Çevikliği Arttırmak İçin Çevik ve Geleneksel Süreç Yaklaşımlarını Dengelemeye Yönelik Bir İnceleme

Necmettin Ozkan[1], Ayça Tarhan[2]

[1]Kuveyt Türk Katılım Bankası, Kocaeli, Türkiye
[2]Bilgisayar Mühendisliği Bölümü, Hacettepe Üniversitesi, Ankara, Türkiye
[1]necmettin.ozkan@kuveytturk.com.tr, [2] atarhan@hacettepe.edu.tr

Özet. Çevik yazılım geliştirme, kendi faydasına makul bir zemin oluşturmak amacıyla geleneksel yöntemi göz ardı ederek ona karşı ön yargılı bir yaklaşımla ortaya çıkmıştır. Mutlak çeviklik belirli bir bölgenin ötesinde de bir hak olmasına karşın, bu yaklaşım, çevikliğin kendi uç noktasında bir konfor alanı ve beraberinde zayıflık doğurmuştur. Bu nedenle, bu çalışma çevik yazılım geliştirmenin daha geniş, sürdürülebilir ve daha az bağlam bazlı olması amacıyla çevikliğin geleneksel yaklaşımlarla dengelenmesini ve normalleştirilmesini önermektedir. Çalışma geleneksel yaklaşımların katkısıyla çevik yazılım geliştirmenin iyileşmesi gereken konularını belirlemiş ve belirlenen her bir konu daha sonra, her iki yaklaşımın avantajlarını ortak bir zeminde birleştirmek ve çevik yazılım geliştirmenin eksikliklerini gidermek için geleneksel yaklaşımların güçlü yönlerine dayanan öneriler ile zenginleştirilmiştir.

Anahtar Kelimeler: Çevik, Geleneksel, Plan Güdümlü, Yazılım, Scrum, Şelale, Denge

An Investigation into Increased Agility by Balancing Agile and Traditional Process Approaches

Necmettin Ozkan[1], Ayça Tarhan[2]

[1]Kuveyt Türk Participation Bank, Kocaeli, Turkey
[2]Computer Engineering Department, Hacettepe University, Ankara, Turkey
[1]necmettin.ozkan@kuveytturk.com.tr, [2] atarhan@hacettepe.edu.tr

Abstract. The agile software development (ASD) has come with a biased approach against to its traditional counterparts because of expunging it by overriding to create a reasonable ground for its own benefit. This approach has created a comfort zone at its extreme edge and associated weakness for the agility in software development even though the absolute agility is a right for beyond such a zone. Thus, this study suggests balancing and normalizing the ASD with traditional approaches for a broader, sustainable and less context-based agility. We begin by identifying the topics the ASD should improve with the contribution of traditional approaches. Each corresponding topic is then elaborated with

CEUR-WS.org/Vol-2201/UYMS_2018_paper_37.pdf

suggestions based on the strengths of the traditional approaches to reach a middle ground to combine the advantages of the both and fix the shortfalls of the agile development.

Keywords: Agile, Traditional, Plan-Driven, Software, Scrum, Waterfall, Balance

1 Introduction

Traditional software development, with its Waterfall Methodology instance, has been dominating the sector for several decades [23]. The Waterfall approach assumes that systems are fully specifiable, predictable and developed through extended and detailed planning [9, 22]. Thus, it follows a systematic and linear approach [7] to software development life cycle. The more systematic (self-confident) approach makes it more linear (claiming being perfect). On the other side, the inward belief that the plan will not work as expected – especially in today's turbulent business environment – creates more anxiety and more control to eliminate it with high assurance. Especially in the latest decades, the traditional approach is criticized due to some drawbacks and shortcomings in its nature [9]. In response to the problems of this methodology, agile approach has appeared [14] to meet the need for faster time to market, shorter development cycles, lower development cost, and the ability to move and change quickly [30]. However, the current agile approaches do not always provide the optimum solutions to the problems that are faced by the traditional approaches [14].

Agile approaches, defined with the ability to respond to change, have generally been seen as the contrary to traditional (heavyweight, disciplined, predictive, plan-driven) approaches due to opposing viewpoints of [9, 24]. A common example is the paradigmatic assumption in the agile world that the Agile Software Development (ASD) Manifesto is more about a replacement of traditional methods especially with its four underpinning values: Individuals and interactions over processes and tools, Working software over comprehensive documentation, Customer collaboration over contract negotiation, and Responding to change over following a plan [53]. While it is stated there is value in the items on the right, there is no new or current attempt to interpret these values or their new position in the Agile world. Instead, the left side expunges the right side by overriding it [12]. Today, in practice, the left side has clearly separated itself from the right by emphasizing on the disadvantages of traditional methods to create a reasonable ground for its own benefit. This emerged “black and white” segregation has created a dual polarization between these two edges. As a result, Agile methods work well for projects within particular contexts: small; co-located teams; high customer involvement; requirements that change over weeks or months; variable scope/ price contracts; and few legal or regulatory constraints on development processes [35].

The context-dependency mentioned above also generates greatest risks of derailing agile methods for: large size, large systems with a lack of architectural focus; software development not driven by customer demand; traditional governance; novice team; and very high constraints on some quality attributes (e.g. safety-critical system and real-time constraints) [34]. Staying at their extreme edges, both the agile and traditional approaches thus have situation-dependent shortcomings that, if left unaddressed, can lead to failure [24]. For many projects, the pure traditional project management is not

effective or the pure agile method is not good enough [2, 14]. This brings us to the idea that the more a particular project's conditions differ from the home-ground conditions, the more risk in using one approach in its pure form, and the more valuable blending in some of the opposite methodology's complementary practices becomes [33].

Supporting that absolute agility should not be monopolized by a specific context, there is a need for a balance that enables to harness the bests of the agile and traditional process approaches [1, 5, 10, 24, 27] to blend the capabilities of the both edges and push the ASD outside of its comfort-zone. We believe that such an intermediate solution should first evolve on the design axis, like what 'gray' really means. Therefore, this study approaches the subject from the theoretical aspect to bring the two approaches closer on this design axis, and leaves the following issues out of the scope:

- Applying agile methods in an ecosystem where classical methods already exist or vice versa,
- How these two approaches can be used together maintaining their own essence (Hybrid Agile Methods).

Accordingly, the research questions (RQs) were identified as below. In trying to answer the RQs, while considering ASD in general terms through values and principles, Scrum and Waterfall have been preferred to study in particular. This selection contributes to be more concrete and to cover mostly used practices in the field [36].

- RQ1: What are the points that need to be strengthened to enable the agile methods to reach more potential?
- RQ2: How can these issues be resolved, what are the suggestions, and what contributions can traditional methods make for the current state of agility?

The remaining of this work is organized as follows. We provide an overview of traditional and agile methods and the previous related work in section 2. We then show in section 3 the topics the ASD should improve as relevant to RQ1. Each corresponding topic includes suggestions based on the strengths of the Waterfall model to reach a middle ground to combine the advantages and fix the shortfalls of the agile development as relevant to RQ2. In section 4, the subject is evaluated with discussions, and finally in section 5, conclusions and future work are mentioned.

2 Related Works

Table 1 provides a summary of the characteristics of the Waterfall and Scrum as the foremost models of the two edges, which is taken from an extensive overview of the traditional and agile approaches provided in [55].

Table 1. Basic characteristics of Waterfall and Scrum models [55]

Waterfall	simple, linear, ordered activities; regular and heavy documentation; activities include feasibility study, requirements analysis, design, implementation and unit test, integration and system test, and operation and maintenance; once an activity is completed, it is difficult to return back to it; beneficial when requirements can be clearly and completely identified at the start of the project and the changes to these requirements can be restricted thereafter
Scrum	focuses on practices about 'project' management and does not include engineering details; postulates that the project scope is to change during the

	project; has practices which can evade instability and complications instead of having specific practices of software development (e.g. every executable increment of the software is completed within maximum 30 days called "sprint" and delivered to the customer)

Many of today's projects will not fit to the extremes of pure agile or traditional approaches. There exist many shades of gray in the spectrum between agile and traditional methods [10] and most projects will be in the gray area in between [14, 31]. While these two approaches seem conflicted, a reasonable balance between agile and traditional development seems to be a clearer approach to deal with the ASD limitations [14], even for achieving the flexibility and benefits promised in the agile philosophy.

"Balancing" the agile approaches with the traditional ones does not mean any mix or combination of them in a hybrid way. Combining the agility and the discipline in the same project may face several challenges [14, 32]. Similarly, combining them in a same organization [14] or in a same team [31] is out of scope in this study, as hereditary issues will continue to exist if they maintain their own essence. Developing a balance should take advantage of the strengths and mitigate the weaknesses of the agile approaches without either losing the benefits of agility, instead of coexistence of these approaches preserving their 'contradictory' natures.

Yet, there are previous works studying on hybrid models, mixing compatibility of agile and traditional methods in different setups such as [2, 10, 14, 24, 31, 33, 42], which are out of scope in this study as mentioned above. Disciplined Agile Delivery [11] that aims at specifically scalable agile solutions tries to address an entire solution development life cycle, from concept to product, presenting disciplinary methods at some degree. It continues to use the phenomenon of the project and offers detail in areas such as architecture and design with a centralized approach. However, a project manager role is not one of the roles in it and it is not clear how the iron triangle of project is defined and managed. There is also no objection in it to the solid and static events approach and physical dependency issues coming from the core.

To identify other relevant studies, "balance traditional agile" keywords were used in a systematic review of the literature made between 04/05/2018 and 06/05/2018. Within the studies retrieved from ACM Digital Library, IEEE Xplore, DOAJ, Web of Science, (the first 200 records sorted by relevance in) Google Scholar, Science Direct and Scopus; 309 results were investigated. Any examples of studies addressing the subject from the design aspect to strike a balance between the two approaches as in this study have not been encountered. This picture reinforces the assertion of that agile approach's compatibility with traditional methodologies is largely unexplored [24] especially from the design aspects and this study tries to fill the gap.

3 Evaluation of Traditional versus Agile Practices

To address the root causes of the issues and to uncover the fundamental points in the ASD within the related context, we go back to the origin of it to re-visit the values and principles of the Agile Manifesto and the Scrum Guide [46]. Finally, the following subjects are identified, defined and discussed to answer the research questions of this study.

Up-Front vs. Emergent: Due to the evolutionary nature of software projects, changing markets and evolving technology [5], change usually becomes inevitable in many aspects of projects including requirements, circumstances, and stakeholders [4]. As a result, requirements present some degree of variability on the course of a project. The reasons behind this "requirements drift" [4] may be listed as the following:

- The nature of software development is so complex that allowing upfront requirement gathering is not feasible [9].
- Things change on the way because of highly volatile business environments.
- Humans use distortion, deletion and generalization to express their thoughts and such cognitive issues may cripple requirement elicitation process [44].
- Incapacity of people involved in the corresponding processes may cause things change when things get closer.

Agile approaches bring proposals to the first and second items. The last two still stay there as bigger issues to deal with in the ASD. With formidable responsibility on the customer's part, empowering incapable customers with more initiatives on software development, and regarding customers' change requests spring from distortions, deletion and generalization as absolute, innocent and harmless demands may danger the quality, cost and time scale of projects. Therefore, the success of agile development relies on finding customers who are expected to be collaborative, representative, authorized, committed, and knowledgeable, that is not an easy task especially for complex systems [43]. Moreover, without clear requirements and feedback, the teams are forced to "make more business decisions than the team would like" [35], because in some way sprint has to progress. This highly dependence on customers may also cause failures if the customers are misaligned with the stakeholders' goals [42].

In agile approaches, there is an assumption that requirements can only be finalized quite late in the development cycle [49], keeping it away from upfront designs. An incremental way in requirement gathering may also lead to dependency problems in design [22]. Moreover, agile methods do not provide adequate design documentation necessary for future development [6].

On the other side, for the similar concerns, COBIT (Control OBjectives for Information and related Technologies) [54] explicitly call for business sponsor's approval for the proposed solution approach and high-level design specification and involvement of business sponsors and other stakeholders in quality reviews of project increments. Levels of uncertainty change over the life of the project, usually at the highest level at the beginning and gradually diminish as more aspects of the project are clarified [5]. Thus, the project can start at high levels with initially vaguely defined requirements and as much as it is clarified over a time of period, it evolves towards versions that are in more detailed. Based on this, there should also be an initial high-level plan, followed by detailed planning at each iteration that leads to the implementation [27]. A Sprint Zero can be placed for overall requirements gathering [20] and design even though requirements can be vaguely defined. Thus, the risk of confusion in terms of project objectives and deliverables will reduce. The important thing to note is that, while more general design is done at the first stage, it can be gradually introduced in detail in an adaptive manner later on when things get closer.

Enough Documentation vs. Heavy Documentation: Principles of the ASD prefer working software to comprehensive documentation [15]. However, this statement opens gates to misimpressions and bases for a source of misleading in practice. Documentation is not equivalent to software. This misleading comparison leads to the misconception that the documentation matters can be solved by early software delivery. In addition, in agile methods, it is not possible to do heavy documentation in short iterations that do not allow it or do not need it. By using this newly introduced gap at the design, agile software developers may omit documents due to sprint pressure [17], and they may consider documentation as a secondary and non-creative activity [18]. Many agile software developers are prone to use to follow the agile values as an excuse to hack undocumented, poorly designed code [10]. Finally, significant reduction of documentation and the claim that the source code itself should be the documentation [21] reduce the formality and the quantity of documentation [16]. This lack of documentation causes defects in software evolution and maintenance, lack of visibility for project monitoring and technical solutions, and poorly understood requirements [19].

However, documentation needs continue to live for developers [28, 37] and for the software to develop. If software development requires heavy design documentation, this need of the development should be met. Thus, the Agile Methods must decide where to place the balance point in documentation [13]. In searching this place of balance, as members of the team prefer simple and practical documentation techniques [38], lean (not necessarily agile) approaches aiming at avoiding unnecessary documentation should be preferred to reach 'just enough' documentation, that can be ‘heavy’ if required.

Dynamic vs. Static Iteration/Event Durations: While the manufacturing sector has shaped the classical methods, the agile methods have been influenced by this approach and have continued to experience the planning phenomenon, albeit not severely, but with different dimensions. Every agile method has a plan, although not the same way as the traditional methods do, but the 'plan' part in the agile methods cannot be ignored [14]. The framework designers, rather than the people applying Scrum, have fixed the maximum duration and the frequency of the iterations/meetings. In such case, some assumptions arise: 1) It is possible to break the business needs down to the small pieces such that it would be possible and meaningful to manage them on a regular monthly and daily basis; 2) It is possible to create a potentially shippable product in each solid and static iteration; 3) Based on this, with no exception, it is expected to start counting from (sprint) 1 to come with a potentially shippable product and so forth; and 4) The time planning and designs of sub-parts of sprints especially in large-scale are expected to be perfectly mechanized with a perfect flow of events.

However, each customer need is a whole and it is, sometimes, not dividable into such solid, static and short sprint lengths literally. Same difficulty prevails for time boxes of events at the developer side, especially for large-size projects when it calls for longer time then what the framework forces. Large-scale developments that require harmonies of multiple sets increase the difficulties that are remarkable even for a single team.

The time frame for each iteration is so short that developers find the schedule too tight [22]. This leads to delays in each iteration and hardships in establishing an efficient communication between team members and clients [22]. Time pressure may also lead to reduced quality assurance instead of de-scoping [28]. Moreover, developers may find the periodic meetings and 'an indefinitely constant pace' of iterations boring and tiring [22]. Continuous, short and static length in sprints can produce press and consequently stress for developers [39] and can become a routine after a while [40].

The number of phases and the duration of the iterations should be dependent on the levels of ambiguity, uncertainty, innovation [27] and value. Relevant stakeholders should decide the finish time of iterations at run time, rather than at very beginning of the iteration starting with a strict deadline. The same criteria can apply to events, instead of making them static with time-box constrains. This approach is also parallel to the view of Conboy and Fitzgerald whom study of experts' opinion on Agile methods notes that "the very name agile suggests that the method should be easily adjusted to suit its environment" [41].

Project vs. Product-Based Development: By definition, the Scrum Guide regards a sprint as a project and a project as a sprint [46]. In the Scrum Guide (project) planning cannot find a proper place and the horizon in this manner does not go beyond sprint planning borders. From this point of view, it neglects the (scalable) project space and this differs from project definition of the customer who describes it from a holistic and a wider (and scalable) perspective. Similarly, project manager role is debatable and by prominent Scrum authorities it has been underestimated and engaged to Scrum master role [45], even though Scrum master role is for the pure Scrum processes, and not for project. Additionally, design of program and portfolio management stays at the level of management of epics, features and theme, free from their project relevance. Rather than being project oriented, Scrum, not explicitly the Agile Manifesto, focuses on product and its continuous delivery [45]. The role of the product owner and the constructions around the product are aimed at managing the product at the center.

However, it is difficult to find a proper place for a product-oriented approach in the field of Information Technology (IT), which is dominated by a process-oriented approach like in COBIT, service-oriented approach like in ITIL (Information Technologies Infrastructure Library), and a project-oriented approach like in PMBOK [48]. Moreover, the thinking has shifted from pure product focus to a combination of service and product [47], and the pure product concept is prone to disappear inside service. On top of all that, while a focus on product of customers may be useful in the context of the industry, there is a difference in the software field: the development process is complex and dynamic and it deserves an interest at least as much as the result itself.

People also want to manage changes, especially large and complex ones, via a project. Regardless of frameworks, the concept of project inevitably is a living phenomenon in the real life of IT, same for project manager [8]. The reaction of the agile world to this situation emphasizes this fact: ten thousand results returning from Google Scholar with "agile project management" keyword, for just the time being.

This study comes with the idea that possible definition of project in Scrum can be parallel to the definition of project provided by PMBOK [48]. The project definition provided by PMBOK is more suitable for the general definition from the customer's

perspective that is free from any applied methodology. According to PMBOK, "project is a temporary effort to fulfil a unique product, service or result". The temporary nature of the projects expresses a definite start and an end. This end is met by finalizing the project when it is understood that the purpose of the project is achieved or that it is clear that the purpose cannot be achieved or when the need for the project disappears. Similarly, in Scrum, a project in this sense is finalized when all the work items related to the project is completed or the cost of the next iteration (sprint) is more than the value of the iteration. Thus, this study recommends managing projects dynamically based on (dynamic) value at the run time. This study also highlights the contributions from the abstraction and ability of connecting of a project. Project may also work for encapsulating an end-to-end solution development, covering pre- and post-development stages including project transition, trainings and creating user instructions, documentation materials, that is ignored in Scrum. In addition, this work highlights the necessity of well-established project management for enabling scalable program/portfolio management.

Digitalization vs. Physical Dependencies: Scrum's core elements are designed to leverage physical co-location and extensive face-to-face interactions [28], being typically exemplified in ceremonies, physical boards etc. Especially, the face-to-face communication mandates, naturally, two limitations: synchronizing people in terms of same time and place. From time perspective, one of the most prominent features of such a communication method is that it limits capabilities to the present time only. Meetings that are a means of communication are a derivative of the present time. Thus, abilities relevant to the past and future time decline. Addiction to the same place requires the teams to be close together. This has led to the conclusion that agile methods are appropriate for co-located teams [29]. Aside from this, the manner of face-to-face talks, which are advised as a communication tool [15], is a kind of challenge even for co-located teams, as it requires the same place at the same time for all relevant actors of multiple simultaneous teams and parties. Thus, the Scrum's nature depending on meetings and face-to-face communication on physical platforms manifests itself as a factor inhibiting flexibility and accompanying agility.

The lack of documentation, process and tools reinforces physical limitation as well. Documentation and tools especially together play roles as a means of digitalization. Declining such capabilities by the lack of power of digitalization may decrease meeting/discussion efficiency [28]. Although documentation is in agile methods used in the context of design and communication manner, in general it serves as the storage of information (belonging to the employee), the replication (of employee information), the transmission (of information in a not effective but efficient way), spreading (information easily to multiple locations), creating a history with traceability features. Similarly, for tools and processes as well, these aspects of information should be re-considered for a right balance of digitization to create capabilities for all variants of time and location axis. The Agile Methods must also keep pace with the requirements of the digital age and benefit from the advanced digitization capabilities of this era (such as e-collaboration, electronic boards [35]), by utilizing accumulated documentation, process and tool capabilities.

Centralization vs. Decentralization: "Divide, operate and integrate" is the ASD's overall approach to large pieces. During the all stages, inter-team coordination and communication are major problems of large organization adopting agile methods needs to solve [50]. The teams may live in their (not necessarily yet potentially possible) individual, isolated and feudalized environments because of being in self-management and self-organized nature [51]. They may at least potentially run at discrete direction different from the common goals of projects, programs or designs. Thus, in classical structures, while separation of developer functions creates horizontal silos, in Scrum's modular and granular structure vertical corridors that descent from customer to the team will cause formation of vertical silos and one way or other silos at the end. One key issue in this regard is to investigate how to balance inter-team coordination and self-management in multi-team development [3].

Even though central roles and structures out of 'self-organized' teams are regarded as practices against to the core [38], yet they are needed for large organizations. As mentioned by Jurgen Appelo, "Agility means self-balanced, not self-organized" [52], including a balanced point between centralization and decentralization.

4 Discussion

It is quite possible to adopt a labeled set of agile practices or a set of practices that perfectly conform to the Agile Manifesto and not become agile [34]. We take this viewpoint a step further; it is quite possible to adopt the Agile Manifesto and not reach absolute agility, as there is no guarantee or a proof of assurance stating that the Agile Manifesto fully ensures the absolute agility for software development. Thus, there is a potential gap between the capabilities of the Agile Manifesto and the absolute agility.

Kruchten [34] defines agility as "the ability of an organization to react to changes in its environment faster than the rate of these changes". This definition leads us to the point; agility is not only the right of a certain zone or organizations of a certain size and kind. On the other hand, the ASD fits and is most likely to succeed only within its own "home ground" [24], called as the "agile sweet spot" [26] or the "comfort zone" for Scrum in particular [25]. This is a dilemma that must be overcome.

There rationally may exist places where an absolute agility may be naturally needed, beyond such a "comfort zone" or "agile sweet spot". This is why, although agile methods enjoy their comfort zone, some organizations have already started to push agility in software development outside of this out-of-box comfort area [29, 35].

In pushing the ASD outside of its comfort zone, this study proposes to normalize its biased position, spring from its radical approach to the traditional approaches, with the contribution of its apparently 'contradicting' counterpart. We believe that they can work together for better improvement, even starting with the design stage by balancing to take advantage of their strengths and compensate for their weaknesses (of the agile side in this case).

If "…there is value in the items on the right" as stated in the Agile Manifesto, then this study aims to bring the values on the right to the surface, to consider them for more sustainable and broader agility. In doing so, this study argues that being abstract is more agile than being concrete (same as project is more than product and being digitalized is more than being dependent on physical entities), being dynamic is more agile than

being static, managing over balanced centralized structures is more agile than managing district structures, and being enough is more agile than being less or more.

5 Conclusion

The illusion of staying at the comfort zone may have led to thinking that the agile method has universal value, that its represents some ultimate recipe, the holy grail of software engineering [34]. Sticking to this believe, while agile evangelists exhort teams to adopt their methods whole, in such every project following a particular method must adopt every practice, as described in the manuals, books and courses [35], we challenge this assertion by this study.

Agile is one of the adjectives in the universe and disciplined, sustainable, mature, stable, strong are some others. While 'agility' in the current state of software development enjoy its comfort zone alone, organizations today still need to have some other abilities that the Agile world intentionally or unintentionally ignores. We suggest searching for a proper integration and harmony of agile mindset mentally, theoretically and practically with other realities and needs of organizations. To help us define such adequate process or set of practices outside of the agile sweet spot, cold-headed and impartial investigation is required even though such research is generally not very easy to conduct [34]. It is hoped that this work will contribute in this manner.

While both the ASD and Scrum-specific issues have been addressed in the study, these two relatives have, whenever needed, been tried to be distinguished as far as possible throughout the study. Nevertheless, the reader may be required to pay extra attention to separate the issues solely specific to Scrum from those of the ASD introduced in general.

The assertions (especially for pain points and solution proposals) provided in this study should be justified within practice. It is important that selected sample set be experienced in both traditional and agile methods, preferably in a same organization. After the case study, the deductions obtained in this study should be revised, if necessary. This remains as a future work.

References

1. Bjarnason, E, Krzysztof W., Björn R: A case study on benefits and side-effects of agile practices in large-scale requirements engineering. In: 1st Workshop on Agile Requirements Engineering, ACM, (2011).
2. Mojdeh, R.: A comparative study on hybrid IT project management. International Journal of Computer and Information Technology. 3(5), 1096-1099 (2014).
3. Moe, N. B., Dingsøyr, T.: Emerging research themes and updated research agenda for large-scale agile development: a summary of the 5th international workshop at XP2017. In: XP2017 Scientific Workshops. ACM, (2017).
4. Henderson, P.: "Why Large IT Projects Fail". [Online]. Available: http://de.scientificcommons.org/43269531. [Accessed 04 05 2018].
5. Noureddine, A. A., Meledath D., Samira Y.: A Framework for Harnessing the Best of Both Worlds in Software Project Management: Agile and Traditional. In: Information Systems Education Conference, (2009).
6. Pressman, R.: Software Engineering, A Practitioner's Approach. McGraw Hill, (2005).

7. Royce, W. W.: Managing the Development of Large Software Systems: Concepts and Techniques. In: 9th international conference on Software Engineering, pp. 328-338. (1987)
8. Shastri, Y., Hoda, R., Amor, R.: Does the project manager still exist in agile software development projects? In: 2016 23rd Asia-Pacific Software Engineering Conference (APSEC), pp. 57–64. (2016).
9. Vaishnavi, K., Jhajharia, S., Verma, S.: Agile vs. waterfall: A comparative analysis. International Journal of Science, Engineering and Technology Research. 3(10), 2680-2686 (2014).
10. Parsons, D. J: Army Simulation Program Balances Agile and Traditional Methods With Success. The Journal of Defense Software Engineering. (2006).
11. Ambler, S.W., Lines, M.: Disciplined Agile Delivery: A Practitioner's Guide to Agile Software Delivery in the Enterprise. IBM Press, New York (2012)
12. Janes, A. A., Succi, G.: The dark side of agile software development In: ACM international symposium on new ideas, new paradigms, and reflections on programming and software, pp. 19-26. (2012).
13. Paulk, M. C: Agile Methodologies and Process Discipline, Institute for Software Research, Paper 3 (2002).
14. Galal-Edeen, G. H., Riad, A. M., Seyam, M. S.: Agility versus discipline: Is reconciliation possible? In: International Conference on Computer Engineering & Systems, IEEE, (2007).
15. Agile Manifesto (2001), http://www.agilemanifesto.org (Accessed on April 2018)
16. Turk, D., France, R., Rumpe, B.: Limitations of agile software processes. In: Third International Conference on Extreme Programming and Flexible Processes in Software Engineering, pp. 43–46. (2002).
17. Sneed, H. M.: Dealing with Technical Debt in agile development projects. Lect. Notes Bus. Inf. Process. LNBIP, vol. 166, pp. 48–62. Springer, (2014).
18. Clear, T.: Documentation and Agile Methods: Striking a Balance. SIGCSE Bull. 35(2), 12–13 (2003).
19. Uikey, N., Suman, U., Ramani, A.: A Documented Approach in Agile Software Development. Int. J. Softw. 2(2), 13–22 (2011).
20. Tadeo, P. L., Borrego, G.: Scrumconix: Agile and documented method to AGSD. In: 11th International Conference on Global Software Engineering (ICGSE), IEEE, (2016).
21. Vijayasarathy, L.R.: Agile Software Development: A survey of early adopters. Journal of Information Technology Management. 19(2), (2008).
22. Marian, S., Mircea, M., Ghilic-Micu, B: Software development: Agile vs. traditional. Informatica Economica. 17(4), (2013).
23. Laplante, P. A., Colin J. N.: The demise of the waterfall model is imminent. Queue. 1(10), (2004).
24. Barry, B., Turner, R.: Balancing agility and discipline: Evaluating and integrating agile and plan-driven methods. In: 26th International Conference on Software Engineering, IEEE, (2004).
25. Lyon, R., Evans, M.: Scaling Up – pushing Scrum out of its Comfort Zone. In: Agile Conference, pp. 395-400. IEEE, (2008).
26. Reifer, D. et al. Scaling Agile Methods IEEE Software. July/August, (2003)
27. Binder, J., Aillaud, L., Schilli, L.: The project management cocktail model: An approach for balancing agile and ISO 21500. Procedia-Social and Behavioral Sciences. 119, 182-191 (2014).
28. Pernille, L., Kuhrmann, M., Tell, P.: Is Scrum fit for global software engineering?. 12th International Conference on Global Software Engineering (ICGSE), IEEE, (2017).
29. Dingsøyr, T., Moe, N. B.: Research Challenges in Large-Scale Agile Software Development, ACM Software Engineering Notes, vol. 38, pp. 38-39 (2013).
30. Conn, S.: A New Teaching Paradigm in Information Systems Education: An Investigation and Report on the Origins, Significance, and Efficacy of the Agile Development Movement. Information Systems Education Journal, 2(15), 3 – 18 (2004)
31. Boehm, B., Turner, R.: Balancing Agility And Discipline: A Guide For The Perplexed. Addison-Wesley, Boston (2004).

32. Boehm, B. Turner, R.: Management Challenges to Implementing Agile Processes in Traditional Development Organizations. IEEE Software 22(5), 30-39 (2005).
33. Boehm, B. Turner, R.: Using Risk to Balance Agile and Plan-Driven Methods. IEEE Computer 36(6), 57-66 (2003).
34. Kruchten, P.: Contextualizing agile software development. Journal of Software: Evolution and Process 25(4), 351-361 (2013).
35. Hoda, R., Kruchten, P., Noble, J., Marshall, S. Agility in context. ACM Sigplan Notices 45(10), 74-88 (2010).
36. Versionone, State of agile survey, https://explore.versionone.com/state-of-agile/versionone-11th-annual-state-of-agile-report-2, last accessed 2018/04/06
37. Sekitoleko, N., et al.: Technical dependency challenges in large-scale agile software development. In: International Conference on Agile Software Development, Springer, Cham (2014).
38. Melo, C.O., Santana, C., Kon, F.: Developers motivation in agile teams. In: 38th Euromicro Conference on Software Engineering and Advanced Applications, (2012)
39. Whitworth, E., and Biddle, R.: The social nature of agile teams. Agile conference (AGILE), (2007).
40. Šteinberga, L., and Šmite, D.: Towards a contemporary understanding of motivation in distributed software projects: solution proposal. Scientific Papers, vol. 15, University of Latvia (2011).
41. Conboy, K., Fitzgerald, B.: The Views of Experts on the Current State of Agile Method Tailoring, IFIP, vol. 235, pp. 217–234 (2007).
42. Vinekar, V., Slinkman, C., Nerur, S.: Can Agile and Traditional Systems Development Approaches Coexist? An Ambidextrous View. Information Systems Management 23(3), 31-42 (2006).
43. Nerur, S., Mahapatra, R., Mangalaraj, G.: Challenges of migrating to agile methodologies. Commun. ACM 48(5), 72-78 (2005).
44. Bandler, R.: Get the life you want. Health Communications, Deerfield Beach, Fla. (2008).
45. Ozkan, N., Kucuk, C.: A Systematic Approach to Project Related Concepts of Scrum. Revista de Management Comparat International 17 (4), 320-334 (2016).
46. Sutherland, J., Schwaber, K.: The Scrum Guide. (2017).
47. Parry, G., Newnes, L. Huang, X. Goods, Products and Services. In: Angelis, J., Parry, G. and Macintyre, M. (eds.) Service Design and Delivery, Service Science: Research and Innovations in the Service Economy (2011).
48. Project Management Institute: PMBOK Guide. Pennsylvania (2008).
49. Elshamy, A., Elssamadisy, A.: Divide after You Conquer: An Agile Software Development Practice for Large Projects. In: XP 2006, LNCS, vol. 4044, Springer, Oulu, Finland (2006).
50. Bjørnson, F. O., Vestues, K, Rolland, K. H.: Coordination in the large: a research design. In: XP2017 Scientific Workshops. ACM, (2017).
51. Ingvaldsen, J. A., Rolfsen, M.: Autonomous work groups and the challenge of inter-group coordination. Human Relations 65 (7), 861-881. (2012).
52. Jurgen Appelo, Agility Scales: Shifting Teams in Better Shapes, https://www.youtube.com/watch?v=yYzSnF8IekM, last accesses 2018/04/29.
53. Rolland, K. H., et al. Problematizing agile in the large: alternative assumptions for large-scale agile development. In: Thirty Seventh International Conference on Information Systems, Dublin (2016).
54. COBIT 4.1, ISACA, http://www.isaca.org.
55. Tarhan, A., Yilmaz, S. G.: Systematic analyses and comparison of development performance and product quality of Incremental Process and Agile Process. Information and Software Technology 56 (5), 77-494. (2014).