team is another key for the development of a technically excellent product.

  There are no metrics that would help to identify competencies and expertise of the team.
- Team members with great motivation

  Agile principles suggest building projects around motivated individuals. For this reason, an environment of trust and support is created. Basically, it is expected from motivated teammates to go an extra mile, behave how they believe is best for the project success or at least openly raise questions and discuss concerns they have.

  There are no metrics in the list that would allow measuring the motivation of team members.
- Managers knowledgeable in agile

  The role of managers in agile changes significantly compared to the one in the waterfall setting. Instead of actually planning the project and giving orders, a manager would rather collaborate with the team in order to come to a consensus on how and when things should be done. Understanding of such change is critical for the agile project success.

  There are no metrics that could directly indicate the knowledge managers have about agile.
- Managers who have adaptive management style

  The goal of project manager doesn't change in agile. It is still about keeping the project on track. What is different is the approach. Instead of using power leverage, project managers in agile rely on leadership. Project manager creates a collaboration-friendly environment where he or she then leads the way in this cooperation.

Metrics that would allow defining the management style are not present on the list.

- Appropriate technical training to team

  This attribute relates to the fact that team members should be aware of the paradigm where they operate. No only should they know their own responsibilities and what is expected from them, but also to be able to assist those who might have questions about the development process. This can be achieved by holding technical training sessions.

  There are no metrics that would help to distinguish a team that has appropriate technical training from the one that doesn't.

<u>*Customer involvement*</u>

- Good customer relationship

  The key to good relationship with customers is in finding a balance between collaboration and contract. A good relationship should be started by both parties (customer and contractor) initially agreeing that agile requires learning, that might lead to changes in scope and timeliness of the project. When customers are presented with benefits of such approach and make an independent decision, there will be much less room for conflicts, meaning that good relationship will prosper.

  In the list, there are no metrics that provide insight into the relationship with customers.

- Strong customer commitment and presence

  Agile requires a strong customer commitment since customers are ideally involved to the project on the full-time basis as an equal team member. It is an additional cost from the customers' perspective since people involved in the project should spend less time on other responsibilities they might have. Not understanding this and not willing to commit, would threaten the project success, since customers provide valuable feedback and re-define requirements from the project.

  There are 2 metrics that indirectly allow measuring the customer commitment.

- Customer having full authority

  Agile welcomes change and because of that project requirements are specified in fewer details and allowed to evolve. Constant approving, disapproving and prioritizing ever-changing requirements is needed. Customer (with technical input from the development team) has full authority to do it since it is one of the vital tasks in the development process.

There are no metrics in the list, that would allow to quantify this attribute.

To sum up, this CSF emphasizes the customer role in the project success. In the list, there are only 3 metrics that might be used to measure this CSF.

Project management process

- Following agile-oriented requirement management process

  There are two extremes in the requirement management process. It can be completely informal when everything happens in oral form or by chaotic email exchange. It can be rigorously formal when all requirements are written down in a very detailed manner, and every single change to initial requirements needs to be discussed and evaluated by a certain Configuration Control Board. In agile, especially Scrum, a mixed approach is utilized. Requirements are written down to the product backlog, then prioritized by a client who then may or may not accept them as done.

  3 metrics can be used to indirectly define the requirement management process.
- Following agile-oriented project management process

  Similarly, to the requirement management process, project management can also be either informal or very formal. Agile, as a lightweight framework, tends to be closer to the informal part of the spectrum. Instead of estimating product attributes, such as size, effort and schedule are being estimated. The progress is tracked quite thoroughly. When changes are identified, instead of taking corrective actions, the whole plan is changed.

  In the list, there are 2 metrics that are related to the project management process.
- Following agile-oriented configuration management process

  While configuration management process is not directly provided by the agile framework, certain assumptions can be made. On the one hand, automated code version control tools should be in place, since agile welcomes changes and they shouldn’t overwrite each other. On the other hand, formal document control is not crucial, as the role of documentation is de-emphasized in agile. As teams work in an iterative manner, a certain baseline should be developed early on, to act as a basis for future work. Moreover, releases in agile are happening much more often and faster, meaning that configuration item identification and build management processes should be adjusted as well. Modern tools allow for that, but additional training might be needed.

  There are no metrics on the list that would allow gaining insight about the configuration management process.

- Good progress tracking mechanism

  Agile allows tracking progress during and in-between increments. The main challenge of such tracking is related to dealing with all the changes that are happening. It may be problematic to estimate the project end date, while product backlog is constantly growing. However, there is a solution. Changes in backlog are expected to level out eventually. If the velocity of the team is greater than the rate at which new backlog items are added, it is possible to estimate when all backlog items will be developed. Good progress tracking allows to do that and take actions if needed.

  There are 4 metrics that allow judging about the progress tracking mechanism.

- Strong communication focus with daily face-to-face meetings

  As one of 12 Agile Principles states: "The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.". The agile framework values verbal and nonverbal communication between team members in person more than extensive documentation, because it is believed that much more information can be shared this way. One of the best examples is Scrum daily stand-up meeting, during which team members are sharing their current status and concerns they have.

  There are no metrics that allow measuring whether there is a strong communication focus on daily in-person meetings.

- Honoring regular working schedule

  Overtime is a norm for many modern organizations, where teams are struggling to meet high expectations that were set from outside. Agile is a framework, where overtime is seen as something abnormal. The nature of agile allows for leveling up demands and maintaining a pace that everyone, including customer, would approve.

  There is 1 metric that indirectly might allow estimating the working schedule in the organization.

Agile software engineering techniques

- Well-defined coding standards up front

  Coding standards have two key benefits. First, they help to keep the technological excellence of the code high. Second, they facilitate the development process. It is better, when developers have a common ground, i.e. understand the code of each other more easily and quickly, by having a common approach to coding.

There are no metrics that would help to understand the quality of coding standards in the organization.

- Pursuing simple design

  As estimates about the future are often inaccurate, developers shouldn't account for them, when developing a certain story. What is meant by that, is that simplest possible solution should be developed for each single feature, without accounting for interdependencies with the planned features. It is believed that this approach would lead to less rework compared to the one where future features are taken into consideration.

  There are no metrics that might help to identify pursuing simple design.

- Rigorous refactoring activities

  Refactoring is related to keeping the integrity of the product high. Developers should aim at keeping the system as a whole as simple as possible. That is why, when there is an opportunity to improve previously written code, it should be done.

  Metrics helping to identify whether there is refactoring done, are not present in the list.

- Right amount of documentation

  While in Agile working software is valued more than comprehensive documentation, there is still no direct indication of how documents should be managed. The general idea is that documents should have a clear purpose, be concise and simply engineered.

  There are no metrics related to the documentation in the list.

- Correct integration testing

  Instead of integrating all the code at the end of each increment, most agile methods propose to do it more regularly. Some teams do continuous integration. The idea is that each developer should integrate their code as soon as it is done. If there are any problems with integration, then work is not considered done.

  Metrics related to integration testing are not on the list.

Delivery strategy

- Regular delivery of software

  Software should be delivered regularly and often in order to mitigate risks. This way customers can provide feedback on the product and needed adjustments can be verified.

  There are three metrics that help to identify, whether the software is delivered on a regular basis.

- Delivering most important features first

  Customer or product owner as a main representative of the customer is the one who is responsible for prioritizing items in the backlog. Same people decide whether developed functionality matches with the one intended by the backlog item. To comply with the direction set by the customer, a team should try to deliver items with higher priority first.

  There is one metric that indirectly helps to identify whether most important features are delivered first

Effective benefits delivery and co-operative management process

- Predictable development process

  Effective processes lead to predictable results that in turn can make stakeholders continuously see the value of the product developed. In most cases, final benefits are not directly delivered by direct subordinates

  There are six metrics that can help to see whether the development process is predictable.
- Stakeholders are engaged

  All parties involved in the development process should be satisfied with the process. Active engagement allows sharing opinions and expectation in a consistent manner, meaning there is a higher chance they will be heard.

  There is just one metric that helps to measure the engagement of stakeholders.

Summary

It turns out that existing metrics portfolio, used in contemporary organizations is not suitable for the evaluation of the project context. This means that other measuring techniques should be used. To understand which ones are more suitable, it is needed to define the purpose of dealing with factors. As mentioned earlier, factors are the things that “must go right for the project to succeed”, meaning that they represent a best practice of development with accordance to agile project management.

The checklist seems most suitable for this purpose. Hales B. and Pronovost P. define checklist as “a list of action items or criteria arranged in a systematic manner, allowing the user to record the presence/absence of the individual items listed to ensure that all are considered or completed.” However, a checklist is not just a simple enumeration of items; rather it is a prioritized list that points out to essential attributes that should be considered in a certain field. (Hales, B. M. and Pronovost, P. J., 2006)

That is why it is needed to additionally develop a checklist for CSFs of project success and their attributes. The checklist should not only list factors and their attributes but also provide information about the relative importance of certain factors.

2.3 Summary of Chapter 2

The complexity of project success was noted, and it was decided to develop a holistic view of project success. In order to define success criteria, the concept of project success was split into two components: Project efficiency and General project success. Based on those findings and other insights from the literature it was decided to make a list of success criteria that would include: Cost, Timeliness, Quality, Scope and Stakeholder satisfaction.

Next, it was mentioned that industry-specific success factors should be taken into consideration. After reviewing the literature, a paper investigating CSFs of project success was found. That way CSFs for four out of five success criteria were found. To fill in the gap, other works were reviewed, and CSF for stakeholder satisfaction was found. By combining success criteria with the factors that influence those criteria, a matrix of agile project success was created.

It was decided to see whether existing measurement techniques utilized by the contemporary organization could potentially be used to both get an insight about meeting success criteria and CSFs influencing those criteria. As metrics are widespread in software industry a unified list of metrics that would reflect the actual metrics used in contemporary organizations was created. To do that an extensive literature review published in 2015 was taken as a basis and metrics from several more recent papers and the largest survey of agile teams were added. After cleaning from duplicates, the final list included 51 metrics.

After the list of metrics was comprised it was decided to check whether it could cover both success criteria and factors. The possibility to do so was explained by the fact that certain metrics can be leading indicators (Factors) and lagging indicators (Metrics) at once. In the case of criteria, it turned out that all metrics could be assigned to certain success criteria. In the case of factors, just a few metrics were suitable. This lead to a conclusion, that current measurement techniques utilized in a modern organization are partly biased towards lagging indicators.

In order to compensate for such a bias, it was decided to develop a checklist of factors, that could be used in addition to metrics currently in use. It was found in the literature, that good checklist should emphasize areas that need more attention. For that purpose, the expert opinion could be used.

CHAPTER 3: A CHECKLIST FOR ASSESSING PROJECT CONTEXT

As mentioned earlier, a checklist should be organized in a systematic manner. There is no need to do categorization. Critical success factors are the categories, and their attributes are the criteria that should be considered. The question that is left unanswered is their relative importance. Finding it would make the checklist more useful for party assessing the project success. There are three reasons for that. First, assuming limited time resources of the team that can be spent on progress and context tracking, it is needed to know what should be measured and what is optional. Second, to make better decisions one must clearly understand the interrelationship of items on the checklist. Third, the perceived importance of industry specific CSFs, such as “Agile software engineering techniques” is interesting, since it would allow to see how easily this checklist could be adopted to use cases beyond software development.

Additionally, the list of CSFs that was developed by Chow and Cao (2008) was supplemented by a new category “Effective benefits delivery and co-operative management process” as a part of this work. The relative importance of this category could act as an evidence that the addition to the list was beneficial. Moreover, Chow and Cao (2008) propose their own view on the relative importance of CSFs, implying that more categories a factor influences, more is its relative importance. Study might help to see if their view is confirmed.

To define the relative importance of categories on the checklist it was decided to make use of expert opinion. There are few methods available that might be used for this task, but the fuzzy Analytical Hierarchy Process (AHP) fits best of all.

3.1 Fuzzy Analytic Hierarchy Process (AHP) analysis method

In our case, multiple different CSFs need to be assessed. Multi attribute decision making (MADM) techniques are suitable for this task, since they allow to deal with a selection of criteria that are measured on different scales. Since project context estimation is a vague process, another advantage of MADM over other decision-making methods is that no single measurement scale is needed for all criteria. Out of all MADM techniques the Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS), Outranking, and AHP are used most often.

AHP, developed by Saaty (1980) is the most popular method used in the literature. It breaks down complex systems, into easier to comprehend elements, organized into a hierarchical system. The main tool that is proposed by this method is a pairwise comparison. Basically every element on every level of hierarchy is compared with each other on a nominal scale. Based on these individual comparisons, an individual matrix is created. From the matrix it is possible to derive comparative weights of each of the categories.

The major drawback of these methods, including AHP, is that they require input of objective information. In reality, this type of information is either lacking, or there is no sufficient amount of it. In these cases, it is needed to rely on opinion and knowledge of experts. To mathematically account for imprecise estimation and general uncertainty that come with the involvement of experts, additional components were incorporated to the methods mentioned above. These components follow fuzzy logic that was developed by Zadeh (1965) who introduced the fuzzy set theory. It was developed to help to deal with uncertainty of human thought. This theory was then applied in variety of fields, including decision-making.

Fuzzy AHP is actively used for incorporation of expert opinion. Calabrese et al. (2013) apply the fuzzy AHP approach to a company in ICT field. Their goal was to get an understanding of best practices in competing organizations. They obtained data by interviewing managers of the analyzed companies and used it to rank several organizational aspects by their importance. Other examples of applications of fuzzy AHP in IT include works of Lee et al. (2008) and Cebeci (2009).

There are many fuzzy AHP methods proposed by various authors. In this paper, it was decided to follow the approach developed by Chang (1996), since it allows to estimate the relative weights of the categories, without tremendous computations, that other methods have in order to offer possibilities out of the scope of this research. Fuzzy AHP method by Chang uses triangular fuzzy numbers and is computed in a similar manner to the crisp AHP, thus not requiring additional operations (Bozbura et al., 2007).

3.2 Questionnaire for practitioners of Agile PM in IT projects

Since there are seven CSFs to compare, the number of pairs in pairwise comparison was equal to 21. Respondents were asked to choose a factor that is equally or more important out of pairs and to evaluate the relative importance of each on the scale of 1 to 9 (1 being "equally important" and 9 "absolutely more important"). The question that was asked for each of the 21 pairs can be seen below (Table 8).

Table 8 Questionnaire for Fuzzy AHP

#	**When you evaluate the context of the project, for you, it is more important to assess…**
1	Team environment or Team capability
2	Team environment or Customer involvement

#	**When you evaluate the context of the project, for you, it is more important to assess…**
3	Team environment or Project management process
4	Team environment or Agile software engineering techniques
5	Team environment or Delivery strategy
6	Team environment or Effective benefits delivery and co-operative management process
7	Team capability or Customer involvement
8	Team capability or Project management process
9	Team capability or Agile software engineering techniques
10	Team capability or Delivery strategy
11	Team capability or Effective benefits delivery and co-operative management process
12	Customer involvement or Project management process
13	Customer involvement or Agile software engineering techniques
14	Customer involvement or Delivery strategy
15	Customer involvement or Effective benefits delivery and co-operative management process
16	Project management process or Agile software engineering techniques
17	Project management process or Delivery strategy
18	Project management process or Effective benefits delivery and co-operative management process
19	Agile software engineering techniques or Delivery strategy
20	Agile software engineering techniques or Effective benefits delivery and co-operative management process
21	Delivery strategy or Effective benefits delivery and co-operative management process

A mix of purposive sampling and self-selection sampling techniques were used. Speakers of major agile conferences and notable experts in the field were approached individually, and links to the survey were published in closed online communities of agile practitioners.

In total 14 responses were received from experts. The respondents had many years of experience as scrum masters, product owners, and senior software developers. Respondents worked both for Russian and international companies and came from diverse backgrounds.

Answers of experts were transformed into triangular fuzzy numbers. For example, if an expert indicated that Team capability is more important than Delivery strategy and that it is

"Strongly important", their answer would be transformed into (6, 7, 8) when comparing Team capability and Delivery strategy and into (1/8, 1/7, 1/6) when comparing them vice versa. This way the whole pairwise comparison matrix is filled in with fuzzy numbers.

After all matrices had been created, they were aggregated into a single matrix. Then fuzzy comparison values were found, transformed into relative fuzzy weights and finally into normalized relative weights for each CSF.

Based on these weights, as a result, a prioritized list of CSF was created. Results of the study and managerial application were then discussed.

3.3 Results interpretation and discussion

Critical Success Factors were ranked by their relative importance using the Fuzzy AHP method. Table 9 shows the final ranking (where 1 – most important, 7 – least important) of each CSF, including the normalized relative weight. Since the fuzzy version of AHP was used, it is possible to declare that vagueness of expert opinion was accounted for.

Table 9 Results of Fuzzy Analytic Hierarchy Process

Rank	Critical Success Factors	Normalized relative weights (Ni)
1	Team capability	19%
2	Customer involvement	18%
3	Effective benefits delivery and co-operative management process	17%
4	Agile software engineering techniques	14%
5	Project management process	12%
6	Delivery strategy	10%
7	Team environment	9%

It can be seen from the results, that the most important factor for evaluating the project context is Team capability, it is followed by Customer involvement and Effective benefits delivery and co-operative management process. Positions 4 to 7 are occupied in the following order: Agile software engineering techniques, Project management process, Delivery strategy, Team environment. There are three takeaways based on the results of the study.

First, it can be clearly seen that experts view on the relative importance of CSFs differs from the one of Chow and Cao (2008). According to the authors, there are two different groups of factors based on the amount of success criteria they influence (>1 or =1). First group includes: Delivery strategy (3), Team capability (2), Agile software engineering techniques (2). Second group includes: Team environment (1), Project management process (1), Customer involvement

(1). This can be explained by the fact that while a certain factor can influence several success criteria at once, its cumulative perceived impact on project might be relatively small.

Second, while first two factors in the ranking can be expected to be in the top, since they are the cornerstones of the Agile methodology. The third factor was added to explain the Stakeholder Satisfaction success criteria. The fact that it found its place in the top half of the ranking can act as a further confirmation that Stakeholder Satisfaction plays an important role in the project success assessment and should be considered.

Third, it is possible to split CSFs into two categories. First three factors with Ni, higher than average (14,28%) would be considered as core items that should be considered, when evaluating the project context. Last four factors would fall into the recommended category, meaning that it would be beneficial to consider these factors, when evaluating the project context.

Based on the findings of the study, it is possible to compose a checklist for project managers (See Table 10).

Table 10 Checklist for evaluating the project context

CORE ITEMS
Category: **Team capability**
☐ Team members have high competence and expertise ☐ Team members have great motivation ☐ Managers are knowledgeable in agile ☐ Managers have an adaptive management style ☐ There is an appropriate technical training to the team
Category: **Customer involvement**
☐ There is a good customer relationship ☐ There a good customer commitment and presence ☐ Customer has a full authority
Category: **Effective benefits delivery and co-operative management process**
☐ Development process is predictable ☐ Stakeholders are engaged
RECOMMENDED ITEMS
Category: **Agile software engineering techniques**
☐ Well-defined coding standards are available up front

☐ Team is pursuing simple design ☐ Rigorous refactoring activities are present ☐ Right amount of documentation is developed ☐ Correct integration testing is done
Category: **Project management process**
☐ Agile-oriented requirement management process is followed ☐ Agile-oriented project management process is followed ☐ Agile-oriented configuration management process is followed ☐ There is a good progress tracking mechanism ☐ There is a strong focus on communication with daily face-to-face meetings ☐ Regular working schedule is honored
Category: **Delivery strategy**
☐ Software is delivered regularly ☐ The most important features are delivered first
Category: **Team environment**
☐ The whole team is collocated ☐ Teamwork is coherent and self-organizing ☐ Team size is small ☐ There are no multiple independent teams

3.4 Summary of Chapter 3

In the beginning of the chapter, the need to have a prioritized checklist was supported with three arguments. Project managers would be able to save time by focusing on core items, to make better decision by understanding relative importance of each category and to understand whether it is possible to apply the checklist beyond the IT related implications.

MADM techniques were presented, and AHP was chosen as the most appropriate in terms of goal and scope of the project. The concept of fuzzy logic was then presented as a way to incorporate the ambiguity of human thought into the analysis. A fuzzy AHP approach proposed by Chang (1996), making use of triangular fuzzy numbers, was chosen.

The pairwise-comparison questionnaire, was developed and presented. Purposive sampling and self-selection sampling techniques were then announced and the process of collecting the data was presented. In total 14 responses from diverse group of experts were received.

Based on the fuzzy AHP analysis output the final ranking of CSFs was created. Three major takeaways were presented. First, relative importance of factors cannot be solely explained by the number of criteria these factors influence. Second, factor related to the Stakeholder Satisfaction success criteria, that was added to CSFs identified by Chang (1996) proved to have a normalized relative weight higher than average, confirming that the above mentioned category should be considered when assessing the project success. Third, two groups of factors were composed, based on their Ni: core items and recommended items.

In the end, based on the findings of the study a checklist for project managers was developed.

IMPLEMENTATION PROCEDURE

In order to simplify the onboarding process for project managers that are willing to implement the proposed approach an implementation procedure was formulated. It can be split into three steps.

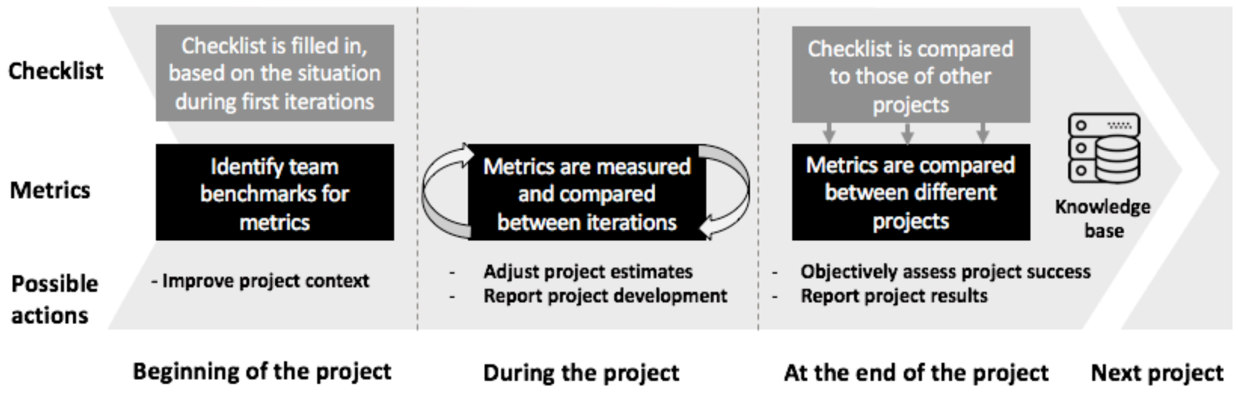
First step happens in the beginning of the project. Checklist is filled in to record the initial setting of the team and make project manager aware of all possible attributes that influence the project success. Right after the checklist is filled in, certain changes might be made, based on the items left unmatched or mismatch of checked items in the current checklist compared to those in the checklists of the previous projects. Next, during first iterations, team identifies their own benchmarks for metrics, such as team velocity. Based on them it is possible to make better projections regarding the ongoing project.

Second step takes place throughout the project. On this step, project managers track metrics, compare them with team benchmarks. They can adjust project estimations. Also on this stage certain metrics, especially those that can be easily represented visually, can be used to deliver development status updates to external stakeholders in consistent and organized manner.

Lastly, third stage occurs in the end of the project. Based on metrics and projection accuracy throughout the project it is decided to what degree a project was successful. Then, previous projects with similar perceived success level are identified (while team benchmarks are unique to every project, it is possible to compare deviations or differences in differences for majority of metrics). After that, checklist is compared to those of projects with similar perceived successfulness. Checklists help to put perceived project success into context, and see whether teams managed to “walk the extra mile”, or failed to benefit from the favorable project environment. Knowing that information allows to assess project success more objectively and enhances report on project results, by providing “a bigger picture”.

In the end Checklist and data collected for metrics are sent to the company knowledge base. That information is stored, in order to be used to help better evaluate future projects.

The implementation procedure is summarized in the figure 5.



Checklist
Metrics
Possible actions
Checklist is filled in, based on the situation during first iterations
Identify team benchmarks for metrics
- Improve project context
Metrics are measured and compared between iterations
- Adjust project estimates
- Report project development
Checklist is compared to those of other projects
Metrics are compared between different projects
- Objectively assess project success
- Report project results
Knowledge base
Beginning of the project
During the project
At the end of the project
Next project

Figure 5 Implementation procedure summary

CONCLUSION

Strive for continuous improvement of all processes is one of the vital components of the success in the modern, ever-changing world. In IT project management, it first led to the transition from strict and "heavy" project management methodologies towards those that are flexible and lightweight. This transition was accelerating during past several years, making agile a new dominant project management methodology in the information technology sector.

Scholars didn't leave the rise of agile unnoticed. Recently, a lot of research on the topic was published with majority of papers being focused on guiding the decision-making process of agile adoption. As more players adopt agile, tracking and control of the methodology becomes increasingly important for sustaining a competitive advantage. Despite that, few papers covered this aspect and those that did, were mostly based on the case studies. There was no unified approach for evaluating the agile IT project success and latest industry surveys showed that organizations were perceiving this issue as one of the main challenges in achieving their goals with methodology.

The goal of this study was to fill the gap by developing a unified, prioritized quantitative approach to measuring the success of Agile IT projects. To reach this goal, two research questions were formulated: What parameters should be measured in order to differentiate a successful Agile IT project, from the one that is not? What measuring techniques can be used to assess these parameters?

As project success evaluation appeared to be not a straightforward process, due to several parties, time scales and perspectives involved, it was first decided to develop a holistic view of project success. As a result, a matrix comprised of success criteria and critical success factors that affect these criteria was created.

In order to find appropriate measurement techniques for every component of the success matrix, it was needed to see what tools contemporary organizations already use for tracking and control. A unified list of metrics used in organizations was created based on various sources, including those from the most up to date academic papers and industry surveys. It turned out that metrics cover success criteria really well while are unsuitable for measuring the CSFs.

A checklist was proposed as a tool for recording statuses of CSFs. As checklist is a prioritized set of items, it was needed to prioritize CSFs. Fuzzy Analytic Hierarchy Process was chosen, as it allowed to incorporate experts' opinion, while accounting for the uncertainty of human thought. Prioritized list of CSFs was used for the checklist composition, however it also revealed several interesting moments that will be also discussed in the next paragraphs.

In terms of **scientific contribution**, there are several aspects worth mentioning. Firstly, the fact that CSF, related to Stakeholder satisfaction, that was added as a part of this work turned out to be perceived as relatively more important than half of the CSFs. It can act as a further confirmation that Stakeholder Satisfaction plays an important role in the project success assessment and should be considered. Secondly, a unified approach to project success evaluation was created. It can act as a base for the future research e.g. on different time scales. Lastly, results of fuzzy AHP reveal that industry specific CSFs are thought not to play a major role in project success. This means that project success evaluation approach proposed by this paper could be adopted to other industries in the future research.

Practical contribution can be also split into several parts or "applications". Firstly, implementation of the proposed tracking and control methodology provides an ability to evaluate project success more objectively. Secondly, checklist and list of metrics allow to recognize potential deviations and take actions to counter them. Lastly, methodology proposed in this paper simplifies reporting to external stakeholders, which is extremely important as agile is facing more risk-averse public in larger enterprises.

REFERENCES

Alliance, A. (2001). Agile manifesto. *Online at http://www.agilemanifesto.org*, *6*(1).

Alliance, S. (2015). The 2015 State of Scrum Report. *Online at www. scrumalliance. org/why-scrum/state-of-scrumreport/2015-state-of-scrum*.

Abbas, N., Gravell, A. M., & Wills, G. B. (2010, August). The impact of organization, project and governance variables on software quality and project success. In *AGILE Conference, 2010* (pp. 77-86). IEEE.

Beck, K. (1999). Embracing change with extreme programming. *Computer*, *32*(10), 70-77.

Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., ... & Kern, J. (2001). Manifesto for agile software development.

Benington, H. D. (1983). Production of large computer programs. *Annals of the History of Computing*, *5*(4), 350-361.

Boehm, B. W., & Ross, R. (1989). Theory-W software project management principles and examples. *IEEE Transactions on Software Engineering*, 15(7), 902-916.

Bozbura, F. T., Beskese, A., & Kahraman, C. (2007). Prioritization of human capital measurement indicators using fuzzy AHP. *Expert Systems with Applications*, *32*(4), 1100-1112.

Calabrese, A., Costa, R., & Menichini, T. (2013). Using Fuzzy AHP to manage Intellectual Capital assets: An application to the ICT service industry. *Expert Systems with Applications*, *40*(9), 3747-3755.

Chang, D. Y. (1996). Applications of the extent analysis method on fuzzy AHP. *European journal of operational research*, *95*(3), 649-655.

Cebeci, U. (2009). Fuzzy AHP-based decision support system for selecting ERP systems in textile industry by using balanced scorecard. *Expert Systems with Applications*, *36*(5), 8900-8909.

Chow, T., & Cao, D. B. (2008). A survey study of critical success factors in agile software projects. *Journal of systems and software*, 81(6), 961-971.

Cockburn, A. (2004). *Crystal clear: a human-powered methodology for small teams*. Pearson Education.

Cooke-Davies, T. (2002). The "real" success factors on projects. *International journal of project management*, *20*(3), 185-190.

Dybå, T., & Dingsøyr, T. (2008). Empirical studies of agile software development: A systematic review. *Information and software technology*, 50(9), 833-859.

Edmonds, E. A. (1974). A process for the development of software for non-technical users as an adaptive system. *General Systems*, *19*(215C18), 8.

Greening, D. R. (2015, January). Agile Enterprise Metrics. In *System Sciences (HICSS), 2015 48th Hawaii International Conference on* (pp. 5038-5044). IEEE.

Hales, B. M., & Pronovost, P. J. (2006). The checklist—a tool for error management and performance improvement. *Journal of critical care*, *21*(3), 231-235.

Hastie, S., & Wojewoda, S. (2015). Standish Group 2015 Chaos Report.

Hewlett Packard Enterprise Development Company LP. (2016). Capitalize on current IT trends.

Hummel, M. (2014, January). State-of-the-art: A systematic literature review on agile information systems development. In *System Sciences (HICSS), 2014 47th Hawaii International Conference on* (pp. 4712-4721). IEEE.

Huo, M., Verner, J., Zhu, L., & Babar, M. A. (2004, September). Software quality and agile methods. In *Computer Software and Applications Conference, 2004. COMPSAC 2004. Proceedings of the 28th Annual International* (pp. 520-525). IEEE.

Javdani, T., Zulzalil, H., Ghani, A. A. A., Sultan, A. B. M., & Parizi, R. M. (2013). On the current measurement practices in agile software development.

Jørgensen, M., & Moløkken-Østvold, K. (2006). How large are software cost overruns? A review of the 1994 CHAOS report. *Information and Software Technology*, *48*(4), 297-301.

Jugdev, K., & Müller, R. (2005). A retrospective look at our evolving understanding of project success. Project Management Institute.

Keil, M. (1995). Pulling the plug: Software project management and the problem of project escalation. MIS quarterly, 421-447.

Kumar, G., & Bhatia, P. K. (2012). Impact of Agile Methodology on Software Development Process. *International Journal of Computer Technology and Electronics Engineering (IJCTEE)*, *2*(4).

Kupiainen, E., Mäntylä, M. V., & Itkonen, J. (2014, June). Why are industrial agile teams using metrics and how do they use them?. In *Proceedings of the 5th International Workshop on Emerging Trends in Software Metrics* (pp. 23-29). ACM.

Kupiainen, E., Mäntylä, M. V., & Itkonen, J. (2015). Using metrics in Agile and Lean Software Development–A systematic literature review of industrial studies. *Information and Software Technology*, *62*, 143-163.

Lee, A. H., Chen, W. C., & Chang, C. J. (2008). A fuzzy AHP and BSC approach for evaluating performance of IT department in the manufacturing industry in Taiwan. *Expert systems with applications*, *34*(1), 96-107.

Leon, A., & Koch, A. S. (2004). *Agile software development evaluating the methods for your organization*. Artech House, Inc..

Lindstrom, L., & Jeffries, R. (2004). Extreme programming and agile software development methodologies. *Information systems management*, *21*(3), 41-52.

Lindsjørn, Y., Sjøberg, D. I., Dingsøyr, T., Bergersen, G. R., & Dybå, T. (2016). Teamwork quality and project success in software development: A survey of agile development teams. *Journal of Systems and Software*, *122*, 274-286.

One, V. (2015). *9th annual state of agile survey*. Technical report, Version One.

Padmini, K. J., Bandara, H. D., & Perera, I. (2015, April). Use of software metrics in agile software development process. In *Moratuwa Engineering Research Conference (MERCon), 2015* (pp. 312-317). IEEE.

Petersen, K., & Wohlin, C. (2010). The effect of moving from a plan-driven to an incremental software development approach with agile practices. *Empirical Software Engineering*, *15*(6), 654-693.

Petersen, K., & Wohlin, C. (2011). Measuring the flow in lean software development. *Software: Practice and experience*, *41*(9), 975-996.

Petersen, K. (2012, May). A palette of lean indicators to detect waste in software maintenance: A case study. In *International Conference on Agile Software Development* (pp. 108-122). Springer Berlin Heidelberg.

Project Management Institute. (2013). *A Guide to the Project Management Body of Knowledge (PMBOK guide)*, 5th Edition.

Racheva, Z., Daneva, M., Sikkel, K., & Buglione, L. (2010). Business value is not only dollars–results from case study research on agile software projects. In *Product-Focused Software Process Improvement* (pp. 131-145). Springer Berlin Heidelberg.

Rahmanian, M. (2014). A comparative study on hybrid IT project management. *International Journal of Computer and Information Technology*, *3*(05), 1096-1099.

Rosser, L., Marbach, P., Lempia, D., & Osvalds, G. (2014). Systems Engineering for Software Intensive Projects Using Agile Methods. *International Council on Systems Engineering IS2014, Las Vegas, US-NV*, *30*.

Saaty, T. L. (1980). The analytic hierarchy process: planning, priority setting, resource allocation.

Schwaber, K. (1997). Scrum development process. In *Business Object Design and Implementation* (pp. 117-134). Springer London.

Serrador, P. and Turner, R. (2015). The Relationship Between Project Success and Project Efficiency. *Project Management Journal*, 46(1), pp.30-39.

Serrador, P. and Pinto, J. K. (2015). Does Agile work?—A quantitative analysis of agile project success. *International Journal of Project Management*, *33*(5), 1040-1051.

Shenhar, A. J., & Dvir, D. (2007). Project management research-the challenge and opportunity. *Project management journal*, *38*(2), 93.

Staron, M., Meding, W., & Söderqvist, B. (2010). A method for forecasting defect backlog in large streamline software development projects and its industrial evaluation. *Information and Software Technology*, *52*(10), 1069-1079.

Turner, R. and Zolin, R. (2012). Forecasting Success on Large Projects: Developing Reliable Scales to Predict Multiple Perspectives by Multiple Stakeholders Over Multiple Time Frames. *Project Management Journal*, 43(5), pp.87-99.

van Kelle, E., van der Wijst, P., Plaat, A., & Visser, J. (2015, May). An empirical study into social success factors for agile software development. In *Proceedings of the Eighth International Workshop on Cooperative and Human Aspects of Software Engineering* (pp. 77-80). IEEE Press.

Vinodh, S., Balagi, T. S., & Patil, A. (2016). A hybrid MCDM approach for agile concept selection using fuzzy DEMATEL, fuzzy ANP and fuzzy TOPSIS. *The International Journal of Advanced Manufacturing Technology*, *83*(9-12), 1979-1987.

Zadeh, L. A. (1965). Fuzzy sets. *Information and control*, *8*(3), 338-353.

APPENDICES

Appendix 1 Final list of metrics

Metric	Definition
Budget vs. actual cost	Comparison between
Build status	Build broken or not
Burn-up chart	Graphical representation
Check-ins per day	Number of commits (code, automated test, specification) per day
Cost types	Not defined in primary study
Cumulative flow chart	Not defined
Customer/user satisfaction	Customer buy-in in %
Cycle time	Time it takes for x size story to be completed
Cycle time Estimation accuracy	Difference between planned and actual cycle time
Defect count	Number of defects
Defect resolution	Average time it takes to resolve a defect
Defect trend indicator	Indicates if amount of defects in the coming week will increase, stay the same or decrease from this week
Defects into production	Not defined
Deferred defects	Probably means a number of defects that are known but are not fixed for the release.
Downstream impact	In case of team interdependencies, the number of downstream teams affected
Effort At Risk	Sum of rejection impact ratios per PBI
Effort estimate	Estimated effort per story in ideal pair days
Enhancement Rate	% of items from the last backlog included to the production in the next backlog
Estimation Shift	Positive or negative % change in estimation of an iteration
Expenses Prognosis	Extrapolation of expected product costs
Focus factor	Velocity ÷ Work Capacity
Forecast Horizon	The point in the Product Backlog just before the first unestimated Product Backlog Item
Individual hours per iteration/week	Not defined
Iteration burndown	Graphical representation of the work completed and left in the iteration
Lead time	The average time it takes for one request to go through the entire process from start to finish
Number of test cases	Not defined in primary study
Number of unit tests	Number of unit tests
Percentage of adopted work	∑(Original Estimates of Adopted Work) ÷ (Original Forecast for the Sprint)

Metric	**Definition**
Percentage of found work	∑(Original Estimates of Found Work) ÷ (Original Forecast for the Sprint)
Planned vs. actual release dates	Difference between planned and actual release dates
Planned vs. actual stories per iteration	Not defined
Predicted number of defects in backlog	Predicted number of defects in backlog in the coming week.
Priority Shift	Sum of difference (shifts) in priorities
Progress as working code	Product is demonstrated to the customer who then gives feedback
Project Size	Sum of size estimates or quantity of PBIs
Queue time	The average time between sub-processes that the request sits around waiting
Rejected PBIs	Number of rejected PBIs
Release burndown	Not defined
Scope change in a release	Sum of work added or removed from the release
Software Quality	Measured, using quality properties described in ISO/IEC 25010
Story flow percentage	Estimated implementation time per actual implementation time *100
Story percent complete	Not clearly defined in primary study
Technical debt	Technical debt in amount of hours it would take to fix all the issues
Test coverage	Percent of code covered by tests
Test pass/fail over time	Not defined
Test-growth ratio	Difference of amount of tests per difference of amount of Source Code.
Variance in handovers	Changes in amount of handed over requirements
Velocity	Amount of developed scenarios per developer per week
Velocity deviation	Deviation of velocity from sprint to sprint
Velocity of elaborating features	Probably the time it takes to clarify a feature from customer into requirements that can be implemented
Work-in-Process (WIP)	Amount of features or feature level integrations team is working on

Appendix 2 Aggregated Pairwise-comparison matrix

ALL DECISION MAKERS	TE			TC			CI			PMP			ASE			DS			EBD		
Team environment	1.0	1.0	1.0	0.2	0.2	0.3	1.4	1.7	2.0	3.7	4.2	4.7	1.4	1.6	1.9	2.2	2.7	3.2	1.5	1.8	2.2
Team capability	5.5	6.3	7.1	1.0	1.0	1.0	2.4	2.8	3.1	3.4	3.8	4.3	3.6	4.2	4.8	3.1	3.7	4.2	2.4	3.0	3.6
Customer involvement	4.2	4.6	5.1	2.7	3.3	3.9	1.0	1.0	1.0	3.6	4.1	4.5	2.8	3.3	3.8	4.0	4.5	5.1	1.9	2.4	2.9
Project management process	1.7	2.1	2.5	1.9	2.3	2.8	2.0	2.3	2.7	1.0	1.0	1.0	1.7	2.1	2.6	0.2	2.3	2.8	2.7	3.2	3.6
Agile software engineering techniques	2.9	3.5	4.2	1.5	1.9	2.3	2.0	2.3	2.7	2.7	3.0	3.4	1.0	1.0	1.0	2.5	3.0	3.6	1.7	2.2	2.7
Delivery strategy	3.2	3.7	4.2	2.3	2.7	3.1	1.2	1.4	1.7	3.1	3.6	6.6	1.4	1.8	2.2	1.0	1.0	1.0	0.3	0.4	0.4
Effective benefits delivery and co-operative management process	3.8	4.3	4.8	2.5	2.9	3.4	2.3	2.6	2.9	2.4	2.9	3.3	2.7	3.2	3.7	4.1	4.8	5.5	1.0	1.0	1.0