

Is There an Optimal Sprint Length on Agile Software Development Projects?

Nicolas Nascimento^a, Alan Santos^b, Afonso Sales^c and Rafael Chanin^d

Polytechnical School, Pontifical Catholic University of Rio Grande do Sul, Avenida Ipiranga 6681, Porto Alegre, Brazil

Keywords: Software Engineering, Agile, Sprint Length.

Abstract: Agile software development is adopted by the industry as a way to develop applications while also remaining flexible to quickly respond and adapt. At its core, agile relies heavily upon time-constrained iterations, usually named “sprints”, which should provide the development team to deliver a functional version of a software product. This study aims at understanding what is the impact of different sprint lengths in agile software teams. In order to achieve it, we have conducted a field study at a mobile software development course for eight months. The course was organized on three stages, where at each stage ten projects were simultaneously conducted. Data collection was based on project outcome including daily logs and deliverables generated by the teams. Each stage had a different sprint length (1-week, 2-week, or 3-week iterations). Our results indicate that there are differences in some aspects, including project evaluation and weekly impediments. These differences were statistically analyzed regarding the impacts of different sprint lengths in agile teams. Further, we have also observed some correlation between weekly impediments and project evaluation, providing indications of a possible impact on overall projects outcome.

1 INTRODUCTION

The growing demand for faster time to market cycles has shaped the way software products are managed, developed and delivered to users over the years. There is demand on the software industry for shorter software development cycles (Bourque and Fairley, 2018). In this context, traditional project management approaches, including waterfall, have been replaced by agile methodologies, *e.g.*, Scrum, Extreme Programming, Kanban, among others (Hasnain, 2010). Agile methodologies tend to reduce time to market cycles because they are based upon lean principles which foster agility and waste reduction (Tore and Torgeir, 2008; Santos et al., 2013).

At its core, agile relies heavily upon time-constrained iterations, usually named *Sprints*, which should provide the development team to deliver a functional version of a software product. The Scrum guide (Schwaber and Sutherland, 2011), for example, states that *Sprints* are its *heartbeat*, where “*ideas are*


turned into value”. Furthermore, the guide provides a directive that sprints can vary in length, never exceeding one month. As a result, the consequences of different sprint durations are still an open research topic as very few research studies target this.


The contribution of this paper is an empirical field study that evaluates the use of different sprint lengths at an agile software development environment, evaluating the impacts and influences of different sprint lengths structure on software development issues, impediments and deliverables.


This paper is organized as follows: in Section 2 we explore important concepts and the background for this research. Section 3 depicts the methodology used in this study. In Section 4 we explore our results, followed by Section 5, in which we take away the most important insights. Section 6 presents the limitations of this study. Finally, Section 7 concludes the paper and indicates next steps and future works.


2 BACKGROUND

In modern software development, change is a constant and it is often caused by external and uncontrollable factors (Barry et al., 2002). As markets and

^a  <https://orcid.org/0000-0002-0080-8822>

^b  <https://orcid.org/0000-0001-8323-3472>

^c  <https://orcid.org/0000-0001-6962-3706>

^d  <https://orcid.org/0000-0002-6293-7419>

economies quickly and unpredictably change, traditional software development practices, tool and techniques become difficult to apply and to follow appropriately. In addition, these changes impact on the software development project, which commonly grows both in scope and cost, a phenomenon known as *Scope Creep* (Barry et al., 2002). This results in high costs for the development, maintenance and update of software products, thus reducing the competitiveness of software development companies.

In this context, as faster and more flexible software development techniques became more necessary, in 2001, the “*Manifesto for Agile Software Development*” (Beck et al., 2001) was created. As core principles for software development, the *Manifesto* proposes valuing:

- *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.

Thus, Agile Methodologies, as defined by Pressman (Pressman, 2005), are a modern approach to develop software. By embracing continuous change, emphasizing quick deployment of functional software and relying heavily on a close collaboration with the customer during the development lifecycle, agile is suitable to be applied in a variety of projects, even outside of the software development realm.

Although the *Manifesto* states that less value is put in the processes used for developing software, Agile has some commonly applied development approaches. Some of the most commonly applied are *Extreme Programming*, *Feature-Driven Development* and *Scrum* (Anand and Dinakaran, 2016).

Extreme Programming (XP), as well as other agile methods were presented as an approach for software development projects (Conboy and Fitzgerald, 2010). XP is a software development approach which emphasizes productivity, flexibility, informality, teamwork, a limited usage of tech outside of programming and working in short cycles, where each cycle begins by choosing requirements from a backlog (Macias et al., 2003). Beck (Beck and Andres, 2004) presented XP as a lightweight methodology for small and medium size software development teams which deal with vague and fast-changing requirements.

There are many agile methods and, according to recent research performed by Version One with 3,925 participants from North America and Europe (One, 2015), Scrum methods and practices are the used by

majority of the industry. Mariz *et al.* (de Souza Mariz et al., 2010) presented an investigation the relationship among agile practices and the success of projects using Scrum. In this investigation, with 62 software engineers which were associated with 11 projects in nine different companies, results show that eight out of 25 attributes which are associated with agile practices have a significant correlation with project success, suggesting that it is important to consider agile as a way to improve efficacy in projects in the software industry.

Scrum is an agile, iterative and incremental software development approach (Schwaber, 2004) which is also used in complex projects, in which is impossible to predict how everything will occur. Further, Scrum offers a framework and set of practices which keep things visible, allowing teams to know exactly what is happening and adjust as needed to maintain project progress. As Scrum practices are part of an iterative and incremental process, the output of each iteration is a product increment, and the iterations repeat until the project is completed or is terminated. Each iteration in Scrum is called a sprint and lasts from one to four weeks. There are three main artefacts in Scrum: Product Backlog, Sprint Backlog and a potentially functional product increment (Schwaber, 2004). The Product Backlog is a priority-ordered requirements list, usually written as user stories. The Sprint Backlog is a sprint subset of the Product Backlog which is organized during the planning of a sprint, where user stories are identified and implemented according to their priority. In Scrum, activities are estimated in hours by teams (Reichlmayr, 2011), however this approach is not the only option, as there are others such as complexity points.

During each sprint, daily meetings are conducted. In these meetings, each team member lets the team know what he/she has done the previous day, what will be done in the current day and what impediments are blocking development. At the end of each sprint, a product demonstration called Sprint Review held and after this review, a meeting to discuss lessons learned and next steps called Sprint Retrospective (Scharff and Verma, 2010).

In agile software development, given the context of the software industry, there are some relevant aspect that have to be considered, and also in software development more broadly, such as the task of constantly manage people. In this sense, *e.g.* People Management (de Alcântara et al., 2018), an approach which understand people involved in a team or an organization as human beings, plays a fundamental role of effectively adding to the organization effectiveness. Further, It is also important to note that a team’s ex-

expertise and technical ability, specially in larger teams, has to be aligned to encompass the design of the final software product (Grabis et al., 2016). Finally, agile is a methodology that requires modification to the fundamentals of building software, and so it requires a transition strategy when applied to traditional software development environments (Bider and Söderberg, 2016).

3 METHODOLOGY

The research question proposed in this study was:

“What are the effects of using different sprint lengths on agile software development on project performance, amount of impediments and amount of issues?”

In this section, we depict the steps undertaken to address the research question.

3.1 Field Study Protocol

A field study was conducted at a mobile software development course during a period of eight months. The goal of this field study was to empirically understand the influences and impacts of different sprint lengths performing data analysis of sprint plannings, daily meetings and sprints deliverables.

In a general sense, field studies do not generalize obtained results. Rather, they allow researchers to illustrate a particular phenomenon in its original context (Hall, 2008).

The goal of the field study conducted in this research was to assess the influence of sprint length on agile software development. The sample population (50 students) was composed of undergraduate students in a mobile application development program, who were chosen using the convenience criteria due to the fact that participants were selected for their availability. The sample population size was defined using the higher number of available people to participate in this study.

3.2 Data Collection and Analysis

For our field study, we have considered two data points:

1. Teams deliverables;
2. Daily meetings logs;

3.2.1 Team Deliverables

To evaluate the team deliverables, throughout the stages, instructors performed an assessment of each

individual project students had worked on. This assessment revolved around three main points:

1. The final presentation delivered by the each team;
2. The challenge level addressed by each team;
3. The team collaboration during the execution of the project.

Considering these points, each instructor provided an individual evaluation which ranged from 1 to 5, following a Likert-type scale. This evaluation would be equivalent to:

1. Terrible performance;
2. Poor performance;
3. Average performance;
4. Good performance;
5. Amazing performance.

Using the instructors individual assessment, to have single assessment per team, we have obtained the average evaluation of all projects performed by the students.

3.2.2 Daily Meeting Logs

As the goal of the study was to assess the impacts and influences of sprint duration in agile projects, we have used the daily meeting logs of each team. These daily meeting logs were records of the daily meeting the team were performing during the agile development process. In this daily meeting log, each student would answer the following questions:

1. *“What did I do yesterday?”*
2. *“What will I do today?”*
3. *“What are my current impediments?”*

For the purpose of our study, we decided to focus primarily in the third question, regarding impediments. As such, we have split impediments in two different types:

1. **Impediments:** These were impediments which were technical and revolved around the actual implementation of the project. A real example from the logs which was classified as an impediment was *“problems with the backend debugging tool”*.
2. **Issues:** These were impediments caused by external factors, such as illnesses, hard weather conditions or external appointments / limitations.

3.3 Field Study Stages

The field study consolidated data from three stages. Each stage collected data from 10 different teams during the process of agile software development. Each stage differed from the others in sprint duration and project length. Stages setup are presented at Figure 1

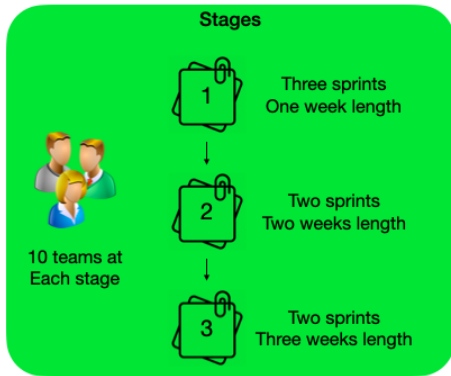


Figure 1: Stages setup.

In this context, Stage 1 was composed by three sprints of one-week length; Stage 2 was composed by two sprints of two-week length; and Stage 3 was composed by two sprints of three-week length.

After the end of each stage, new teams compositions were done for the next stage in order to avoid team bias. Once the data collection was completed, we have prepared and categorized all data points obtained, generating the data analysis foundation. All data was stored at Airtable database¹.

4 RESULTS

This section depicts the field study results from Stage 1, Stage 2, and Stage 3. For each stage, we present the average evaluation score, impediments and issues found by 10 teams. Further, to account for the different durations of each project, we have also used a “weekly” metric for the number of impediments and issues as a normalization strategy.

4.1 Stage 1

Figure 2 presents the first stage data, running three sprints of one week length each. On this configuration setting, majority of teams had a score 4+ on a scale from 1 (worst) to 5 (best). Four teams had more than 5 impediments, however from these four teams, only one team (Team J) scored less than 4 points out of 5.

¹<https://airtable.com>

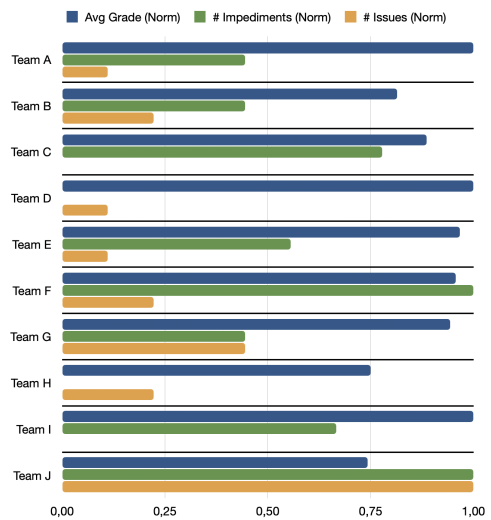


Figure 2: Stage 1 Results.

Some results were translated to weekly values (impediments and issues) in order to enable further comparison between the different stages, using different sprint lengths and different number of sprints. Table 4 describes the overall Stage 1 results.

Table 1: Stage 1 Data.

Aspect	Result
Avg score	4.5
Avg # of impediments	4.8
Avg # of issues	2.2
Avg # of weekly impediments	1.6
Avg # of weekly issues	0.7

During Stage 1, running three sprints of one week each (Figure 3), we have found on average 1.6 impediments per team per week and 0.7 issues per team per week. Only 30% of Stage 1 teams had up to 4 impediments total. Despite the low number of impediments from those teams, their scores have not presented a significant difference when compared with teams that had higher number of impediments.

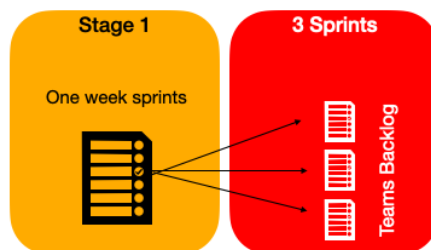


Figure 3: Stage 1 Setting.

Running sprints of one week length, we have found that Team H had the lowest score and have not reported any impediments. Furthermore, it had a small number of issues as well. This could indicate that team H had difficulty to perceive impediments and issues in the sprints.

4.2 Stage 2

Figure 4 presents the second stage data, running two sprints of two weeks length each. In this configuration setting, the majority of teams had a score of more than four on a scale from 1 (worst) to 5 (best), except by two teams (Team G and Team H). Half teams had more than five impediments, however from these teams, only one team (Team G) scored less than 4 points out of 5. Table 2 describes the overall Stage 2 results.

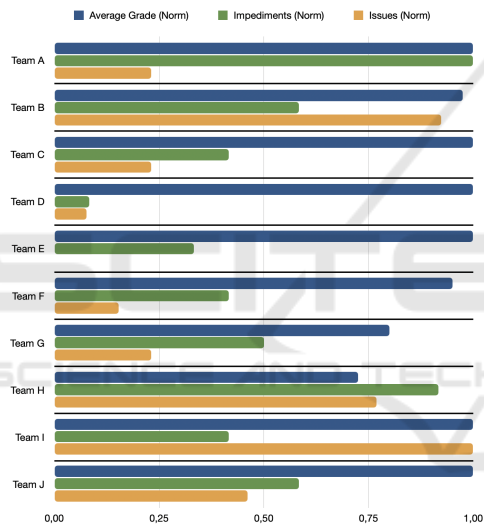


Figure 4: Stage 2 Results.

During Stage 2, running two sprints of two weeks each (Figure 5), we have found on average 1.6 impediments per team per week (not different from Stage 1) and 1.6 issues per team per week (the double when compared to Stage 1). Another difference when compared to Stage 1 is the overall average number of impediments (6.3). It has increased by 23%. However, the sprints length was twice as big (two weeks x one week).

Two teams (Team A and Team H) had a high number of impediments (10 impediments each). However, Team A got the highest score (5 out of 5) and Team H had an average score (3 out of 5). This score difference was caused because the number of Team H issues (60% more than team A). Teams with high number of impediments and high number of issues will not score as a team with high number of impediments and

Table 2: Stage 2 Data.

Aspect	Result
Avg score	4.7
Avg # of impediments	6.3
Avg # of issues	5.3
Avg # of weekly impediments	1.6
Avg # of weekly issues	1.3

low number of issues. The conjunction of the number of impediments plus the number of issues will determine a team success on a configuration of two weeks length sprints.

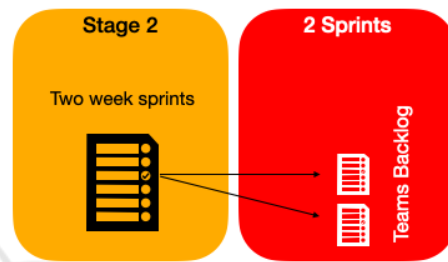


Figure 5: Stage 2 Setting.

4.3 Stage 3

Figure 6 presents the third and the last stage data, running two sprints of three weeks length each. On this configuration setting, 40% of teams had 10 impediments each, however it had no impact on their score 4+ on a scale from 1 (worst) to 5 (best). Three teams had 10 issues each, however it also had no impact on their score 4+ on a scale from 1 (worst) to 5 (best).

Table 3 describe the overall Stage 3 results.

Table 3: Stage 3 Data.

Aspect	Result
Avg score	4.6
Avg # of impediments	8.2
Avg # of issues	8.4
Avg # of weekly impediments	1.4
Avg # of weekly issues	1.4

During Stage 3, running two sprints of three weeks each (Figure 7), the team with the lowest score (Team I) reported a low number of impediments and no issues. Similar to what was found on Stage 1, a team with low number of issues and impediments had the lowest score. This could indicate that reporting a low number of impediments could be related to not perceiving problems during a sprint.

We have not found difference on the average score

Table 4: Statistics Per Team in Each Stage.

Variable	Stage 1	Stage 2	Stage 3
Evaluation score	4.5 ± 1.4 (5)	4.7 ± 1.5 (5)	4.6 ± 1.5 (5)
Impediments	4.8 ± 3.2 (9)	6.3 ± 3.2 (12)	8.2 ± 6.3 (19)
Issues	2.2 ± 2.7 (9)	5.3 ± 4.7 (13)	8.4 ± 8.4 (27)
Weekly impediments	1.6 ± 1.1 (3)	1.6 ± 0.8 (3)	1.4 ± 1.1 (3.17)
Weekly issues	0.7 ± 0.9 (3)	1.3 ± 1.2 (3.25)	1.4 ± 1.4 (4.5)

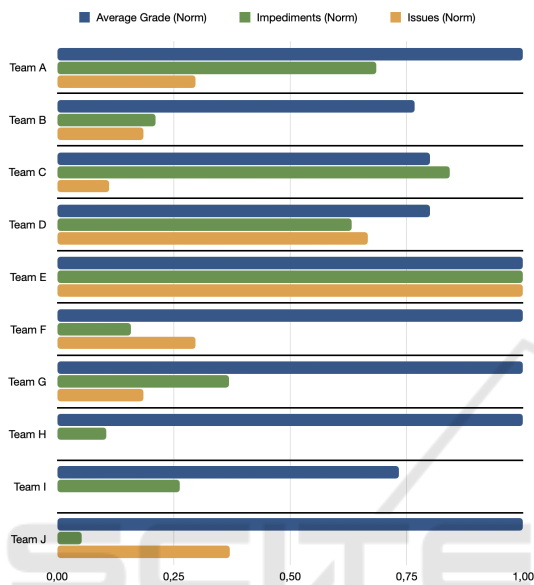


Figure 6: Stage 3 Results.

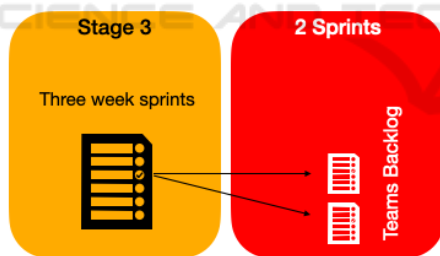


Figure 7: Stage 3 Setting.

of 10 different teams in the three different stages. Regardless of sprint length composition, the average teams score was almost the same across stages. Besides, no significant difference was found on the average number of weekly impediments and average number of weekly issues.

4.4 Combined Results Analysis

Other than the analysis of each stage individually, statistics regarding all three stages are presented in Table 4. For each variable in a stage, we present the average value per team, the standard deviation and the absolute maximum value.

After running a set of unpaired *t* tests for each variable in the different stages, we could not find significant statistical difference in the obtained results. In this context, the differences presented at this first analysis set can only serve as indicatives for future studies.

Moreover, we decided to seek correlation between the evaluation provided by the instructors, weekly impediments and weekly issues. To achieve this, we have used Pearson’s correlation coefficient, which presents the strength and direction of the relationship between two variables. The results obtained are presented in Table 5, where we present the *r*-value obtained and the significance level (*p*-value).

These results from Table 5 indicate that for $p < 0.05$, we have had correlations in two stages:

1. *Stage 2*: A strong correlation between instructors evaluation and weekly impediments and weekly issues.
2. *Stage 3*: A strong correlation between instructors evaluation and weekly impediments.

All other correlations cannot be guaranteed for $p < 0.05$ and thus have been considered not statistically significant.

5 DISCUSSION

Given the results found in the study, we have sought to answer the proposed research question: “*What are the effects of using different sprint lengths on agile software development on project performance, amount of impediments and amount of issues?*”

Our first analysis revolves around the team statistics on each stage. We have observed differences in some aspects, such as evaluation and weekly impediments. These differences, however, have not been statistically significant and can only be seen as indicatives regarding the impacts of different sprint lengths in agile teams. Thus, the insights from this data point, presented in Table 4, indicate that the sprint length does not impact significantly the evaluation, weekly issues and weekly impediments in the context under study.

Table 5: Correlation Results.

Evaluation Correlation	Weekly impediments	Weekly issues
Stage 1 (p-value)	-0.0355 (0.923)	-0.6079 (0.063)
Stage 2 (p-value)	0.6757 (0.032)	0.6874 (0.028)
Stage 3 (p-value)	0.6364 (0.048)	0.3971 (0.256)

The second analysis focuses on understanding how correlation between evaluation and both weekly issues and impediments changes when different sprint lengths are applied in agile software development. In this scenario, we have found two interesting correlations. Those happened during Stage 2, between evaluation and both weekly impediments and issues, and during Stage 3, between evaluation and weekly impediments. Although causality can not be assured, this correlation is an indicative that some influence could be present.

At Stage 2, results indicate that instructors were more likely to positively evaluate team which both had more weekly impediments and issues. In terms of impediments, this could indicate that, when using two-week sprints, teams which could identify implementation problems (and thus reported in the daily meeting log) were more likely to solve these problems. It is difficult to assess specifically if this was the case, but it is a possibility, nonetheless.

Finally, on Stage 3, results indicate a positive correlation with weekly impediments. As this result is similar to the one found during Stage 2, where a correlation between evaluation and impediments was found, it could further connect these two variables. On this context, this result provides an indicative that perceiving failure could be crucial to solving it.

6 LIMITATIONS

Our study was conducted with a limited number of respondents and from the same mobile development program. In addition, our results are drawn based on participants generated data points and all inferences and analysis rely on the validity of these data points. Therefore, results reported in this study are dependent on the participants' honesty, perceptiveness and judgment.

Another limitation of this study could be related to experience level of participants. As students were part of a mobile development course, it is expected that the experience of students could have an impact their performance.

The translation and coding processes were manually performed. Even ensuring they were correctly executed, errors could have been made and conse-

quently influenced results. An additional aspect to consider is that all projects were executed sequentially during the stages and participants were developing software in an education context. On this context, it is possible that the experience from the previous projects somehow improved their development skills, changing their development practices and thus have influenced the results.

7 CONCLUSION

This paper presented results from a field study conducted during eight months with students from a mobile application development course. It has performed data collection from 10 different projects at three different research stages on a total of 30 projects.

As results from quantitative analysis, we have found a strong correlation between instructors evaluation and weekly impediments and weekly issues. Also statistically, we have found a strong correlation between instructors evaluation and weekly impediments. These results were analyzed through a quantitative perspective and were reinforced by looking at previous works conducted on similar topics.

Preliminary results present indicatives that sprint length do not have a direct impact on project evaluation. Generalization of these results would require more research. In addition, we have also found indicatives that weekly issues and impediments could be related to projects evaluation outcome.

As a future work, we will conduct this same research in an industry environment with more experienced developers. To achieve this, we intend to run a qualitative analysis combined with a quantitative analysis in order to further investigate whether our findings are concise.

REFERENCES

- Anand, R. V. and Dinakaran, M. (2016). Popular agile methods in software development: Review and analysis. *International Journal of Applied Engineering Research*, 11(5):3433–3437.
- Barry, E. J., Mukhopadhyay, T., and Slaughter, S. A. (2002). Software project duration and effort: an empirical

- study. *Information Technology and Management*, 3(1-2):113–136.
- Beck, K. and Andres, C. (2004). *Extreme Programming Explained: Embrace Change (2Nd Edition)*. Addison-Wesley Professional.
- Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., et al. (2001). Manifesto for agile software development.
- Bider, I. and Söderberg, O. (2016). Becoming agile in a non-disruptive way-is it possible? In *ICEIS (1)*, pages 294–305.
- Bourque, P. and Fairley, R. (2018). *SWEBOK v3.0: Guide to the Software Engineering Body of Knowledge*. IEEE Computer Society Products and Services.
- Conboy, K. and Fitzgerald, B. (2010). Method and Developer Characteristics for Effective Agile Method Tailoring: A Study of XP Expert Opinion. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 20(1):47–77.
- de Alcântara, P. T. R., Canedo, E. D., and da Costa, R. P. (2018). People management in agile development. In *ICEIS (2)*, pages 92–103.
- de Souza Mariz, L., Franca, and da Silva, F. (2010). An Empirical Study on the Relationship between the Use of Agile Practices and the Success of Software Projects that Use Scrum. In *Software Engineering (SBES), 2010 Brazilian Symposium on*, pages 110–117.
- Grabis, J., Meiers, E., Supulniece, I., Berzisa, S., Ozolins, E., and Svaza, A. (2016). Aligning software design with development team expertise. In *ICEIS (1)*, pages 560–565.
- Hall, R. (2008). *Applied Social Research : Planning, Designing and Conducting Real-World Research*. palgrave macmillan.
- Hasnain, E. (2010). An Overview of Published Agile Studies: A Systematic Literature Review. In *Proceedings of the 2010 National Software Engineering Conference, NSEC '10*, pages 3:1–3:6, Rawalpindi, Pakistan. ACM.
- Macias, F., Holcombe, M., and Gheorghe, M. (2003). A Formal Experiment Comparing Extreme Programming with Traditional Software Construction. In *Proceedings of the Fourth Mexican International Conference on Computer Science*, pages 73–80.
- One, V. (2015). 9th Annual State of Agile Survey. Technical report, Version One.
- Pressman, R. S. (2005). *Software engineering: a practitioner's approach*, chapter 3, pages 65–93. Palgrave Macmillan, New York, NY, USA.
- Reichlmayr, T. (2011). Working towards the student Scrum - Developing Agile Android applications. *ASEE Annual Conference and Exposition, Conference Proceedings*.
- Santos, V., Goldman, A., and Roriz Filho, H. (2013). The Influence of Practices Adopted by Agile Coaching and Training to Foster Interaction and Knowledge Sharing in Organizational Practices. In *46th Hawaii International Conference on System Sciences (HICSS)*, pages 4852–4861.
- Scharff, C. and Verma, R. (2010). Scrum to Support Mobile Application Development Projects in a Just-in-time Learning Context. *Proceedings - International Conference on Software Engineering*, pages 25–31.
- Schwaber, K. (2004). *Agile Project Management with Scrum*. Microsoft Press, USA, 1st edition.
- Schwaber, K. and Sutherland, J. (2011). The scrum guide. *Scrum Alliance*, 21.
- Tore, D. and Torgeir, D. (2008). Empirical studies of agile software development: A systematic review. *Information and Software Technology*, 50(9:10):833 – 859.