Comparing Communication Effort within the Scrum, Scrum with Kanban, XP, and Banana Development Processes

Davide Taibi, Valentina Lenarduzzi
University of Bolzano/Bozen
Bolzano/Bozen
Italy
{davide.taibi; valentina.lenarduzzi}@unibz.it

Muhammad Ovais Ahmad, Kari Liukkunen
University of Oulu
Oulu
Finland
{muhammad.ahmad, kari.liukkunen}@oulu.fi

ABSTRACT

[Context]: Communication plays an important role in any development process. However, communication overhead has been rarely compared among development processes. [Objective]: The goal of this work is to compare the communication overhead and the different channels applied in three agile processes (XP, Scrum, Scrum with Kanban) and in an unstructured process. [Method]: We designed an empirical study asking four teams to develop the same application with the four development processes, and we compare the communication overhead among them. [Results]: As expected, face-to-face communication is most frequently employed in the teams. Scrum with Kanban turned out to be the process that requires the least communication. Unexpectedly, despite requiring much more time to develop the same application, the unstructured process required comparable communication overhead (25% of the total development time) as the agile processes.

KEYWORDS

Communication, Agile Processes, Empirical Software Engineering, Case Study.

1 INTRODUCTION

Communication between developers plays a crucial role in any development process. Moreover, different communication channels and strategies may be applied in different processes [9]. In Agile methods, different means are used for communication among developers and customers, such as retrospective meetings or stand-up meetings. Regarding communication among developers, different communication channels can be used for synchronous and asynchronous communication. However, to the best of our knowledge, no empirical comparison has ever been conducted regarding the communication overhead of different communication strategies and channels among different agile and non-agile methods.

Therefore, we designed an empirical study with the goal of comparing the communication overhead (ratio between development time and communication time). in three agile methods, namely Scrum, Scrum with Kanban, and Extreme Programming, and that of an ad-hoc process ("Banana"). We define communication time as any communication time the developers spent for discussing technical aspects related to the development tasks they carried out.

The study was designed as a multiple case study with replication design, performed with four groups of last-year master students as participants and the same application to be developed by all four teams. The teams developed exactly the same application. The requirements were proposed by the same entrepreneur, and the teams elicited them independently.

The results show that communication time is comparable among the different teams, except for the team applying Scrum with Kanban, where communication only amounted to 6% of the whole development effort. Moreover, despite much more time spent on the development of the same application, the communication overhead of the ad-hoc team was comparable to the overhead of the other two agile teams. The remainder of this paper is structured as follows. Section 2 introduces the background and related work. Section 3 presents the multiple case study and Section 4 the results obtained. Section 5 discusses results and shows the threats to validity and Section 6 draws conclusions and highlights future works.

2 BACKGROUND AND RELATED WORKS

Agile methods are based on constant communication and on sharing information on the project's development among the whole team [19], so communication plays an important role. In Agile, there is a need for continuous and active communication with the customer and the team members [17][3]. Therefore, effective communication among developers, operations, support, customers, management, and business areas is one of the most important factors for project success [20][4]. Communication in Agile can be classified as internal and external. The internal communication process involves the developers and project leaders, while external communication takes place between the development team and the stakeholders [11]. Bhalerao and Ingle [17] classify communication on three levels, defining the "primary level", involving customers and team members and

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org. EASE'17, June 15-16, 2017, Karlskrona, Sweden
© 2017 ACM ISBN 978-1-4503-4804-1/17/06…$15.00
http://dx.doi.org/10.1145/3084226.3084270

aimed at gathering and understanding the requirements; a "mid-iteration", involving team members and sometimes customers so as to reduce any possible requirements ambiguity, and an "end iteration" aimed at providing instant feedback and adding requirements details among team members.

The communication process can be either active or passive. Active communication is the physical and synchronous kind of communication, including face-to-face meetings, telephone conversations, and video conferencing, while passive communication is asynchronous and carried out via mail or by reading documentation [17]. Several works have investigated the role of communication in Agile processes [7][8][9][10] even though empirical investigations are not so common. The majority of the studies emphasize the strategic role of communication as a key success factor [12][13][14]. In two studies [13][14], Mishra et al. [25] analyzed the communication effects related to different team distribution or team size, reporting that an appropriate workspace environment, such as the proximity of rooms, the presence of whiteboards, and a common area, generate a positive effect on communication, especially in small teams. Abbas et al. [15] investigated the effectiveness of agile practices, finding that good communication increases the quality of the software developed. Moreover, the studies [13][14][18] confirm that synchronous communication such as face-to-face meetings with and without a whiteboard play an important role, from the point of view of both internal and external communication, by providing instant feedback, whereas videoconferencing and teleconference calls are not considered so important from both sides. Taking into account asynchronous communication, online chat is barely used and if so, only among developers and almost never with stakeholders. Pikkarainen et al. [11] conducted a case study to investigate internal and external communication, reporting that agile practices improve the communication process even though, in the case of large development teams and multiple external stakeholders, some communication problems may occur. Moreover, they suggest selecting an adequate communication strategy to avoid these communication problems. Estler et al [29] presented a case study comparing agile with. structured distributed software development reporting a higher communication effort required by the distributed development while recently, Storey et al [30] present a study presenting the different influence of social communication channels in software development.

In Table 1, we summarize the most common communication strategies reported in the literature.

Table 1: Communication strategies in Agile

Communication Strategy	
Synchronous	Asynchronous
• Face-to-face • Face-to-face at whiteboard • Online chat • Videoconference • Telephone call	• Email • Documentation (reading and writing)

3 THE MULTIPLE CASE STUDY

We aim to compare the communication effort in three agile development processes and one unstructured development process. Therefore, we designed this study as a multiple case study with a replication design [1].

As depicted in Fig. 1, we first identify the goal and the research questions, we select the cases and we identify the data collection protocol. Then we conduct the studies independently and we write individual case reports so as to draw cross-case conclusions.

In this section, we present the study process adopted. We first describe the study goal, questions and metrics. Then we describe the study design and the study execution.

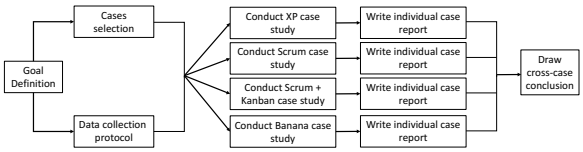

Fig. 1. The Study Process (adapted from [1])

3.1 Study Goal and Metrics

According to our expectation, we formulated the goal of the case study following the GQM approach [2] as follows:

Analyze the communication process
For the purpose of comparing
With respect to its effort
From the point of view of software developers
In the context of agile and unstructured development processes.

This leads to the following research questions:

Q1: Which development process requires more communication time?

We expect that Agile processes, because of their nature, require more communication time than non-Agile ones. Moreover, we expect that developers working in Scrum and XP will spend more time for communicating than those working in Scrum with Kanban since the Kanban board is supposed to speed-up the identification of tasks to do and in progress. Moreover, we also expect developers working with Banana process will spent less communication time, because they should not make daily and stand-up meetings.

Based on the aforementioned hypotheses, we identify the following metrics:

M1.1 Synchronous communication time

M.1.1.1 Communication time during group meetings (retrospectives, stand-up meetings…) (hours)

M1.1.2 Face-to-face communication to support each other in solving project issues (hours)

Please, note that pair programming was not considered as face-to-face communication time.

M1.1.3 Online (chat) communication to support each other in solving project issues (hours)

M1.2 Asynchronous communication time (hours). Time spent for writing emails or GitHub issues to ask for technical support to other team members.

Q2: Which development process has the highest communication overhead?
In this case, we expect all the processes will have similar overhead.

M2.1 Development effort (hours)

M2.2 Total effort. Includes development and communication effort.

M2.3 Communication overhead. We define communication overhead as ratio between development time and communication time.

3.2 Study Design

In order to compare the communication time among different processes, we designed this study as multiple case study with replication design [1]. We involved four development teams in the context of a web application development, asking them to develop the same application, with the same requirements, as reported in Fig. 2.

The study population was composed of master students in computer science from two universities that are part of the Software Factory Network [6]: University of Bolzano-Bozen (Italy) and University of Oulu (Finland). The population had a very similar background since they were all master students in computer science and both universities have similar programs due to the shared European Master in Software Engineering (EMSE) [21]. All the students have knowledge in Agile software development and in all the processes adopted in the study. The software factory is the perfect environment for conducting empirical studies with next generation developers, since participants represents the typical developers entering to the job-market[6]. This is also confirmed by the results of several empirical studies conducted in the past [4], [5], [16], [24], [26], [27]). The groups were defined in a random manner without influencing the study results.

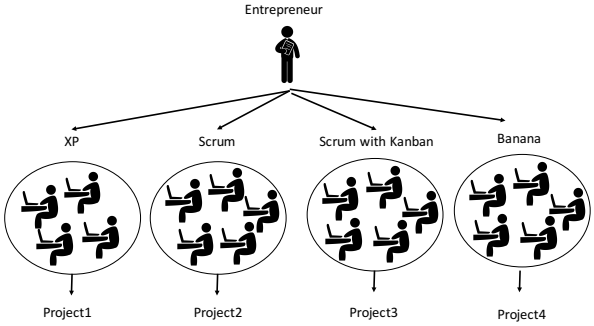

Fig. 2. Study Design

3.2.1 Study Preparation

We randomly assigned the developers to the different development processes. However, we took care to have at least one experienced developer in each group and to avoid to have a group composed only by experienced developers.

One group developed in Scrum, one in Scrum with the support of a Kanbanboard (Scrum+Kanban), one in Extreme Programming (XP) and another one in an ad-hoc process we call "Banana" as commonly used term among practitioners to describe processes that produce immature products, which have to "ripen" after being shipped to the customer, like bananas.

Teams were required to work collocated, in a dedicated room, for minimum of 250 total working hours per person.

The four groups had to develop the same application proposed by the same entrepreneur, who acted as product owner. The entrepreneur was a designer, with no experience in software development that made her available for all the teams with the same effort.

The teams were required to develop an Android application (Serendipity). The idea was selected in a contest for entrepreneurs, where entrepreneurs were asked to submit the minimum viable product [28] description of their project ideas that could be implemented in the software factory lab.

Serendipity is an Android application and a web application used to share a set of sounds in a specific location, aimed at recalling special moments by listening to the sounds. The entrepreneur initially defined the project idea as:

> "*Serendipity means "fortunate happenstance" or "pleasant surprise". This project is meant to be an experience that mixes places and sound to enable you to see places you usually go to with new eyes, in a more poetic, more ecstatic way. While taking walk, you will have access to six music tracks, developed from the actual ambient sound of those places themselves. I specifically chose very popular meeting points in my town (Bolzano), where many people go without even realizing anymore what the place looks like. On a map displayed on your smartphone, these locations are highlighted. When you arrive there, you can listen to the soundtrack created to allow you to enjoy the moment. It should be a discovery process. The perk is that this concept is applicable to any city/place – it would be nice to spread it and let the sound go local*".

We scheduled the communication process with the entrepreneur during the retrospective meetings (or general meeting for the banana process), so as to have another variable under our control. Therefore, the entrepreneur described the project to the groups during the same timeframe in which the requirements were elicited independently.

The two researchers also ensured that the teams implemented the processes they were supposed to use attending all the meetings and attending more than 75% of the development as observer.

3.3 Study Execution

The teams developed the project during the Software Factory Lab of the two universities. We informed the participants about

the study and the usage of the collected data. The four teams worked on the project at different times: one team at the University of Bolzano-Bozen from October 2015 until the end of January 2016, and three teams at the University of Oulu from February 2016 to the end of April 2016. The minimum development effort was fixed in 250 hours with iterations every two weeks for all groups.

The Kanban group was composed of five master students who developed in Scrum with Kanban. The Scrum group was composed of five master students who developed in Scrum. The XP group was composed of four master students who developed in Extreme Programming (XP) while the "Banana" group was composed of five master students who developed with an unstructured process.

In order to ensure the correct succession of requirements and to prevent the development of the previous project in Bolzano from influencing the entrepreneur's perception of her project, we recorded the audio of every meeting, transcribing every requirement elicited in Bolzano so as to ask the entrepreneur to request the same things in the same timeframe, without giving away any details to the other teams.

Each team independently decomposed the requirements from the entrepreneur with a different set of user stories or tasks. Moreover, two researchers attended the requirements elicitation meetings and reported the user stories and tasks associated to each requirement.

3.2.2 Data collection and analysis

The measures reported in Section 3.1 were collected during meetings and during the development process. During the meeting, the researchers took track of the communication time (between developers and between developers and the entrepreneur) while during the development the developers kept track of the communication time spent. Data were collected on a shared online spreadsheet. The collected measures on the processes are compared based on sums, medians, and averages.

4 STUDY RESULTS

In this Section, we present results related to our research questions.

In **Q1** we aimed at comparing the communication time among development processes. All the teams except the Scrum one spent 100% of the communication time on synchronous channels, while the Scrum team also spent 12.5% communicating asynchronously via email. No technical communication on GitHub or other platforms have been used. As reported in Table 2, the teams mainly communicated during group meetings. Developers generally prefer to discuss problems in group. Only in the XP and Scrum processes developers communicated in a one-on-one way, by means of face-to-face meetings or via chat or videoconference. Figure 3 depicts the percentage of time spent by each team in the different communication channels.

Taking into account the total communication time (Table 3), unexpectedly the Banana team spent a huge amount of time for communication - 17 times more than the Scrum+Kanban team - compared to the Agile processes. As expected, the team working in Scrum+Kanban needed less communication than those working in Scrum and XP. Moreover, as expected, Scrum and XP teams spent a very similar communication effort.

In **Q2** we aimed at identifying which development process has the highest communication overhead. Unexpectedly, besides the application to be implemented had exactly the same requirements, the total effort spent to develop the application by the different teams was very different, ranging from a total of 340 hours for the Scrum+Kanban team to 1491 hours for the Banana one (Table 3). This result has an obvious impact on the communication overhead. As a consequence, as reported in Table 3 and Figure 4, the communication time ranges from 15 to 25% in all processes, except for the Scrum with Kanban process, which only took 6.55% of the total communication overhead.

Table 2. Communication effort

Team	Synchronous Communication			Asynchronous Communication
	Group Meetings	*Face-to-face (one on one)*	*Chat*	*Email/GitHub*
XP	23.50	55.30	16.10	0
Scrum	56	9	14.80	11.40
Scrum + Kanban	22.30	0	0	0
Banana	375	0	0	0

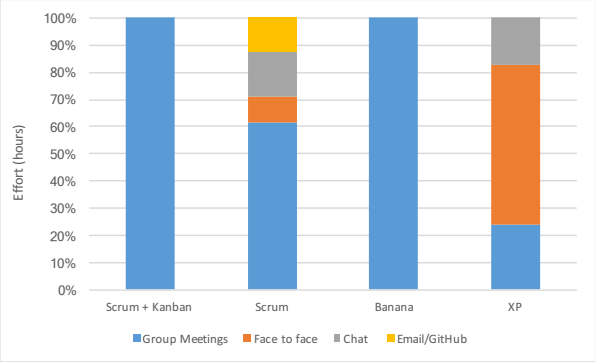

Fig. 3. Communication effort per team

Table 3. Development and communication effort

Team	Effort		
	Total Effort (hours)	**Development (hours)**	**Communication (hours)**
XP	486.4	391.50	94.9 (19.51%)
Scrum	579.2	488	91.2 (15.75%)
Scrum + Kanban	340.3	318	22.3 (6.55%)
Banana	1491	1116	375 (25.15%)

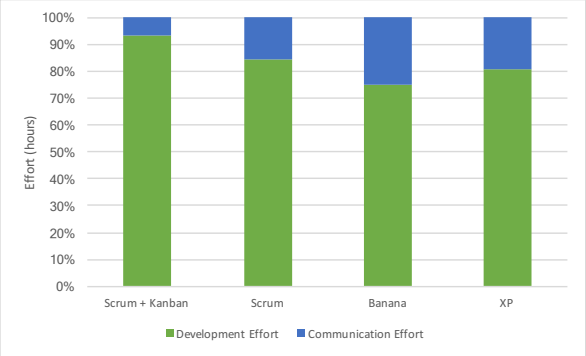

Fig. 4. Communication vs development effort

5 DISCUSSION AND THREATS TO VALIDITY

As confirmed by the obtained results, the communication overhead is not influenced by the development processes. However, the total development time varied dramatically as the team developing in Scrum with Kanban delivered the project with less effort, while the team working with the Banana process took more than 3.5 times as long.

As for the compliance of the selected development processes, the XP team developed with a test-driven development approach, while the two Scrum teams (Scrum and Scrum with Kanban) developed test cases during the process. No test cases were developed by Banana team. As result of the observation of the processes, researchers confirm that the three agile processes were carefully followed. Moreover, as expected, the banana team did not have structured the process. They simply elicited the requirements from the entrepreneur, and started to develop the application in a process that resemble an unstructured waterfall one. They first defined the architecture of the system. Then a developer started working only on the database and on the connection between the code and the database, one developer worked only on the graphical user interface and the other developers on all the remaining tasks. As expected, the banana team was the last one who deliver the first prototype to the entrepreneur, taking more than two months, while the teams working with an agile process delivered the first prototype, implementing only a small set of features, only after four weeks. This result should not be related to the communication strategies but to the process itself.

As a side benefit, as confirmed also by the literature [23], developers reported that the usage of the Kanban Board helped to speed up the process and to understand the whole project backlog because of the simpler graphical representation.

5.1 Threat to Validity

Concerning the internal validity, the subjects were trained in the use of the processes before the studies started. All the teams were composed of master students sufficiently motivated since the project to be developed was part of the lecture program. From our observations, we did not notice any big difference among students. However, beside we tried to select students in the second year of the master, students may have different levels of experience, which may have influenced their communication needs. Students personality could have influenced the results. Different types of personalities may behave differently in the tasks they are assigned, and hence the communication behavior of the group may differ [22]. Even if we tried to provide the same environment and criteria to students, additional confounding factors may have influenced them (motivation to study, grading criteria, etc.). As the teams were small, 4-5 persons only, the background differences in experience and in motivation could have affected the communication. Moreover, the conformance to the processes has been verified by two researchers, so as to assure that the constructs on the usage of the specified processes can be validated.

As for external validity, the software developed was a real project, not a toy project as commonly used in studies involving students, with requirements defined by a real entrepreneur.

Regarding the conclusion validity, the multiple case study was designed by different experts on empirical studies and it was ensured that the subjects of both groups had similar backgrounds and knowledge regarding software development.

Concerning the construct validity, we followed the Goal Question Metric (GQM) approach [2] to define the goal, the questions, and the relative metrics. The goal was refined into clearly defined metrics to avoid misunderstandings.

6 CONCLUSION AND FUTURE WORKS

Communication among developers plays an important role with regard to project success. Several works propose classifications of communication channels and approaches but, to the best of our knowledge, no studies have conducted empirical studies on the comparison of communication techniques among different projects.

For this purpose, we designed an empirical study with next generation developers as participants (master students in their last year). We asked four development teams to develop the same project, whose requirements elicited from an external entrepreneur, applying three agile methods and an unstructured process.

The results show that, unexpectedly, the effort overhead (ratio among communication effort (hours) and development effort (hours)) of the unstructured process is comparable to the one reported by the teams working in Agile.

Analyzing the different communication strategies adopted, all the teams mainly communicated in person. However, while the team working with Extreme Programming had a lot of one-on-one communication to solve project issues, the team working with Scrum with Kanban preferred group meetings and reduced communication as much as possible. This result could be caused by the availability of a dedicated room for the development. Therefore, developers mainly worked collocated and did not need to communicate offline.

It is important to notice that the communication taking place may not be considered as negative, in particular in a learning environment.

This was a preliminary study on the communication effort overhead in agile teams. Although we attempted to keep threats to validity under control by randomizing the participants of the study and controlling the external variables, we are aware that it might not be possible to generalize the results because of the small sample of the study. The research method adopted was the best compromise to keep external variables under control. Time constraints and the limited availability of the participants did not allow us to perform a controlled experiment. The four processes we used in this study are well known and adopted in industry. However, beside the lower communication time required by the agile processes, the results of this study also confirm that agile processes allow to deliver software in a short timeframe.

With this study, we contribute to the body of knowledge by providing the first empirical study comparing the communication effort among agile and non-agile development processes.

Future works include a closer analysis of why some processes need more communication than others, by identifying the strategies which improved and corrupted communication. Therefore, we look forward to replicate this study in a controlled environment and with experienced developers, reducing the scope of the study to analyze specific methods or techniques more than complete methodologies in order to isolate better the different factors.

ACKNOWLEDGMENTS

This work was partly supported by TEKES (Finnish Funding Agency for Technology and Innovation) as part of HILLA program and by the project SQuaSME "recommendation techniques for Software QUAlity improvement in Small Medium Enterprise" funded by the Free University of Bozen-Bolzano.

REFERENCES

[1] Yin R.K., Case Study Research: Design and Methods (Applied Social Research Methods Vol. 5). Sage. ISBN-13: 978-1452242569.

[2] Basili V.R., Caldiera G., Rombach H.D., The goal question metric approach. In Encyclopedia of software engineering, pp. 528–532. (1994)

[3] Del Bianco V., Lavazza, L. , Lenarduzzi, V. , Morasca, S. , Taibi, D. , and Tosi, D. "A Study on OSS Marketing and Communication Strategies", Int. Conference on Open Source Software, OSS 2012 DOI: 10.1007/978-3-642-33442-9_31

[4] Lenarduzzi V., Lunesu, I. , Matta, M. , and Taibi, D. , "Functional Size Measures and Effort Estimation in Agile Development: a Replicated Study", in XP2015, 2015. DOI: 10.1007/978-3-319-18612-2_9

[5] Diebold P., Dieudonné, L. , and Taibi, D. , "Process Configuration Framework Tool", in 39th Euromicro Conference on Software Engineering and Advanced Applications , 2014.

[6] Taibi, D., Lenarduzzi, V., Ahmad, O.M., Liukkunen, K. Lunesu, I., Matta, M., Fagerholm, F., Münch, J., Pietinen, S., Tukiainen, M., Fernández-Sánchez, C., Garbajosa, J., Systä, K. Free Innovation Environments: Lessons learned from the Software Factory Initiatives. In ICSEA 2015 The Tenth International Conference on Software Engineering Advances, pp. 25-30. (2015).

[7] Korkala M., Abrahamsson P., and Kyllonen P., A case study on the impact of customer communication on defects in agile software development. AGILE 2006, pp. 76-88, 2006.

[8] Melnik, G., and Maurer, F., Direct Verbal Communication as a Catalyst of Agile Knowledge Sharing. AGILE 2004, pp.21-31, 2004.

[9] Sarker, S., and Sarker, S., Exploring Agility in Distributed Information Systems Development Teams: An Interpretive Study in an Offshoring Context. Information Systems Research, Vol.20(3), pp.440-461, 2009.

[10] Wang, X., Conboy, K., and Pikkarainen, M., Assimilation of agile practices in use. Information Systems Journal. Vol 22(6), pp. 435-455, 2012

[11] Pikkarainen, M., Haikara, J., Salo, O., Abrahamsson, P., and Still, J., The impact of agile practices on communication in software development. Empirical Software Engineering. Vol. 13(3), pp. 303-337, 2008

[12] Koskela, J., and Abrahamsson, P., On-Site Customer in an XP Project: Empirical Results from a Case Study. Torgeir Dingsøyr (Ed.) Software Process Improvement, Springer, Berlin Heidelberg, pp.1-11, 2004.

[13] Mishra, D., and Mishra, A., Effective communication, collaboration, and coordination in eXtreme Programming: Human-centric perspective in a small organization. Human Factors and Ergonomics in Manufacturing & Service Industries. Vol 19(5), pp.438-456, 2009.

[14] Mishra, D., Mishra, A., and Ostrovska, S., Impact of physical ambiance on communication, collaboration and coordination in agile software development: An empirical evaluation. Information and Software Technology. Vol 54(10), pp.1067-1078, 2012.

[15] Abbas, N., Gravell, A. M., and Wills, G. B., Using Factor Analysis to Generate Clusters of Agile Practices (A Guide for Agile Process Improvement). AGILE 2010, pp.11-20, 2010.

[16] Lenarduzzi V., Morasca S. , and Taibi, D. , "Estimating Software Development Effort Based on Phases", in 39th Euromicro Conference on Software Engineering and Advanced Applications , 2014. DOI: 10.1109/SEAA.2014.54

[17] Bhalerao, S., Puntambekar, D. and Ingle, M., Generalized agile software development life cycle. International Journal of Computer Science and Engineering. Vol I(3), 2009.

[18] Ambysoft, Agile Principles and Practices survey Results: July 2008. http://www.ambysoft.com/surveys/practicesPrinciples2008.html

[19] Turner, R. and Boehm, B., Balancing Agility and Discipline: A Guide for the Perplexed. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA. 2003

[20] Ambler, S., Quality in an agile world. Software Quality Professional. Vol 7(4), pp. 34-40, 2005.

[21] European Master in Software Engineering. Available online: http://em-se.eu/

[22] Kosti M.V., Feldt R., Angelis L. "Personality, emotional intelligence and work preferences in software engineering: An empirical study." Information & Software Technology 56(8): 973-990, 2014

[23] Ahmad, M.O., Markkula, J. and Oivo, M. Kanban in software development: A systematic literature review. 39th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA), 2013, pp. 9-16, 2013.

[24] Taibi, Lenarduzzi, V. , Janes, A. , Liukkunen, K. , and Ahmad, M. Ovais, "Comparing Requirements Decomposition Within the Scrum, Scrum with Kanban, XP, and Banana Development Processes", Agile Processes in Software Engineering and Extreme Programming: 18th International Conference, XP 2017, Cologne, Germany, May 22-26, 2017, Karlskrona (Sweden), pp. 68–83, 2017. DOI: 10.1007/978-3-319-57633-6_5

[25] Misra, S. C., Kumar, V., and Kumar, U., Identifying some important success factors in adopting agile software development practices. Journal of Systems and Software. Vol 82(11), pp. 1869-1890, 2009.

[26] Taibi D., Lenarduzzi, V. , Diebold, P. , and Lunesu, I. , "Operationalizing the Experience Factory for Effort Estimation in Agile Processes", in 21th Evaluation and Assessment in Software Engineeering (EASE), 2017 DOI: 10.1145/3084226.3084240.

[27] Lavazza L., Morasca, S. , Taibi, D. , and Tosi, D. , "Applying SCRUM in an OSS Development Process: An Empirical Evaluation", in 11th International Conference in Software Engineering and Extreme Programming (XP2010), Trondheim, Norway, June 1-4, 2010. Proceedings, 2010, pp. 147-159. DOI 10.1007/978-3-642-13054-0_11

[28] Taibi D., and Lenarduzzi, V., "MVP explained: A Systematic Mapping on the Definition of Minimum Viable Product", in SEAA2016 42th Euromicro Conference on Software Engineering and Advanced Applications 2016, Cyprus, 2016 DOI: 10.1109/SEAA.2016.56

[29] H. C. Estler, M. Nordio, C. A. Furia, B. Meyer and J. Schneider, "Agile vs. structured distributed software development: A case study", Empir. Software Eng. Vol. 19(5), pp. 1197-1224, 2014

[30] M. A. Storey, A. Zagalsky, F. F. Filho, L. Singer and D. M. German, "How Social and Communication Channels Shape and Challenge a Participatory Culture in Software Development," IEEE Transactions on Software Engineering. Vol. 43(2), pp. 185-204, 2017.