Scope Creep or Embrace Change? A Survey of the Connections Between Requirement Changes, Use of Agile, and Software Project Success

Abstract

Traditionally, a high degree of requirement change has been considered harmful for the success of software projects. Software professionals who use agile software development methods tend to view this topic differently. They tend to view requirement changes more as opportunities, which should be welcomed. Possibly, both views are correct but valid in different software development contexts. This paper aims at increasing the understanding of the connections between the degree of requirement change, choice of development method, and project success. Seventy software professionals were asked to provide information about their last software project. A higher degree of requirement changes, here defined as more than 30% of the requirements added, deleted, or changed during the project's execution, was connected with a higher proportion of successful projects in an agile development context, but only when this included frequent deliveries to production. Our results consequently support that the agile claim of "embrace change" has merit, but only in agile contexts.

1. Introduction

When software professionals are asked what they consider the main risk factors of software projects, they tend to include factors related to the requirement specifications. The survey reported in [1] is a good illustration. In that survey, the respondents ranked "misunderstanding the requirements" the second most important risk factor, "lack of frozen requirements" the sixth most important risk factor, and "changing scope/objectives" the seventh most important risk factor. Ranking incomplete and changing requirement specifications as important risk factors is in accordance with the traditional view of software development and requirement engineering. This view typically considers a requirement specification as consisting of "a set of system requirements which, as far as possible, is complete, consistent, relevant and reflects what the customer actually wants" [2].

Some software professionals seem to have different views on changed requirements. Those who use agile development methods recommend, among others, valuing "responding to change over following a plan,"[1] and to promote the principle of "welcome changing requirements, even late in development."[2] They also seem to think of requirement changes during the project's execution as opportunities to increase client values rather than as threats to the success of the project [3]. This corresponds with the observation that agile methods to some extent are designed for flexibility in scope and frequent requirement changes, e.g., as implemented in the common agile practice of flexible scope and frequent deliveries to client with opportunities for feedback and learning during the project execution.

The study reported in this paper tries to shed some light on the connection between requirement changes, development methods, and project outcomes. This include the goal of examining whether both viewpoints could be right, that is, that many requirement changes are connected with better outcomes for agile software projects, but worse outcomes for non-agile software projects. The main research questions are:

RQ1: How is the connection between amount of requirement changes and project outcome dependent on the development method?

RQ2: Among agile software projects, is there a difference in the connection between amount of requirement change and project outcome for project with and without frequent delivery to clients?

The second research question is motivated by our previous research, see [4], where frequent delivery to client were found to be one of the practices with strongest connection to project success.

The remainder of this paper is organized as follows: Section 2 describes selected related work on the effect of requirement changes, Section 3 describes the design and the results of the survey, Section 4 discusses the results and concludes.

2. Related work

A study by Serrador and Pinto [5], which examined 1002 software projects, suggests that the

[1] www.agilealliance.org/agile101/the-agile-manifesto/. Retrieved May 22, 2018.
[2] www.agilealliance.org/agile101/12-principles-behind-the-agile-manifesto/. Retrieved May 22, 2018.

most successful projects were those with most effort spent on specifying the requirements before the projects were initiated. The survey, and review, paper [6] reports "functional, performance, and reliability requirements and scope are not documented" as the second most important software project risk factor. A survey of software managers reports that they considered requirement volatility among the top software failure risk factors [7].

The project survey reported in [8] finds a negative correlation between requirement changes and cost control. Similarly, the study in [9] reports a negative effect of requirement changes on product performance, measured as system reliability, ease of use, ability to meet users' requirements, and user satisfaction. The same study also reports a negative effect of requirement changes on project performance measured as budget and schedule control.

The survey of software projects reported in [10], which examined the connection between increases in the requirement scope and the degree of client satisfaction with the project, found that a large requirement increase was connected with more project failures for traditional projects but not for agile projects. Although this finding is highly relevant for the study in this paper, and indicates that the choice of development method matters for the effect of requirement changes on project outcomes, the study had limitations. The traditional projects, on average, were much larger (and were likely to be more complex) and had a higher number of requirement changes than the agile projects. The difference in how requirement changes and client satisfaction were connected, therefore, could be a result of factors other than the choice of development method.

A survey of 399 agile software projects [11] reports that agile teams' ability to respond to requirement changes, measured as the proportion of change requests implemented (response extensiveness) and the speed (response efficiency), was positively connected to the ability of the software functionality delivered to meet the requirements, achieve goals, and satisfy users. A high response extensiveness had no large effect on the other project success dimensions, suggesting that responding to additional requirement changes was connected to better client satisfaction and benefits, without harming the other project success measures.

The survey reported in [4] found that agile projects with a flexible scope had almost twice as high a success rate as agile projects without a flexible scope. This result may be interpreted as supporting the benefit of adopting the agile principle of welcoming change.

An inherent problem in studying requirement specifications and requirement volatility is that we do not have commonly accepted and easy-to-implement measures of the size and complexity of a requirement change, the types of requirement changes, or the degree of change of a requirement specification [12]. The negative, or positive, consequences of a requirement change may depend, for example, on whether the change is only minor or leads to a large amount of rework, whether due to improved insight into client needs or external changes, and whether the change appears early or late in the project.

The great majority of previous research results, as far as we can see, suggest that more requirement changes are connected with more problematic and less successful software development. However, most of the research was conducted in a non-agile software development context. Therefore, whether the "embrace change" claim made by agile software professionals has some merit remains unproved. This is in particular the case, taking into account that more recent studies [4, 10, 11] give some hope for positive effects of requirement changes in the context of agile projects.

3. The survey

3.1. Design

The survey requested the participants, who were project managers and software developers from different organizations participating in a seminar on software cost estimation, to provide information about their last completed software projects with budgets of more than €100,000. Seventy-five responses were received. Five of the responses were incomplete, i.e., included "don't know" responses, and therefore removed, leaving 70 complete responses. Each response included information about the following:

- The respondent's role (free text) and length of experience (years).
- The budget category of the project: €100,000–1 million[3], €1–10 million, >€10 million.
- The type of development method used: Agile, Waterfall/Traditional, Mixed/other.
- Frequency of completed software functionality delivered to production or to user evaluation with feedback (this variable was included based on the results in [4], where delivery frequency was an essential variable for success with agile projects): None, 1–4 per year, More than 4 per year.
- Percentage of requirements added, removed, and/or updated: 0–10%, 10–30%, More than 30%.
- Reasons why the requirements were added, removed, and/or updated:[4] Learned about client

[3] The original questions were in Norwegian and used Norwegian currency. The budget values are approximate monetary values assuming that EUR 1 = NOK 10.

needs or gained insight during the project execution, External changes, Insufficient requirement analysis before the project started, Other reasons.
- Perceived project outcome (for each of the project success dimensions below, the respondent was requested, based on his/her evaluation, to choose one of the outcome categories: Very successful – Successful – Acceptable – Problematic – Very problematic): Client benefits, Technical quality of software, Cost control, Time control, Work efficiency.

We categorized the total performance (outcome) of a project as follows:
- Successful: The project was evaluated as very successful or successful on all five success dimensions (client benefits, technical quality of software, cost control, time control, and work efficiency)
- Acceptable: The project was not successful but was evaluated as at least acceptable on all five success dimensions.
- Problematic: The project was evaluated as problematic or very problematic on at least one of the success dimensions.

In the analysis section, we mainly present analyses based on the proportion of successful and problematic projects. The proportion of acceptable projects can be derived from the proportion of successful and problematic projects.

Due to few responses for some of the categories, we decided to join the categories "Mixed/other" and "Waterfall/traditional", creating the category "Non-agile". This gives very rough development method categories, but enables a comparison of what was considered agile by the respondents with the other projects. Similarly, the few responses with less than 10% requirement changes led us to join this category with the 10-30% category. The choice of 30% as our boundary value is to some extent arbitrary, but hopefully useful to gain some insight into difference of project with much and with less requirement changes.

3.2. Limitations

When interpreting the survey results, the following limitations should be kept in mind:
- The sample of respondents and their projects is not necessarily representative of other contexts. While this may strongly affect the characteristics of the data set, it may have less impact on the connections we focus on in this study. There is clearly a need for more studies to assess the generality and context dependencies of the results identified.
- The survey asked for the perceived (subjective) performance related to the success dimensions and did not use more objective measures of the project outcome, what they meant by use of agile or non-agile development methods, and a requirement change. Although this makes the evaluations highly subjective, and there will be differences in use of terms among the respondents, it may also have advantages. It may be, for example, that delivering the software one month late is acceptable in one project context but leads to large problems in another context. Mechanical evaluations of measured time overrun may not enable such meaningful distinctions.
- The respondents (34% were project managers or team leaders and 66% were software developers) were all from the provider side of the projects. This may have affected the assessment of the projects' success. The results of a similar survey (see [4]) found, however, that providers and clients tend to give similar evaluations of software projects, even when evaluating the client benefits. In addition, a role bias is mainly a problem for the main (interaction) analyses in this paper if the role bias is different for different development methods, which we believe is not the case.
- The number of responses is low for the interaction analyses of this paper, especially for non-agile projects. This limits the robustness of the results, excludes the use of tests for statistical significance, and points to the need for follow-up studies to validate the findings.

There is no guarantee that the respondents had the required information about the project, even though they chose to respond and had the option of leaving questions unanswered or using the don't know category. The respondents' experience, which, on average, was 14 years (only 6 respondents had less than 4 years of experience), gives some confidence that they were sufficiently competent to possess the required information.

3.3. Results

As can be seen in Table 1, 43% of the projects had more than 30% requirement changes (inserted, removed or updated requirements) during project execution. On average, the projects with more than 30% requirement changes were somewhat more successful (27% of them were successful) than those with less than 30% requirement changes (18% of them were successful). The projects with more than 30% requirement changes were also, on average, slightly less problematic (33% of them were problematic) than those with less than 30% changes

[4] It was possible to give more than one reason for the requirement changes. Twenty-five percent of the respondents did this.

(37% of them were problematic). This not substantial difference in project outcomes related to requirement changes hides, however, a large difference when the development method is included as an interacting variable (see Fig. 1 and Fig. 2).

Table 1. Project characteristics

Variable	Characteristics
Project size	58% less than €1 million 33% more than €1 million 9% larger than €10 million
Development method	74% agile 26% non-agile
Delivery frequency	36% 4 or fewer per year 64% more than 4 per year
Requirement changes	57% less than 30% 43% more than 30%
Reason for change (more than one reason possible)	78% learning/insight 16% external change 27% insufficient up-front analysis
Project outcome	25% successful 39% acceptable 36% problematic

Fig. 1 shows that the proportion of successful projects increased (from 15% to 31%) with more requirement changes for agile projects but decreased (from 25% to 0%) for non-agile projects. Notice that the proportion of successful non-agile projects is higher than that of the agile projects when there are fewer than 30% requirement changes but substantially lower when there are more requirement changes. There were only four non-agile projects with more than 30% requirement changes, which means that we should interpret the decrease in the success rate for the non-agile projects with great care. Previous research (see Section 2), however, supports a decrease in the success rate for non-agile software projects with many requirement changes. The results for non-agile projects with many requirement changes, although based on very few observations, therefore, are in accordance with some previous results.

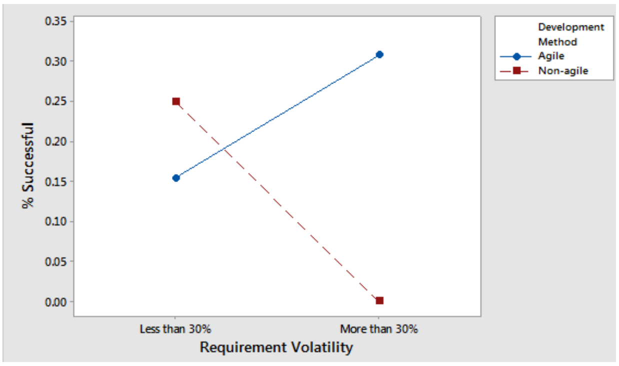

Figure 1. Development methods, requirement change, and proportion of successful projects.

Fig. 2 shows a weak decrease (from 31% to 27%) in the proportion of problematic projects with more requirement changes for agile projects. The corresponding observation for non-agile projects is an increase (from 50% to 75%) in the proportion of problematic projects. As before, the number of non-agile projects with more than 30% requirement changes are few, and the results for non-agile projects with many requirement changes, consequently, are not very robust.

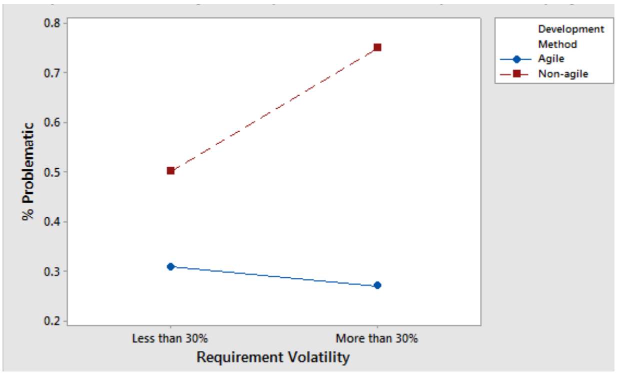

Figure 2. Development methods, requirement change, and proportion of problematic projects.

Table 2 shows the proportion of projects evaluated as "very successful" or "successful" for each of the success dimensions, development methods, and requirement change categories. The data suggest that the use of agile development methods is connected with an increase in the proportion of successes from less than 30% to more than 30% requirement changes for all success dimensions, but especially for the success dimensions technical quality (72% - 48% = 24% point increase) and cost control (50% - 38% = 12% point increase).

Table 2. Proportion projects evaluated to be "successful" or "very successful" for each success dimension, development method, and requirement change category

Requirement change		Less than 30% change		More than 30% change	
Development method		Agile	Non-agile	Agile	Non-agile
Success dim.	Client benefits	77%	54%	85%	25%
	Technical quality	48%	42%	72%	0%
	Cost control	38%	50%	50%	0%
	Time control	46%	50%	54%	0%
	Work efficiency	62%	42%	67%	50%

In total, answering our RQ1, the results displayed in Fig. 1, Fig. 2 and Table 2 suggest that there is an interaction effect from development method on the connection between requirement change and project outcome. The agile software projects performed better in contexts with more requirement changes, while the opposite was the case for the non-agile projects.

Motivated by the results in [4], and answering RQ2, we expected to see a difference in the success rate between agile projects with many (more than four per year) and with fewer (four or fewer per year) deliveries of completed software functionality to production or to user evaluation. This is what we see in Fig. 3 and Fig. 4.

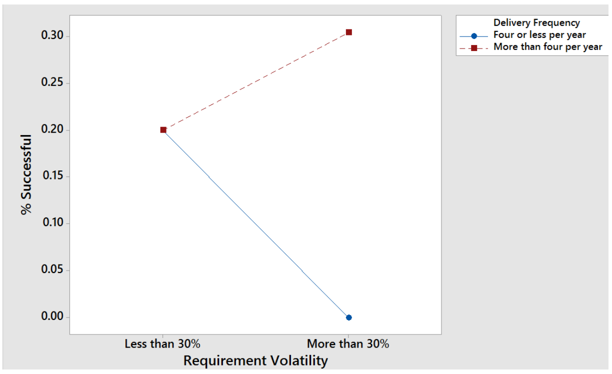

Figure 3. Delivery frequency, requirement change, and success for agile projects.

As can be seen in Fig. 3 and Fig. 4, the agile projects were more successful and less problematic in contexts with many requirement changes when the projects had frequent deliveries (more than 4 per year) to production or proper user evaluation of completed functionality. Frequent delivery did, however, not make any difference in the project's success rate and gave only a slightly lower rate of problematic projects when there were fewer requirement changes. To what extent frequent deliveries to production, with feedback, causes more requirement changes, leads to project success in situations with more requirement changes, or indicates a development context with success inducing elements, such as more involved clients, is hard to see from the data. This is another topic for future examination.

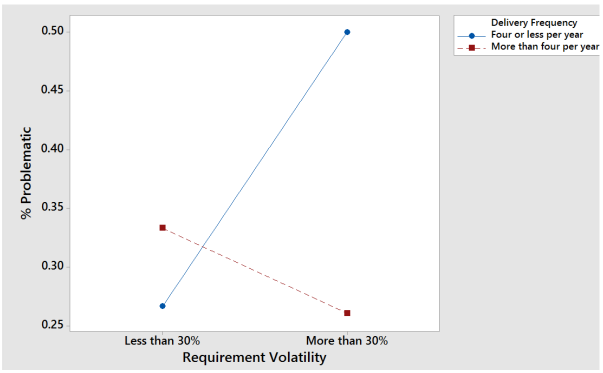

Figure 4. Delivery frequency, requirement change, and problems for agile projects.

Requirement changes may differ considerably in complexity, implications for rework, and how much the changes disrupt the project execution. As an initial step in understanding the influence of the type of requirement change on the project performance, we examined the effect of many requirement changes on project performance for the three reasons (learning or better insight, external changes, and insufficient requirement analysis) individually. The results are displayed in Table 3. We include only the results for the agile projects, because there were too few observations to give similar, meaningful results for non-agile projects.

Table 3. Success and failure rate, per reason (agile projects only)

Req. change	Less than 30% changes			More than 30% changes		
Reason	Le	Ex	In	Le	Ex	In
Success	20%	0%	0%	30%	17%	20%
Accept.	55%	33%	50%	33%	50%	20%
Problem.	25%	67%	50%	22%	50%	60%

[i] Le = Learning/insight, Ex = External, In = Insufficient analysis

The data in Table 3 do not reveal a clear pattern connecting the reasons for and the degree of requirement changes. The proportion of successful projects increased and the proportion of problematic projects decreased with more requirement changes for all requirement change reasons. Notice, however, the higher problem rates for agile projects where the requirement changes were categorized as externally induced or caused by insufficient analysis compared to when the requirement changes were categorized as caused by learning or better insight.

Contextual differences may explain the differences in how the requirement changes, development method, and project performance are connected. Many important contextual variables were not collected, such as how late the requirement change occurred and the skill of the development team. It might nevertheless be interesting to examine if there are essential differences between agile and non-agile projects based on the data we collected; see Table 4. The values related to "Reasons for changes" are the proportion of projects where the reason was believed by the respondent to have caused all or part of the requirement changes, if any, in the project. None, one, or more reasons could be provided for the same project.

Table 4. Context differences between agile and non-agile

Characteristic	Measure or category	Development method	
		Agile	Non-agile
Respondent's experience	Mean length of experience (years)	13	15
Budget size	Proportion costing less than €1 million	62%	44%
	Proportion costing more than €1 million	38%	56%
Requirement change	Proportion with less than 30% change	50%	76%
	Proportion with more than 30% change	50%	24%
Reason for changes	Proportion due to learning/insight during project execution	82%	67%
	Proportion due to external changes	25%	11%
	Proportion due to insufficient requirement analysis	22%	44%

As can be seen, there were fewer, but not substantially fewer, agile projects (38% vs. 56%) in the category of projects with a budget of more than €1 million, more agile projects (50% vs. 24%) in the category of projects with more than 30% requirement changes, and for agile projects, respondents were more likely to provide the requirement reasons "learning/insight" (82% vs. 67%) and "external changes" (25% vs. 11%) and less likely to give the reason "insufficient requirement analysis" (22% vs. 44%). There were no large differences in the average length of respondents' experience for agile and non-agile software projects (13 vs. 15 years). The directions of the contextual differences shown in Table 4 are not surprising. Agile development methods are more commonly used for smaller projects, agile projects receive more requirement changes, and agile software professionals are less likely to think about requirement changes caused by insufficient requirement analysis, given less emphasis on producing up front complete and detailed requirement specifications. The higher degree of externally induced requirement changes may indicate that agile methods were more frequently used in contexts with higher environmental (external factors-based) uncertainty. All these differences point at possible differences in development complexity, for example, slightly larger projects for non-agile and perhaps more requirement uncertainty for agile projects, which, in turn, may explain some of the observed differences in the project outcomes for agile and non-agile projects. There is, however, little that suggest that the identified differences in contexts, which are not very large, explain the reported differences in how well agile and non-agile software projects succeed in situations with much requirement changes.

4. Discussion and conclusion

Most software projects experience that requirements are added, removed, or changed during the project execution. In as much as 50% of the agile and 24% of the non-agile projects included in our survey, more than 30% of the requirements were added, removed, or updated during the project execution. Requirement changes may be viewed as a threat or as an opportunity. Traditionally, requirement changes have been viewed as a risk factor, that is, a threat to the success of a software project. Agile software developers, however, tend to view requirement changes differently. They tend to view changes as creating opportunities to deliver more client benefits, and view them as something that should be welcomed in software projects.

The present results provide support for both views. When agile methods were used, but only when used with frequent deliveries of completed functionality to productions or user evaluation, many requirement changes were connected to higher proportions of successful projects and lower proportions of problematic projects. For non-agile projects and agile projects without frequent deliveries to production, the outcome was the opposite. Many requirement changes for such projects were connected to less successful and more problematic projects.

The connection examined in this paper, that is how the development method influences the connection between requirement changes and software project success, has not been much investigated empirically. The only previous study we were able to identify is the one reported in [10]. As reported in Section 2, that study found a positive connection between a large increase in requirements and more satisfied clients for agile but not for non-agile software projects. Although limited to added requirements, i.e., not including changed requirements, and using client satisfaction as the only success measure, this result is consistent with what we found.

In the present study, non-agile projects (see Table 4) were larger than agile projects but not by much, and we believe the difference is not large enough to explain the differences in project outcomes. Indeed, we found larger projects to be somewhat more successful and less problematic (33% successful and 30% problematic projects) than smaller projects (21% successful and 39% problematic projects).

The limited number of variables and observations in this study means that we were unable to gain much insight into the underlying mechanisms that created the difference in project performance for different levels of requirement changes and different development methods. We cannot, as discussed in Section 3, be sure that the observed differences between successful and problematic projects were caused by, as opposed to just correlated with, differences in development method.

It is perhaps not surprising that a development method designed for flexibility in scope and frequent requirement changes, that is, the agile software development method, leads to better project outcomes than traditional, non-agile, methods when there are many requirement changes. What is perhaps more surprising is that projects following the agile method, when including the agile practice of frequent delivery to client, did *better* when there were *more* rather than fewer requirement changes. Currently, we find it hard to suggest mechanisms that should make it easier to succeed with more rather than fewer requirement changes. We suspect that the use of agile development methods, but mainly when implementing a practice with frequent deliveries to production, combined with many requirement changes correlates with the presence of other, essential success factors. This may include success factors related to more competent and involved clients, better and more frequent feedback and learning during project execution, better benefits management processes, more skilled developer teams, and better software testing facilities [13].

These interpretation challenges, together with the study limitations discussed earlier, mean that there is a need for more, carefully designed studies that not only try to replicate our results and examine the connections, but also try to better understand the context, patterns, and mechanisms that lead to the differences. This may be important in an evidence-based attempt to improve requirement management practices and project outcomes.

Changes in requirements are here to stay, and our ability to manage them is essential for success in software development. The present results provide some evidence in support of that agile development methods, when implementing frequent deliveries to production or to user evaluation with feedback, are a good choice when expecting many requirement changes.

5. References

[1] Schmidt, R., K. Lyytinen, and P.C. Mark Keil, *Identifying software project risks: An international Delphi study*. Journal of management information systems, 2001. **17**(4): p. 5-36.

[2] Sommerville, I. and P. Sawyer, *Requirements engineering: a good practice guide*. 1997: John Wiley & Sons, Inc.

[3] Erickson, J., K. Lyytinen, and K. Siau, *Agile modeling, agile software development, and extreme programming: the state of research*. Journal of database Management, 2005. **16**(4): p. 88.

[4] Jørgensen, M., *A survey on the characteristics of projects with success in delivering client benefits*. Information and Software Technology, 2016. **78**: p. 83-94.

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

[6] Kappelman, L.A., R. McKeeman, and L. Zhang, *Early warning signs of IT project failure: The dominant dozen*. Information systems management, 2006. **23**(4): p. 31-36.

[7] Tiwana, A. and M. Keil, *The one-minute risk assessment tool*. Communications of the ACM, 2004. **47**(11): p. 73-77.

[8] Zowghi, D. and N. Nurmuliani. *A study of the impact of requirements volatility on software project performance*. in *Software Engineering Conference, 2002. Ninth Asia-Pacific*. 2002. IEEE.

[9] Govindaraju, R., et al., *Requirement volatility, standardization and knowledge integration in software projects: an empirical analysis on outsourced IS development projects*. Journal of ICT Research and Applications, 2015. **9**(1): p. 68-87.

[10] Suma, V. and K. LakshmiMadhuri. *Influence of Scope Creep on Project Success: AComparative Study between*

Conventional ApproachVerses Agile Approach. in *IEEE International Conference on Advanced research in Engineering and Technology (ICARET)*. 2013.

[11] Lee, G. and W. Xia, *Toward agile: an integrated analysis of quantitative and qualitative field data on software development agility*. Mis Quarterly, 2010. **34**(1): p. 87-114.
[12] McGee, S. and D. Greer, *Towards an understanding of the causes and effects of software requirements change: two case studies*. Requirements Engineering, 2012. **17**(2): p. 133-155.

[13] Jørgensen, M., P. Mohagheghi, and S. Grimstad, *Direct and indirect connections between type of contract and software project outcome*. International Journal of Project Management, 2017. **35**(8): p. 1573-1586.