The Impact of Situational Context on Software Process: A Case Study of a Very Small-sized Company in the Online Advertising Domain

Görkem Giray[1], Murat Yilmaz[2], Rory V. O'Connor[3], Paul M. Clarke[3]

[1] Kokteyl Corporation, Istanbul, Turkey
gorkem.giray@kokteyl.com
[2] Çankaya University, Ankara, Turkey
myilmaz@cankaya.edu.tr
[3] School of Computing, Dublin City University, Dublin, Ireland
Lero – the Irish Software Research Centre, Limerick, Ireland
{paul.m.clarke,rory.oconnor}@dcu.ie

Abstract. A primary concern of software development is selecting a suitable methodology to implement a software project. However, this selection is affected by many factors, with evidence suggesting that a specific set of factors defines a specific situational context for a project. This situational context leads to a project-specific software process. In this paper, we report on our analysis of a very small-sized company's current software process based on a reference framework that identifies the factors of a situational context. The outcome of our case study confirms the earlier findings that a software process is highly dependent on situational factors. The company has a suitable situational context (such as very small-sized, experienced, skilled, cohesive team with low turnover) to apply agile practices and its software process is more close to an agile rather than plan-driven approach. Moreover, the company is continuously adopting its software process to the situational factors changing from project to project and over time.

Keywords: Software Process, Software Development Process, Situational Context, Situational Factor, Process Selection, Process Tailoring.

1 Introduction

Central to the entire discipline of software engineering is the concept of software process and the improvement of that process. Therefore, selecting and tailoring a software process plays a critical role that guides the transformation of ideas into software products [1] [2] [3]. A software process consists of a set of interrelated practices, each of which is a systematic and repeatable way of achieving a predefined objective [4]. There have been many practices proposed over the last 20 years, each adding some value to software engineering discipline [5]. It has now become a complex problem to tailor a suitable software process for a given specific project. In general, a software process is designed by making some decisions implicitly and using some tacit knowledge [6].

Recently, some efforts have been allocated to make the factors affecting software process explicit [7] [8] [9] and associate these factors with the practices selected [10]. A software process depends on many situational factors, which are sometimes interrelated as well. Ensuring the harmonization between a set of situational factors and a software process is a complex problem [11]. To help to solve this problem, some case studies were conducted to understand how some successful organizations achieve this harmonization [12] [13] [14]. Moreover, [10] consolidates situational factors and the corresponding software process tailoring actions proposed in software engineering literature.

In this paper, we report on our analysis of a very small-sized software development company in Turkey, named Kokteyl (which resides in Istanbul, Turkey) based on the situational factors reference framework described in Clarke et al. [7]. Software processes of small-sized companies have started to gain interest recently since small and medium sized enterprises dominate software development sector in all countries worldwide. In line with this, ISO/IEC 29110 [15] introduced the term “very small entities,” defined as “an enterprise, an organization, a department, or a project having up to 25 people” [16]. According to the Scientific and Technical Research Council of Turkey, half of the software development companies in Turkey have less than 10 employees and 35% of them have 10-50 employees. Our objective is to understand the implicit relationships between the situational factors and the software process adopted by this very small-sized company. By doing so, we aim to contribute to the development of a suite of case studies including commonalities and differences in various types of situational contexts. We believe that such a suite will help to reduce challenges in harmonizing situational context and software process.

Section 2 presents a review of situational factors in the literature. Section 3 briefly discusses the case study research method used in this work. Section 4 presents an overview of the online advertising domain, the case study company operating in this domain, and the current software process of this company. Section 5 provides our analysis of the situational factors that shape the software process of the case study company. Section 6 includes the discussion and threats to validity. Section 7 concludes the paper.

2 Situational Factors

A software development setting is a part of a bigger whole that is made up of human, technologies available for software development, external factors, etc. In such a setting, a software process guides a software development team in translating user needs into a software product [17]. There are many approaches (generally positioned within a plan-driven and agile spectrum nowadays) that form a software process. It is generally accepted that software process is shaped by considering factors which make up the situational context for a project and heavily affects the formation of a software process. Many researchers have remarked the importance of assessing some of these factors as listed in Table 1.

Table 1. A partial list of situational factors identified in the literature.

Factor	Description	References
Team experience	denotes the degree of team experience	[7] [10] [18] [19]
Team size	denotes number of members in development team	[7] [10] [18] [19] [20] [21] [22]
Management commitment	denotes the level of commitment of high-level decision makers foe project (such as senior management, project sponsor)	[7] [10] [19]
Feasibility of requirements	denotes how much feasible to implement requirements are (technologically, economically, etc.)	[7]
Volatility of requirements	denotes how much requirements are changing during the project	[7] [10] [19] [22]
Criticality	denotes the impact of failures, such as loss of human life, financial loss, etc.	[7] [10] [18] [20] [22]
Number of stakeholders	denotes the number of stakeholders in the project	[7] [10] [19]
Stakeholders availability	denotes how much stakeholders of the project are readily accessible	[7] [10] [19]

Boehm and Turner identify five critical decision factors to have a properly balanced software process in plan-driven and agile spectrum for a specific project [18]. Cockburn proposes three variations of Crystal method family based on the assessment of team size and criticality (magnitude of potential loss) [20]. Xu and Ramesh conducted an empirical study to identify the factors that play a key role in software process tailoring [19]. Ambler and Lines identify eight scaling factors a team may need to address to adapt a software process [21]. Kruchten proposes two sets of factors making up a situational context: organization level and project level factors [22]. Kalus and Kuhrmann built up a catalog of 49 factors [10].

Recently, two of the authors have produced and published an initial reference framework [7] to better guide process designers to tailor project specific processes as well as to understand the underlying decisions behind software process tailoring. The reference framework comprises 44 factors classified into 8 main categories and further elaborated into 170 sub-factors.

Using the reference framework, we have investigated the situational factors affecting the software process in the case study company. We used case study research method, which is briefly explained in the next section.

3 Research Method

The case study method has been proposed as a research method in software engineering to investigate a phenomenon in its real-life context [23]. Yin states that case study method should be used especially when the boundaries between phenomenon and context are not clearly evident [23]. As we stated before, a software process is shaped in a situational context. We consider that a case study [24] is proper research method to build a qualitative understanding of how a software process is shaped in a particular situational context [25]. We have used observation and interview techniques to collect data. We analyzed these qualitative data considering the literature and reported our results. To understand the context of this case study better, the next section briefly presents the domain, the case study company, and the software process currently adopted by the company.

4 Case Study Background

In this section, we present the important concepts of online advertising domain; introduce the case study company and its software products; and explain the software process used in the company.

4.1 Online Advertising Domain

The digital age has provided new ways for organizations for interacting with consumers. With the rise of the Internet along with the increased use of computers and mobile devices; online advertising has become one of the preferred advertising forms. Online advertising is a form of advertising, which uses the Internet to deliver marketing messages to consumers. Fig. 1 illustrates the main stakeholders in online advertising domain and the relationships among them.

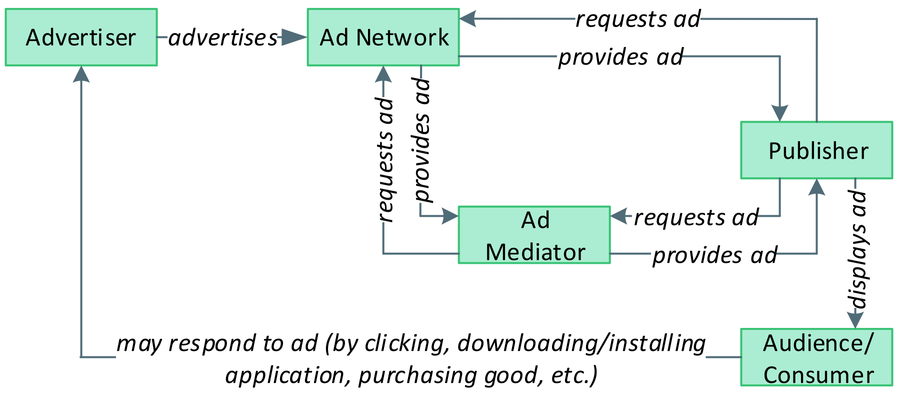

Fig. 1. Main stakeholders in the online advertising domain.

An advertiser is a person, organization or company that places advertisements to communicate a message to audiences/consumers. An ad network aggregates advertisement demand and delivers proper ads to publishers according to their requests. An ad mediator provides access to multiple ad networks (hence multiple ad demand pools) from one central platform. Publishers can maximize revenue by receiving most appropriate ads from a much larger ad demand pool through ad mediators. Moreover, the publisher can increase their fill rate, which denotes the ratio of a number of ad requests in relation to the number of ads provided. From a technical perspective, publishers integrate with only ad mediator's platform and the complexity of integrating with many ad networks is hidden behind ad mediator's platform. A publisher receives ads from ad networks and/or ad mediators and displays ads to audiences/consumers. Common publishers in the online advertisement are web sites and mobile apps. Kokteyl, which is the case study organization in this work, is operating in online advertising domain and is briefly introduced in the next sub-section.

4.2 Case Study Company

Kokteyl is a very small-sized software development company founded in 2002. Shortly after the establishment, they launched a website and a mobile application providing up-to-minute live scores and statistics for football games over the globe. In 2008, they entered to online advertisement sector and started to develop ad mediation software. Moreover, the company is developing and maintaining mobile games. Fig. 2 illustrates the software product portfolio of the company.

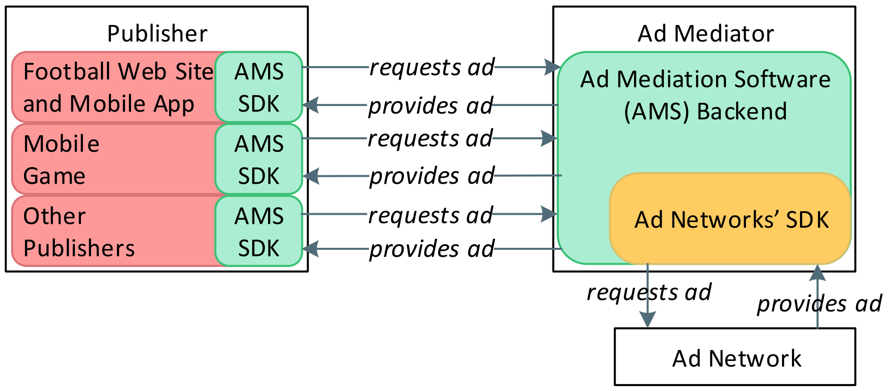

Fig. 2. Company's software product portfolio.

The company owns an ad mediation software (AMS), which has two main components, backend services and an SDK (Software Development Kit) to be integrated with the publisher's software. AMS backend is integrated with more than 40 ad networks through each ad network's SDK. Publishers can receive ads from all of these ad networks only by integrating AMS SDK (one of the benefits of using an ad mediation software as stated above).

The company also developed and maintained a football web site and mobile application with millions of users. A UK-based company has acquired this web site and mobile application in 2016. This football web site and mobile application are still using AMS for displaying ads.

The company also develops and maintains 5 mobile games, which are also publishers for AMS. There is also other publishers' software (which are the customers of Kokteyl) that are using AMS to display ads.

As of now, the company's main business is built on AMS. They are trying to convince more publishers with high page views and active users to use AMS. Their revenue is coming from commission fees out of ad revenues of these publishers.

The company employs a software development team, which are organized around two software products as shown in Table 2. Five developers are creating and maintaining AMS backend, including server-side services, data persistency, and report development. 5 developers are developing and maintaining AMS SDK for Android and iOS operating systems. Backend side and mobile app for the games are being developed by 1 and 2 developers respectively and 1 graphic designer is designing the user interfaces for the games. No team member has been assigned to develop football web site and mobile app anymore as a UK-based company acquired them.

Table 2. Team organization per software product.

Software product	Component	Number of team members
AMS	Backend	5 developers
	Android SDK	3 developers
	iOS SDK	2 developers
Mobile Games	Backend	1 developer
	Mobile app	2 developers
	UI	1 graphic designer

The team is employing a software process that is more close to agile paradigm (less focus to documentation and more focus to working software; less focus to process and more focus to individuals and collaboration; less focus to plan according to contracts and/or long-term constant requirements) as briefly explained in the next sub-section.

4.3 The Software Process of the Case Study Company

Fig. 3 illustrates the software process from eliciting requirements to deployment. For the sake of simplicity, we eliminated the iterative steps and showed the main flow of the process. This process depicts the steps taken for the implementation of one requirement; therefore the company adopts an approach much more close to agile rather than waterfall. The company does not adopt any software process standard.

The details of the process are as follows classified as the main phases of software development:

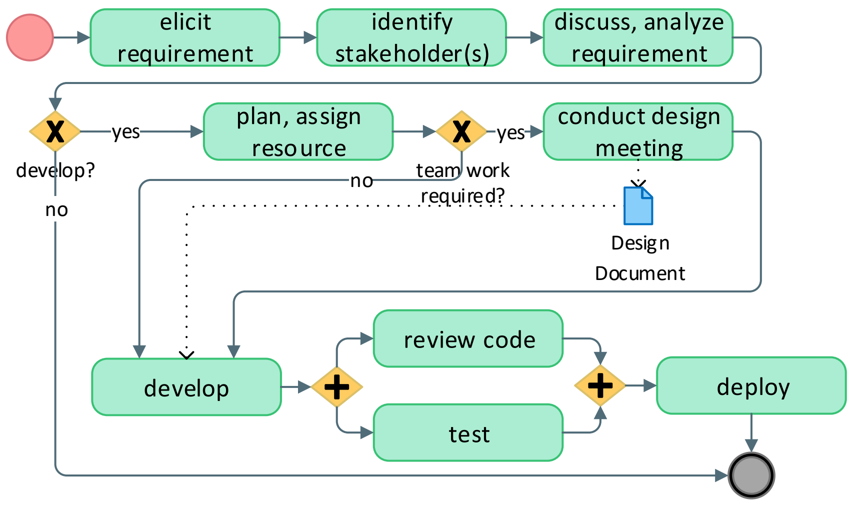

Fig. 3. High-level software process of case study organization.

Requirements engineering: The members of the team elicit a requirement from customers, competitors' products, and improvement opportunities (mainly from problems) identified by the team. The team leads identify relevant team member(s); discuss and analyze this requirement. If they consider this requirement worth developing, the required resources are assigned; otherwise, the requirement is discarded. The requirements are tracked using a project management tool, Asana, and undergo the following statuses: (1) Ice box: the requirement is not yet decided to be developed and may be discarded; (2) Backlog: the requirement is decided to be implemented; (3) In progress: the requirement is being analyzed and implemented; (4) Test: tests are being conducted; (5) Done: the relevant implementation has been completed.

Design: If the development needs team work rather than one member, a design meeting is conducted. The output of this meeting is a design document, which defines the solution. Moreover, an owner is assigned to this requirement, who takes full responsibility to involve relevant stakeholders based on the analysis and design conducted and follows up the implementation until the end of the deployment.

Development: Relevant members complete development tasks sequentially and/or in parallel.

Test: The team is organized so that there are at least two team members who are responsible for a component and code changes made by a member are reviewed by the corresponding peer. Each member conducts unit, integration, and system tests. User acceptance test is also done within the team. No regression testing practice and test automation are applied.

Deployment: The team prefers a rollout to a small group of users for evaluation and testing. At the first step, the team members receive and start to use the new version. Afterwards, the users of a mobile game (generally 100s of users) are receiving the new version. When a new version of AMS SDK is being deployed, the evaluation is being made by a restricted pool of users (1000s of users) of publishers. At the last step, the new version is deployed for all of the users.

The next section analyzes the situational factors that have led the software process explained in this section, using the reference framework [7].

5 Application of Situational Factors Reference Framework

Two techniques have been employed for applying situational factors reference framework to the case study company: (1) Observation; (2) Interview. One of the authors, who is knowledgeable about the situational factors, has worked in the company's office and conducted both silent and interactive observations [26] for 4 days. Occasionally, the author asked questions about the current software process to find out why such a process is being applied. The author also conducted two interviews with 3 team leads. The author introduced the situational factors reference framework to the leads and asked questions to them by using the sub-factors listed in the framework [7]. Table 3 lists the dominant situational factors affecting Kokteyl's software process.

Table 3. Situational factors identified in the case study.

Category	Sub-factors identified in the case study
Personnel	*Team size:* The team consists of 19 members: 13 developers, 2 analysts, 1 project manager, 1 graphic designer, 2 user support personnel.
Personnel	*Experience:* The average experience of the team is 7.2 years. The leads/senior members of the team have a working experience of more than 15 years.
Personnel	*Skill:* The team is highly skilled in software development. Each of the senior members is specialized in a key area, namely development on Android or iOS, and backend development. These senior members are providing on-the-job training and guidance to the junior members. Moreover, the team is experienced in large-scale data management, distributed computing, and concurrent computing.
Personnel	*Cohesion:* The team is working together in an open office and has successfully completed many projects in the past. A newcomer has to complete a trial period in which her/his harmony with the team is observed. As a result, the cohesion of the team is high.
Personnel	*Turnover:* Turnover, especially with key developers, is low. 5 developers out of 12 are working in the company for more than 8 years.
Organization	*Size:* With around 25 employees Kokteyl is a very small-sized organization. This allows efficient information exchange through conversations and face-to-face meetings.
Operation	*End users:* All the end users are outside the organization, mobile game users and publishers' owners. The users of AMS (publishers' owners) are strategically much more important than mobile game users since publishers are reaching thousands, millions of users. Some of the users are cooperative while some are demanding.

Table 3. Situational factors identified in the case study.

Category	Sub-factors identified in the case study
Operation	*Prerequisites:* AMS SDK is integrated with every single mobile application. Its integration should be easy and smooth.
Application	*Deployment profile:* When a new version of AMS SDK is ready, all the publishers should adapt their application accordingly and launch a new version. While some publishers update their application quickly, some prefer to stay with the old version for any reason. Therefore, AMS backend and SDK should be backward compatible with a number of old versions, which are still being used by publishers.
Application	*Component reuse:* The team prefers to reuse components if its quality has been proved. They try to validate its quality by checking who are using that component and what their comments are. If the references and comments are positive, they validate the component by using it.
Application	*Performance:* There is a significant requirement for very high performance from AMS. AMS should respond to thousands of requests per second and the response time should stay in a couple of milliseconds.
Business	*Business drivers:* The company's sustainability relies upon mainly the success of AMS, hence most of the efforts are dedicated to the improvement of this product.
Business	*Payment arrangements:* The company does not make money by delivering software products based on fixed or time and materials based contracts. The revenue relies on commission fees out of ad revenues of the publishers.
Business	*Customer satisfaction:* The strategically important customers are publishers. It is very important to provide high quality software to these publishers since they can easily switch to another as mediation product. Moreover, there are not too many publishers, therefore it is important to try to keep all of them satisfied in order not to lose revenue.

In the following section, we discuss the relationships between situational factors and software process identified in the case study company.

6 Discussion

Kokteyl can be considered as a successful very small-sized company since it has been working for software industry for 16 years and managed to develop a software product that has been acquired by a large-scale company based in the UK. Based on this business success, we can conclude that the company has applied a proper software process and managed to adapt it according to changing situational factors. Kokteyl has not designed and adapted its software process using situational factor reference framework explicitly. On the other hand, we can observe that they evaluated many situational factors and selected the proper practices implicitly.

The company does not adopt any process framework (e.g. Scrum), instead a process expert selects a set of practices to tailor a software process close to an agile approach. There are some minor differences between the practices applied by the teams developing AMS and mobile games. While mobile game development team has a fixed release cycle of 4 weeks, AMS development team does not have a fixed release cycle. Since AMS is the software product on which the company's revenue relies on, the team has to respond to new requests and problems very quickly. Therefore, they release a new version of AMS when a significant and urgent change has to be made and deployed. Not fixing the release cycles ensures to keep high customer satisfaction (AMS customers, which are strategically important).

The acquisition of football web site and mobile app by the UK-based company is an important development that has affected some of the situational factors. The company used to be a publisher with a very high traffic and was making money from publishing ads. The team was focused on developing a web site and a mobile app. They were neglecting regression testing and test automation because of their overhead. When a problem is observed, they were fixing it by updating the web site and/or sending a new version of the mobile app to the relevant store. With the AMS, the deployment has become a two-step process whose second part is done by publishers. When a new version of AMS is released, all the publishers should update their applications by including the new AMS SDK. Therefore, Kokteyl started to assess including regression testing in their software process. To decrease the overhead of regression testing, test automation is also being considered.

The customers of the football web site and mobile app used to be satisfied by providing data timely along with a good user experience. On the other hand, the development of AMS requires the design of complex algorithms whose outputs cannot be validated precisely. The team tracks realized revenues to try to understand whether they have improved an algorithm. Design of such algorithms is related to recent research areas, such as machine learning. Therefore, the team considers to include practices regarding research and development in their software.

The situational factors identified during this case study are already included in the situational factors reference framework. This is an additional evidence on the wide coverage of this framework. On the other hand, it does not seem to be practically possible to have a complete framework on situational factors, as pointed out in [7] and [10]. We aim to conduct more case studies in various sectors and try to extend the framework and improve the content, language, and structure. In addition, we can also provide scales to evaluate situational factors, such as creating a taxonomy of organization size along with intervals (i.e. very small organizations have less than 25 employees). Moreover, adding the associations between situational factors and software development processes to the framework would contribute to software engineering discipline. Such associations would especially contribute to situational method engineering discipline, whose objective is creating a software process for a specific situation [27].

Our discussion is subject to some validity threats, which are briefly described. During the data collection, we used one researcher who has conducted observations and interviews. Although we tried to mitigate this threat by crosschecking observation and interview outputs and prolonged involvement [24] (by making several visits to the case study company), there is still a potential risk that a single researcher can draw wrong conclusions. Regarding external validity, it is not possible to generalize the findings based on just this one case study. To reduce the impact of this threat, we evaluated the findings based on the literature and other case studies to explore some common patterns.

7 Conclusions and Future Work

To investigate how companies design a project-specific software process, we conducted a case study in a very small-sized software development company, named Kokteyl, in Istanbul, Turkey. The company does not adopt any process framework or standard. We have observed that they have formed a suitable software process based on the situational context the company and its projects reside. There are also some differences between the software processes used for different projects. We have identified that these differences are based on the situational factors identified in the literature [7] [10]. Moreover, the company constantly fine-tunes its software process based on the changes in the situational factors.

Further work is required to confirm and validate these findings. To this end we propose the following: (1) we will continue to conduct such case studies to form a suite of case studies to help practitioners in reducing challenges in harmonizing situational context and software process; (2) we plan to guide Kokteyl for improving their company-wide a research and development skills by analyzing changes in situational factors and proposing corresponding actions for adopting the software process accordingly.

References

1. MacCormack, A., Verganti, R.: Managing the sources of uncertainty: Matching process and context in software development. Journal of Product Innovation Management 20, 217–232 (2003).
2. Jones, C.: Development practices for small software applications. CrossTalk, The Journal of Defense Software Engineering 21(2), 9-13 (2008).
3. Rong, G., Boehm, B., Kuhrmann, M., Tian, E., Lian, S., Richardson, I.: Towards context-specific software process selection, tailoring, and composition. In Proc. of the 2014 Int. Conf. on Software and System Process (ICSSP 2014), pp. 183-184. ACM, USA (2014).
4. Jacobson, I., Ng, P.W., McMahon, P.E., Spence, I., Lidman, S.: The essence of software engineering: Applying the SEMAT Kernel. Addison-Wesley Professional (2013).
5. Kennaley, K.: SDLC 3.0: Beyond a tacit understanding of agile. Fourth Medium Press, (2010).
6. Yilmaz, M.: A software process engineering approach to understanding software productivity and team personality characteristics. PhD Thesis, Dublin City University (2013).
7. Clarke, P., O'Connor, R.V.: The situational factors that affect the software development process: towards a comprehensive reference framework. J. Inf. Softw. Technol. 54(5), 433–447 (2012).

8. Jeners, S., O'Connor, R.V., Clarke, P., Lichter, H., Lepmets, M., Buglione, L.: Harnessing software development contexts to inform software process selection decisions. Software Quality Professional 16, 35–36 (2013).
9. Ng, P., Huang, S., Wu, Y.: On the value of essence to software engineering research: A preliminary study. 2nd SEMAT Workshop on a General Theory of Software Engineering (GTSE 2013) pp. 51-58. (2013).
10. Kalus, G., Kuhrmann, M.: Criteria for software process tailoring: a systematic review. In Proceedings of the 2013 International Conference on Software and System Process - ICSSP 2013, pp. 171–180. ACM Press, New York, USA (2013).
11. Clarke, P., O'Connor, R.V., Leavy, B.: A complexity theory viewpoint on the software development process and situational context. In: Proceedings of the 2016 International Conference on Software and System Process (ICSSP 2016). (2016).
12. Marks, G., O'Connor, R.V., Clarke, P.M.: The impact of situational context on the software development process – A case study of a highly innovative start-up organization. In SPICE, vol. 155, pp. 455–466. (2017).
13. Clarke, P.M., O'Connor, R.V., Solan, D., Elger, P., Yilmaz, M., Ennis, A., Treanor, R.: Exploring software process variation arising from differences in situational context. In EuroSPI, vol. 748, pp. 29–42. (2017).
14. O'Connor, R.V., Elger, P., Clarke, P.M.: Exploring the impact of situational context – A case study of a software development process for a microservices architecture. In Proceedings of the International Workshop on Software and Systems Process - ICSSP'16, pp. 6–10. (2016).
15. O'Connor, R.V., Laporte, C.: The Evolution of the ISO/IEC 29110 Set of Standards and Guides. International Journal of Information Technologies and Systems Approach 10 (1), 1-21 (2017).
16. Larrucea, X., O'Connor, R.V., Colomo-Palacios, R., Laporte, C.Y.: Software Process Improvement in Very Small Organizations. IEEE Software, 33(2), 85–89. (2016).
17. Systems and software engineering –Vocabulary, ISO/IEC/IEEE 24765:2010(E). (2010).
18. Boehm, B., Turner, R.: Observations on balancing discipline and agility. Agile Development Conference, pp. 32-39. (2003).
19. Xu, P., Ramesh, B.: Using process tailoring to manage software development challenges. IT Professional, 10(4), pp. 39–45. (2008).
20. Cockburn, A.: Crystal Clear A human-powered methodology for small teams. Addison Wesley Professional (2004).
21. Ambler, S.W., Lines, M.: Disciplined agile delivery: A practitioner's guide to agile software delivery in the enterprise. 1st edn. IBM Press (2012).
22. Kruchten, P.: Contextualizing agile software development. Journal of Software: Evolution and Process 25(4), 351-361 (2013).
23. Yin, R.K.: Case study research - Design and methods. SAGE Publications (26) (2003).
24. Runeson, P., Host, M., Rainer, A., Regnell, B.: Case study research in software engineering: Guidelines and examples. 1st edn. Wiley Publishing (2012).
25. O'Connor, R.V.: Using grounded theory coding mechanisms to analyze case study and focus group data in the context of software process research. In Mora, M., Gelman, O., Steenkamp, A., Raisinghani M. (eds.) Research Methodologies, Innovations and Philosophies in Software Systems Engineering and Information Systems, Chapter 13, pp. 1627-1645. IGI Global (2012).
26. Wiegers, K.: Software Requirements. 3rd edn. Microsoft Press (2013).
27. Henderson-Sellers, B., Ralyté, J., Ågerfalk, P., Rossi, M.: Situational method engineering. Springer-Verlag Berlin Heidelberg (2014).