Software Quality – Traditional vs. Agile: an Empirical Investigation

Article · October 2016

DOI: 10.48550/arXiv.1610.08312

5 authors, including:

Mohamad Kassab
Pennsylvania State University
141 PUBLICATIONS **1,849** CITATIONS

SEE PROFILE

JooYoung Lee
Australian National University
70 PUBLICATIONS **913** CITATIONS

SEE PROFILE

Manuel Mazzara
Innopolis University
528 PUBLICATIONS **7,201** CITATIONS

SEE PROFILE

Software Quality – Traditional vs. Agile: an Empirical Investigation

Mohamad Kassab*†, JooYoung Lee*, Manuel Mazzara*, Giancarlo Succi*, Rasul Tumyrkin*
*Innopolis University, Russia
{m.kassab, j.lee, m.mazzara, g.succi, r.tumyrkin}@innopolis.ru
†Pennsylvania State University, Malvern, USA

***Abstract*—It is well known that the software process impacts the quality of the resulting product. There are also anecdotal claims that agile processes result in higher level of quality than traditional methodologies. However, still solid evidence of this is missing. This work reports in an empirical analysis of the correlation between software process and software quality with specific reference to agile and traditional processes. More than 100 software developers and engineers from 21 countries have been surveyed with an online questionnaire. We have used the percentage of satisfied customers estimated by the software developers and engineers as the main dependent variable. The results evidence some interesting patterns: architectural styles may not have a significant influence on quality, agile methodologies might result in happier customers, larger companies and shorter projects seems to produce better products.**

I. Introduction

Since the inception of Agile Methods there have been allegations that they increase quality and productivity. Moreover, as discussed by [9] there are also several empirical studies providing evidences that agile methods and also their individual practices, especially pair programming, may have beneficial effects especially on quality but also on productivity. However, as also [9] makes clear, there are empirical studies that provide opposite evidences too. All of this claims for some generalization of the findings. A possibility for generalization comes from statistical metanalysis [10]. However, metanalysis requires a comprehensive analysis of the original data, and such data is often not fully supplied in the studies.

An alternative approach consists in running a survey and collecting experiences from across a wide variety of contexts, in order to identify the most profitable directions. In the context of agile methods, this approach has been taken first by [34], surveying the impact of extreme programming on developers satisfaction and identifying a situation later widely accepted, that agile methods tend to positively impact the satisfaction of developers and to lower the turnover.

The latter approach is the one championed and followed in this paper. Since quality is *multidimensional property* of software systems, different aspects of it have been considered. Among the several proposals of aspects, we have followed the one by [4]: Availability, Interoperability, Modifiability, Performance, Security, Testability and Usability.

A questionnaire was prepared to gather the experience of software developers on the impact of agile methods on the mentioned aspects of software quality. It was then made available and advertised at major conferences and events from May 2015 to September 2015. 687 people from 37 countries accessed it and 103 people from 21 countries completed it to the end. The completion rate was 15% and the average time taken to complete the survey was 10 minutes. We also included the results of the partially completed responses. When respondents aborted the survey, they tended to do so on or near question 15, we speculate from survey fatigue.

The results of the questionnaire evidences that the adoption of agile methods improves all the aspects of software quality under consideration, often in a statistically significant way and with a strong effect size. Given the large and diverse sample of the respondents, these results have a high likelihood to be extensible to the overall population of software projects.

The paper is organized as follows. Sec. II provides the overall technical and scientific background of this paper. Sec. III presents the experimental design of the work. Sec. IV analyses the results of the questionnaire and sec. V discusses the overall outcomes. Eventually, Sec VI draws the conclusions and outlines the future research in this field.

II. Background

Comparison of Agile and traditional projects is not a new topic in Software Engineering. Debates started right after the Agile manifesto [6] had been submitted in order to improve Traditional methods. We describe the main characteristics of both approaches in the following subsections.

A. Traditional methodologies

Most of the important decisions are made during the stage of design and once the product is well designed the project can continue with the building phase which is supposed to be very predictable. Thus the stage of building the product only follows the"perfect" design of the system. Since their main feature is detailed planning followed by designing phase, they are useful when the project is said to be very large and the level of risk is high. Some projects which are estimated to last very long are often developed with traditional methodologies [26],[27],[33].

B. Agile methodologies

It had been developed as a reaction to the traditional methods. For many people the appeal of these agile methodologies is their reaction to the bureaucracy of the engineering

methodologies. The work in [12] contains a set of principles for software development in which requirements and solutions evolve through collaboration between self-organizing, cross-functional teams [8]. It promotes adaptive planning, evolutionary development, early delivery, and continuous improvement, and it encourages rapid and flexible response to change [32]. The result of all of this is that agile methods have some significant changes in emphasis from engineering methods.

C. Methodologies and Software Quality

Both methodologies have goals to achieve best quality and satisfy customers. The way how exactly they affect on quality is still mostly unknown. We tried to bring our view on how quality could be measured for different development methods. We have discussed differences of customers' satisfaction levels for agile vs waterfall projects in our previous paper [35], where generally customers have been more satisfied with results of agile methodology.

Before explaining our way of analyzing this issue we have discovered how other researches analyzed the impact of chosen methodology on software quality.

In [25], different methodologies in software development process are explored and the authors investigate the reasons why software industries shifted from Traditional RE to Agile RE. This research has performed comparative studies of different software development methodologies, when and where they can be used. The paper also gives introduction to agile software development methodology and how to apply them.

In [11], authors conduct a survey to provide evidence-based assessment of the level of agile adoption by software development organizations, in relation to the general profile of the respondents (country of origin, business sectors, roles, etc.). Then they compare the results with different types of practices followed, such as agile techniques adopted, team organization and communication techniques, and project management.

In addition, [31] compares software development processes using agile and traditional methods and proposes an incursion in the software development, from traditional to agile.

The main objective of [20] is to conduct an empirical study into the choice among the most popular Agile methodologies, Scrum, Extreme Programming and Kanban. Further, this paper provides for a comparative analysis among various agile software development methodologies. Survey results reveal higher adoption of Scrum based development in present-day software industry as compared to Extreme Programming and Kanban methodologies.

In [23] , the authors compare traditional requirements engineering approaches and agile software development. They analyzed commonalities and differences of both approaches. Their findings were that the RE phases is not very clear in agile: they are merged and repeated in each iteration. Also, documentation is a main difference between the RE process and agile methods: There is a lack of documentation in agile. Changes being traceable are regarded important in RE, and agile methods also manage the requirements well with index cards. Traditional RE does not have a good way of stakeholder involvement as agile methods: customers are only involved in early stages in RE while they are involved during the whole process under agile. The authors then recommended that a minimum of documentation should be ensured in an agile environment as a possible way for the agile to benefit from the RE methods. Scott Ambler [2] suggests a set of best practices for RE using agile methods. Many of the practices follow directly from the principles behind the agile Manifesto. Ambler also suggests using such artifacts as CRCs, acceptance tests, business rule definitions, change cases, dataflow diagrams, user interfaces, use cases, prototypes, features and scenarios, use case diagrams, and user stories to model requirements. These elements can be added to the software requirements specification document along with the user stories. For requirements elicitation he suggests using interviews, focus groups, JAD, legacy code analysis, ethnographic observation domain analysis, and having the customer on site at all times.

Bose, Kurhekar and Ghoshal [7] suggested that agile requirements are effective to get continuous feedback from customers while the limits of agile methods are still not defined well. They suggested that the requirements management should be adopted to ensure the traceability, which is important when the requirements are likely to be changed in an agile method. The verification of early requirements descriptions being included are also proposed. The literature is rich with other survey studies on RE practices, development needs, and preferred ways that target local geographic location (e.g. Finland [22], Chile [24], and Malaysia [30]).

In [21] the authors (Neill and Laplante) presented results of a comprehensive survey that was conducted in 2002 showing requirements engineering practices across a broad range of industries and projects types. The results were surprising in that they indicated, among other things, that the waterfall model was still widely used and that various techniques associated with agile development were not widely employed. Unpublished results from a similar survey in 2008 [18] indicated that the findings from 2002 had remained largely unchanged. The 2002 survey results were highly cited (181 times via Google Scholar), and seemed to provide a template for more focused requirements surveys. For example Khurum et al [17] conducted a brief survey to uncover challenges in organizations to effective requirements engineering, Chernak surveyed a small group of companies to determine the prevalence of requirements reuse and Verner et al [36] sought to uncover specific issues with respect to requirements management. In [15] , the author an empirical study based on a conducted survey on the requirements engineering practices for agile software development, and comparing these with the traditional waterfall.

III. Experimental Design

This work is a case study which is conducted to investigate how software process in place impacts the quality of the resulting product. To understand the relationship, we conducted a very large survey during the last year which was completed by more than 100 software developers and engineers from 21

countries. We use the goal-question-metrics (GQM) approach to organize the empirical data analysis [3].

A web-based survey instrument was created using the web-based QuestionPro survey tool (www.QuestionPro.com). The survey consisted of 19 questions. The survey questions were designed after a careful review to specialized literature on conducting survey studies (e.g. [14], [29], [19], [28], [13]). A summary of our survey questions is available via the link[1]. While the respondents reported a wide range of experiences; they were asked to base their responses on only one software project that they were either currently involved with or had taken part in during the past five years. Figure 1 shows the distribution of positions of participants.

Using the conjectures in our hypotheses as means of constructing specific questions, the survey was arranged into five sections: First section aiming at capturing general project characteristics first. Then, a series of questions were asked in the second section to determine the participants knowledge of architectural styles and whether if any were applied into the surveyed projects. In case of incorporating architectural styles into the projects; the respondents were then asked to report on the criteria they used to select these styles in the third section and the challenges they faced while incorporating them in the forth section. Since quality requirements are the major that shapes the software architecture [5]; a series of questions were then asked in the fifth section to report on the level of customers satisfaction with these qualities while the final product is in use.

We drew our survey participants from multiple sources but primarily from members of the following Linked-In professional groups, to which one or more of the authors belonged: "Software Engineering Productivity: Software Architecture", "Techpost Media","ISMG: Software Architecture" and "ISMG Architecture World". A invitation on these groups was posted under the subject "Software architecture in practice". The participation to this survey was entirely anonymous and voluntarily.

A. Structure of the data

In this analysis we take into consideration also the answers given by people who partially completed the questionnaire for statistical analysis not requiring pairing or correlating information, as suggested in [16]. Given this population, responses to the survey are more likely to reflect the opinions and biases of any given projects development team rather than those of other groups represented in a software development effort. In this section we will consider two aspects of respondents' profiles: *distribution of business* and *type of respondents*.

Distribution of business: The distribution of businesses that survey respondents have associated themselves with entails a lot of different fields. The data indicate that respondents are well distributed across a wide range of business domains. All fourteen of the provided domains have been selected at least by few participants. Furthermore, the *"other"* category included responses such as social media, transportation, automotive, virtualisation, meteo and etc.

Type of respondents: In order to understand the types of respondents, a number of questions regarding *"Organizational Characteristics"* have been asked. Respondents to this survey characterize themselves as programmers and developers 41% of the time, and software engineers, 17% percent of the time. One third of the respondents characterize themselves as architects and 9% percent as managers (project managers, scrum managers and product owners).Other respondents include system engineers, testers, consultants - reaching a total of around 3%.The majority (more than 36%) of respondents represent small companies, with an annual budget of less than 5 million US dollars, within the listed business domains (all company sizes are measured both, in terms of annual budget and number of employees). It is noticeable that about one third of respondents ignore the budget of their companies (this is consistent with the fact that developers often are not exposed to financial information).

B. Goal-Question-Metrics

In this work we follow the "Goal, Question, Metric" approach (GQM), an approach to software metric defined in [3]. We define the goal using the standard GQM template as follows:

- Analyzing *survey results data*.
- For the purpose of *evaluating it*.
- With respect to *relationships between software process and quality*.
- From the view point of *developers*.
- In the context of *industrial software development projects*.

The GQM method is used for identifying the most appropriate metrics for performing the comparison between software process and quality.The first step is to state the measurement goal of the study and the objectives. The second is deriving from the goal a set of questions whose answers help to determine whether the goal is being achieved. The third step is to design and develop a set of metrics to provide answers to these questions:

- **Goal**: Assessing the impact of software development process on quality
- **Question**: How a set of quality attributes (Performance, Security, Usability, Modifiability, Availability, Testability, Interoperability and Overall quality) are perceived as improved by software specialists in Traditional vs. Agile development.
- **Metric**: *t-test* analysis between the two groups of data for each of the quality aspects

On the basis of the GQM, we can frame our empirical investigation in terms of the following research hypothesis:

HP0 (Null Hypothesis): Agile Methods do not have any effect on the quality perceived by the developers of the produced systems

[1] http://www.questionpro.com/a/summaryReport.do?surveyID=4182537

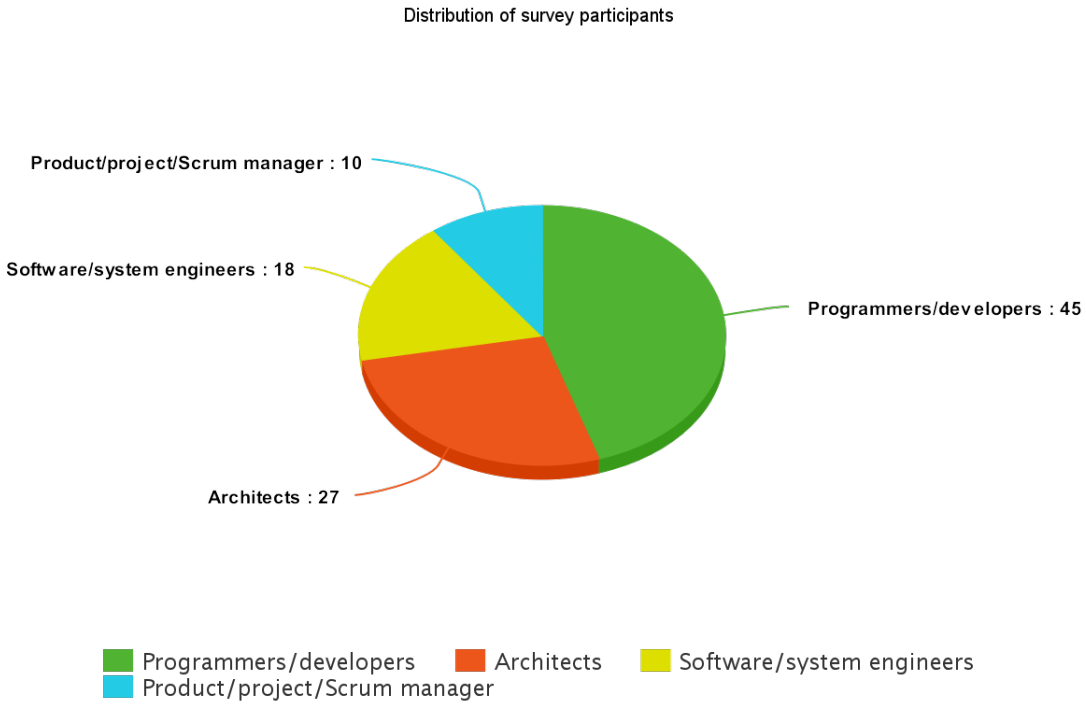

Fig. 1. The survey participants reflected a diverse range of positions for the surveyed projects: 45% of the participants describing themselves as programmersdevelopers, 27% as Architects, 18% as SoftwareSystem Engineers and 10% as product projectScrum managers

HP1 (Alternate Hypothesis): Agile Methods have an effect on the quality perceived by the developers of the produced systems

The agile methods that were reported in our study were SCRUM, Extreme Programming, Spiral, Prototyping, and Lean Software Development as shown in Figure III-B.

IV. Analysis of the Data

In this section, we analyze the responses of the survey results to determine if there exist significant differences between the groups of agile and non-agile methods. In total, 59 responses for the use of agile method and 44 answered to non-agile method. The two sample t-test analysis was used with the hypothesis defined in III-B.

We define as significant level α for our statistical tests 0.05, as usual in software engineering.

In Table I, we show statistical distributions for each feature. A positive value of t indicates that there is a higher quality in the non agile team, a negative value that there is a higher quality in the agile team. As mentioned, the null hypothesis is that there is no significant difference between the quality levels in agile and non agile environments.

As we can observe from the table, in all criteria the agile situation performs better than the non agile. Moreover, for functionality, security, usability, and overall quality, such difference is significant at the desired level, meaning that we can consider it not coming from change but truly generalizable.

It is remarkable to notice that the quality criteria related to the end user experience (security, usability, and overall quality) appear the one mostly affected by the adoption of an agile process and the ones statistically significant at the selected α level, indicating that such process is particularly suitable to optimize the satisfaction of the end customer.

	t	p
Performance	-1.80	0.08
Security	-2.29	0.02
Usability	-2.28	0.03
Modifiability	-1.19	0.24
Availability	-1.32	0.19
Testability	-1.69	0.10
Interoperability	-1.48	0.14
Overall Quality	-2.12	0.04

TABLE I
Results of the t-test.

It could be argued that we are in a situation of multiple tests, so that we would need to use the Šidák correction for multiple comparisons [1]. We could argue that such correction is never applied over software engineering data and does not apply here since we evaluate different hypotheses, we do not randomly try to see if there is any connection between agility and quality. Still using such very conservative correction over 8 tests out of which 3 are positive would lead to an α value of 0.02, which is also practically satisfied in the cases under consideration.

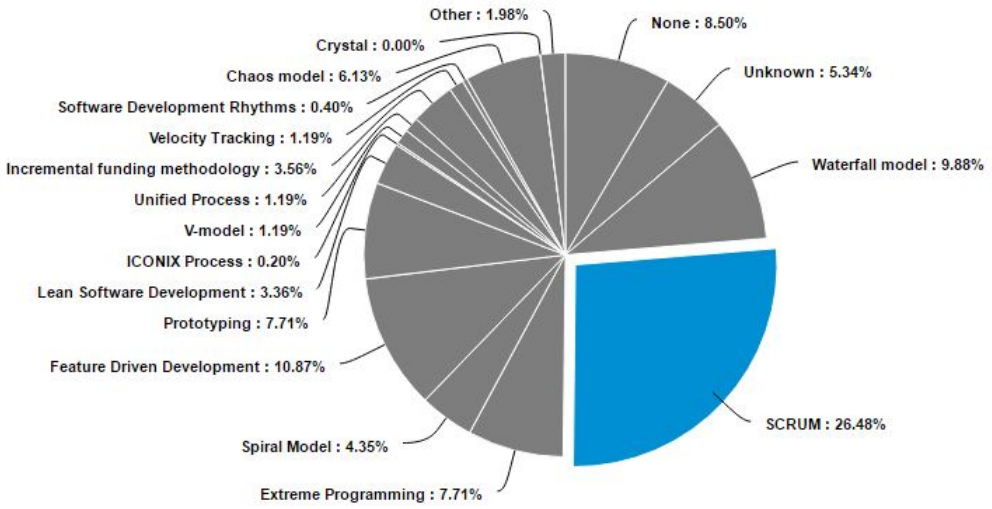

Fig. 2. Distribution of software development practices. SCRUM appears to be dominant.

Fig. 3. A graphical representation of Table I. The relationship of T and P values are shown.

V. Discussion

The results of the questionnaire evidences in a non-anecdotal and empirical manner that the adoption of agile development methods brings an improvement to all the aspects of software quality that have been studied: Performance, Security, Usability, Modifiability, Availability, Testability, Interoperability and Overall quality. The choice of the aspects of software quality on which to focus followed [4] and it is a further elaboration of the result of our previous paper [35]. The results are supported in a statistically significant way and with a strong effect size. Given the large and diverse sample of the respondents, these results have a high likelihood to be extensible to the overall population of software projects.

The results of this study have a strong relevance for software industry. In particular, the breadth of the respondents to the survey, give it a unique credibility and reliability. The study is a clear (re)affirmation that developers, architects and managers interested in increasing and/or maintaining software quality should consider carefully the adoption of agile methods. This is particularly true for companies with a strong focus on security, usability, and testability,

VI. Conclusions

Evidence that relates software process to quality is often anecdotal or, when data is presented, it is to some extent contradictory or clashes with opposite evidence coming from different samples. The matter is controversial and we have no intention to present conclusive data here or have the last say. However, the survey we present in this paper involved more than 100 software specialists around the world, and several questions have been actually answered by hundred of respondents.

The fact that a significant part of respondents dropped the questionnaire in the middle demands for better survey design, which is under development. In any case, the current survey has been reopened and, at the moment, we have a larger amount of data available that requires to be analyzed. A further confirmation of the trend seen here would make the conclusions even stronger, at a point that could not be ignored by scientists and practitioners.

In this paper we have established methods and approach to the research which will be applied and extend in future. Our future steps are:

1) Analyzing the extended data set
2) looking for confirmation (or confutation) of the results presented in this paper
3) Drawing the necessary conclusion
4) Consulting software specialists on the process to adopt on the basis of the results

ACKNOWLEDGEMENTS

We would like to thank Innopolis University for logistic and financial support. Our gratitude goes to colleagues at IU who participated in discussions and seminars, to PR department who advertised the questionnaire and to all the anonymous respondents who dedicated their precious time to shed light on the matter.

REFERENCES

[1] Hervé Abdi. Bonferroni and Šidák corrections for multiple comparisons. In *Encyclopedia of Measurement and Statistics*, pages 103–107. Sage, 2007.

[2] S Ambler. Agile requirements changement management, 2014.

[3] Victor R. Basili, Gianluigi Caldiera, and H. Dieter Rombach. The goal question metric approach. In *Encyclopedia of Software Engineering*. Wiley, 1994.

[4] Len Bass, Paul Clements, and Rick Kazman. *Software Architecture in Practice*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2 edition, 2003.

[5] Len Bass, Paul Clements, and Rick Kazman. *Software Architecture in Practice*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2 edition, 2003.

[6] Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, and Dave Thomas. Manifesto for agile software development, 2001.

[7] Sujoy Bose, Manish Kurhekar, and Joydip Ghoshal. agile methodology in requirements engineering. *SETLabs Briefings Online*, 2008.

[8] Ken W. Collier. *Agile Analytics: A Value-Driven Approach to Business Intelligence and Data Warehousing*. Addison-Wesley Professional, 2011.

[9] Enrico di Bella, Ilenia Fronza, Nattakarn Phaphoom, Alberto Sillitti, Giancarlo Succi, and Jelena Vlasenko. Pair programming and software defects–a large, industrial case study. *IEEE Transactions on Software Engineering*, 39(7):930–953, 2013.

[10] Snezana Djokic, Giancarlo Succi, Witold Pedrycz, and Martin Mintchev. *Meta Analysis — a Method of Combining Empirical Results and its Application in Object-Oriented Software Systems*, pages 103–112. Springer London, London, 2001.

[11] Papatheocharous E. and Andreou A. S. Empirical evidence and state of practice of software agile teams. *J. Softw. Evol. and Proc.*, page 855866, 2014.

[12] Martin Fowler. The new methodology, 2005.

[13] Robert M. Groves, Floyd J. Fowler, Jr., Mick P. Couper, James M. Lepkowski, Eleanor Singer, and Roger Tourangeau. *Survey Methodology*. Wiley, Hoboken, N.J., second edition, 2009.

[14] G Hoinville and R Jowell. *Survey Research Practice*. SCPR, 1 edition, 1982.

[15] Mohamad Kassab. An empirical study on the requirements engineering practices for agile software development. In *2014 40th EUROMICRO Conference on Software Engineering and Advanced Applications*, pages 254–261. IEEE, 2014.

[16] Mohamad Kassab, Colin Neill, and Phillip Laplante. State of practice in requirements engineering: contemporary data. *Innovations in Systems and Software Engineering*, 10(4):235–241, 2014.

[17] Mahvish Khurum, Niroopa Uppalapati, and Ramya Chowdary Veeramachaneni. Software requirements triage and selection: state-of-the-art and state-of-practice. In *2012 19th Asia-Pacific Software Engineering Conference*, volume 1, pages 416–421. IEEE, 2012.

[18] Vincent Marinelli. *An analysis of current trends in requirements engineering practice*. 2008.

[19] C. Marshall and G.B. Rossman. *Designing Qualitative Research*. Sage Publications, 2006.

[20] Gurpreet Singh Matharu, Anju Mishra, Harmeet Singh, and Priyanka Upadhyay. Empirical study of agile software development methodologies: A comparative analysis. *SIGSOFT Softw. Eng. Notes*, 40(1):1–6, February 2015.

[21] Colin J Neill and Phillip A Laplante. Requirements engineering: the state of the practice. *IEEE software*, 20(6):40, 2003.

[22] Uolevi Nikula, Jorma Sajaniemi, and Heikki Kälviäinen. *A State-of-the-practice Survey on Requirements Engineering in Small-and Medium-sized Enterprises*. Lappeenranta University of Technology Lappeenranta, Finland, 2000.

[23] Frauke Paetsch, Armin Eberlein, and Frank Maurer. Requirements engineering and agile software development. In *WETICE*, volume 3, page 308, 2003.

[24] Alcides Quispe, Maira Marques, Luis Silvestre, Sergio F Ochoa, and Romain Robbes. Requirements engineering practices in very small software enterprises: A diagnostic study. In *Chilean Computer Science Society (SCCC), 2010 XXIX International Conference of the*, pages 81–87. IEEE, 2010.

[25] Preeti Rai and Saru Dhir. Impact of different methodologies in software development process. *International Journal of Computer Science and Information Technologies*, 5:1112–1116, 2014.

[26] Rudd McGary Robert K. Wysocki. *Effective Project Management: Traditional, Adaptive, Extreme*. John Wiley and Sons, 3 edition, 2003.

[27] W. W. Royce. Managing the development of large software systems: Concepts and techniques. In *Proceedings of the 9th International Conference on Software Engineering*, ICSE '87, pages 328–338, Los Alamitos, CA, USA, 1987. IEEE Computer Society Press.

[28] J.J. Shaughnessy, E.B. Zechmeister, and J.S. Zechmeister. *Research Methods in Psychology*. McGraw-Hill Higher Education. McGraw-Hill, 2003.

[29] D. Silverman. *Doing Qualitative Research: A Practical Handbook*. SAGE Publications, 2000.

[30] Badariah Solemon, Shamsul Sahibuddin, and Abdul Azim Abd Ghani. Requirements engineering problems and practices in software companies: An industrial survey. In *International Conference on Advanced Software Engineering and Its Applications*, pages 70–77. Springer, 2009.

[31] Marian Stoica, Marinela Mircea, and Bogdan GHILIC-MICU. Software development: Agile vs. traditional. *Informatica Economic*, 17(4), 2013.

[32] Stoimen. An introduction to the agile, 2011.

[33] Stoimen. Main features of the traditional software development methodologies, 2011.

[34] G. Succi, W. Pedrycz, M. Marchesi, and L. Williams. Managing the development of large software systems: Concepts and techniques. In *Proceedings of the 3rd International Conference on Extreme Programming (XP)*, XP 2002, pages 212–215, 2002.

[35] Rasul Tumyrkin, Manuel Mazzara, Mohammad Kassab, Giancarlo Succi, and JooYoung Lee. Quality attributes in practice: Contemporary data. In *10th KES International Conference, KES-AMSTA 2016 Puerto de la Cruz, Tenerife, Spain, June 2016 Proceedings*, pages pp 281–290. Springer International Publishing, 2016.

[36] June Verner, Karl Cox, Steven Bleistein, and Narciso Cerpa. Requirements engineering and software project success: an industrial survey in australia and the us. *Australasian Journal of information systems*, 13(1), 2005.