Proceedings of the Federated Conference on Computer Science and Information Systems pp. 969–973
DOI: 10.15439/2018F53
ISSN 2300-5963 ACSIS, Vol. 15

Agile to Lean Software Development Transformation: a Systematic Literature Review

Filip Kišš and Bruno Rossi
Faculty of Informatics
Masaryk University, Brno, Czech Republic
Email: {390917,brossi}@mail.muni.cz

Abstract*—*Context: **Lean development has been often proposed as an adaptation to agile for scaling-up to larger contexts.** ***Goals:*** **we wanted to better understand the "agile-to-lean" transformation, in terms of: i) reported benefits, ii) challenges faced, iii) metrics used.** ***Method:*** **we performed a Systematic Literature Review (SLR) about "agile-to-lean" transformations.** ***Results:*** **reduced lead time, improved flow, continuous improvement, and improved defect fix rate were the main reported benefits. Adaptation to lean thinking, teaching the lean mindset, identification of the concept of waste, and scaling flexibility were the main challenges. Lead time was the most reported metric.**

I. Introduction

NOWADAYS, many software organizations use agile methodologies for their software development processes, finding benefits for process improvement [1]–[3]. Lean software development [4] has been used to optimize development processes, mainly due to the concept of waste reduction involved in the optimization of all activities producing inefficiencies [4]. This view is complementary to agile principles, more focused on all activities that create value for the customer.

The term lean software development originated from the work of Mary and Tom Poppendieck [4]. Lean software development can be characterized as a combination of lean manufacturing with the lean IT principles and their application into software development [5]. This approach is driven by a series of seven principles: eliminate waste, decide as late as possible, amplify learning, deliver as fast as possible, empower the team, build integrity in, and see the whole [4], [5].

Many authors suggest that lean thinking can be used as guiding principles to implement and adopt agile development practices [4]. Wang [1] has identified and analyzed various combinations of lean and agile as reported by previous research: lean principles can either be used to adapt existing agile practices or to scale-up the agile software development practices [1]. However, there is no universal type of "agile-lean" combination that can be used for every situation [1].

The goal of this paper is to identify benefits, challenges, and metrics used in *"agile-to-lean"* transformations. Identification of such factors can allow to better understand *"agile-to-lean"* transformations, providing more evidence about how such transformation is happening.

II. SLR

A Systematic Literature Review (SLR) is a process which summarizes, organizes, and documents previous research of a field in a systematic way [6]. To conduct the review on "agile-to-lean" transformations, we followed the SLR guidelines by Kitchenham and Charters [6].

A. Needs for an SLR

"Agile-to-lean" is a less explored research area compared to "waterfall-to-agile" (e.g., Middleton [7]). There are other SLRs performed on lean methodologies but they are focused either at the business level [8], or at the level of metrics used in lean / agile software projects within industry [9]. One literature review on the "agile-to-lean" transformation [10], focused on categorizing and comparing the transformations including 30 experience reports. Compared to the study by Wang et al. [10], our study is more focused on benefits, challenges and metrics.

B. Research questions

RQ1. *What are the **benefits** that have been reported after the adoption of lean principles in the context of an ongoing agile development process?*

RQ2. *What are the **challenges** that have been reported after the adoption of lean principles in the context of an ongoing agile development process?*

RQ3. *Which **metrics** have been used to measure the "agile-to-lean" transformation?*

C. Search strategy & study selection

We have collected research papers available online in three digital repositories: Web of Science (WoS), IEEExplore, ACM Digital Library (DL) (on 2017-06-10, Fig. 1). We used *"lean software development OR agile transformation OR lean transformation"* as search string, as we preferred to start with a more general query and filter out results later.

The first stage of search strategy (automated search in online databases) included only studies written in English. The EndNote reference manager software was used for excluding duplicates and narrowing down the initial search results from 1,787 to 856 research papers (Fig. 1). We have included journals and conference papers published since 2003, after the work of Mary and Tom Poppendieck [4].

In the second stage (filtering based on title and abstract), multiple search criteria have been applied such as the exclusion of papers that involved lean manufacturing or any other subject areas outside software engineering. A total of 131 papers have passed the second stage of the search strategy (Fig. 1).

Stage three (full-text filtering) was performed manually going through the remaining entries. 18 papers were included based on full text reading. Quality criteria (section II-D) were applied to find papers involving organizational transformation from agile to lean. After quality assessment, a total of 8 papers were included in the final SLR list (Fig. 1).

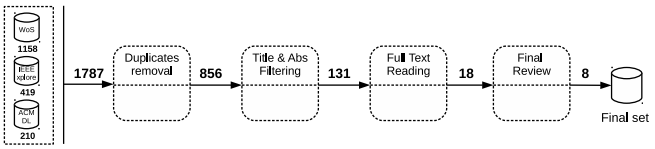

Fig. 1. SLR papers selection process

D. Study quality assessment criteria

After 18 papers were selected (Table I), we have conducted a quality assessment process to select the papers fitting the exact purpose of this research:

- C1. *Is the agile methodology at the initial state of the transformation?*
- C2. *Is the reported company making a transformation towards lean software development practices?*
- C3. *Does the paper provide metrics used for the transformation and states clear outcomes?*

Papers were given points from 0 (meaning *"no"*), 0.5 (*"partly"*) to 1 (*"yes"*). All papers reaching the score marked as ≥ 2.5 for $sum(C1, C2, C3)$ (Table I) were included for conducting the literature review (Table II).

TABLE I
QUALITY ASSESSMENT OF THE SELECTED PAPERS

SLR	Article	C1	C2	C3	Score
S1	Hayata et al. [11]	yes	no	no	1.0
S2	Jakobsen and Poppendieck [12]	yes	yes	yes	3.0
S3	Kuusela and Koivuluoma [13]	yes	partly	no	1.5
S4	Middleton and Joyce [2]	yes	yes	yes	3.0
S5	Misaghi and Bosnic [14]	yes	yes	yes	3.0
S6	Paasivaara et al. [15]	no	yes	yes	2.0
S7	Perera and Fernando [16]	yes	yes	yes	3.0
S8	Petersen and Wohlin [17]	yes	yes	yes	3.0
S9	Rodríguez et al. [18]	no	yes	yes	2.0
S10	Rodríguez et al.[19]	yes	yes	yes	3.0
S11	Samanta et al. [20]	-	yes	yes	2.0
S12	Schnitter and Mackert [21]	yes	partly	partly	2.0
S13	Sjøberg et al. [22]	yes	no	partly	1.5
S14	Swaminathan and Jain [23]	yes	yes	yes	3.0
S15	Trimble and Webster [24]	no	yes	no	1.0
S16	Vilkki [25]	no	yes	no	1.0
S17	Viswanath [26]	no	yes	yes	2.0
S18	Walter et al. [27]	yes	yes	partly	2.5

III. SLR RESULTS

A. Benefits (RQ1)

To answer our first research question (RQ1, Fig. 2), we have reviewed and categorized the reported benefits.

1) Reduced lead time: The most common benefit that has been reported is reduced lead time as it was found in six studies [2], [16], [17], [19], [23], [27]. Middleton and Joyce [2] describe lead time as the total time recorded from a customer request to the completed work delivery. Reducing

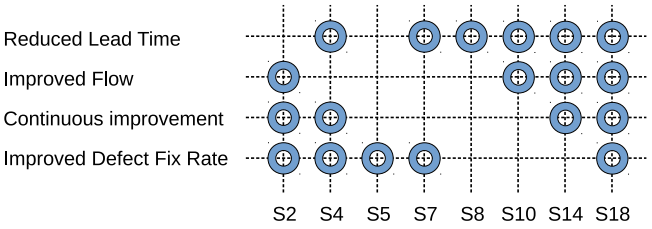

Fig. 2. Benefits mapped to papers. O = benefit reported in the paper

lead times contributes to flow improvement so that activities are organized continuously, enabling smooth deliveries to the customer. Middleton and Joyce [2] also state that they have experienced 47% less variance and on average 37% shorter lead time to deliver software, which is a significant improvement. Moreover, Walter et al. [27] have recorded an enhancement of approximately 70% which can dramatically increase the time-to-market responsiveness. Early feedback from the customer implies frequent integration, an important factor to improve the software product quality.

2) Improved flow: Improved flow was reported in four cases [12], [19], [23], [27]. Walter et al. [27] have discovered that the secret to improve the flow is to control Work-in-Progress (WiP) items. By reducing the number of simultaneous tasks they have reached lean flow state with constant throughput. Flow can be seen as the number of WiP items, so that lowering the number of such items can speed-up the whole process and more features can be implemented and eventually delivered. Therefore, to speed-up the flow, it is essential to remove waste in the inventory to avoid piling up user stories. Rodríguez et al. [19], Middleton and Joyce [2] state that limiting WiP is an important element for achieving flow. Limited WiP leads to more organized activities so that the improved flow can lead to smooth deliveries [19]. Additionally, continuous integration and test automation significantly supported the flow by frequent and smaller builds [19]. According to Swaminathan and Jain [23], tracking and acting on the visual indicators provided by the cumulative flow diagram helped to maintain a uniform flow. Moreover, it also helped to identify and remove bottlenecks and improve the efficiency of the process. This increase in efficiency also enforced rapid development and continuous improvement to software delivery [23]. Jakobsen and Poppendieck [12] have improved the flow of story implementation from 30% to 60%. The flow was improved in various areas such as test, development, project start-up and customer activities related to contracting and ongoing clarifications, major benefits for the company.

3) Continuous improvement: Four studies [2], [12], [23], [27] experienced continuous improvement as a benefit of their transformation. In Swaminathan and Jain [23], the story rate per iteration was used as a metric to measure continuous improvement, which was proved by the evidence of cumulating story points. On the other hand, the basis for the continuous improvement in Walter et al. [27] was established by letting each team self-organize and set their own WiP size.

Continuous improvement carried out on a daily basis showed a significant increase in the software delivery pre-

TABLE II
PRIMARY STUDIES LINKED TO THE REFERENCES WITH COUNTRY AND DOMAIN OF THE CASE STUDY/EXPERIMENT

Paper ID	Reference	Domain	Type	Year	Country
S2	Jakobsen and Poppendieck [12]	Software company: complex and critical IT solutions	case study	2005	Denmark
S4	Middleton and Joyce [2]	Webmedia department software processes	case study	2009	UK
S5	Misaghi and Bosnic [14]	Software company: leading supplier of systems for the supply chain	case study	2011	Brazil
S7	Perera and Fernando [16]	Students groups during len-to-agile transformation	experiment	2007	Sri Lanka
S8	Petersen and Wohlin [17]	Large provider of ICT to service providers	case study	2009	Sweden
S10	Rodríguez et al. [19]	Wireless Embedded Systems	case study	2010	Finland
S14	Swaminathan and Jain [23]	Multinational IT consulting organization	case study	2012	India
S18	Walter et al. [27]	Software development for big telecommunication companies	case study	2015	Brazil

dictability [2]. Moreover, data collected over a twelve months period showed significant improvement as the time taken to resolve issues was reduced by 81%.

4) Improved defect fix rate: Adopting lean principles to an ongoing agile process has reflected in improved defect fix rate in five papers [2], [12], [14], [16], [27]. Lean promotes finding and fixing defects early, so there is a better and control over quality from the beginning. Therefore, continuous integration tools are often widely used in lean software development paradigm. Middleton and Joyce [2] not only managed to fix issues in a shorter period, but also experienced a lower amount of bugs. As the bug rate decreased, the team had reportedly more time for completing customer stories. Better product quality was reported by Misaghi and Bosnic [14] with the reduction of time spent on bug fixing. Similar to the study by Middleton and Joyce [2], fewer errors were released because of the higher amount of time for new features and improvements [14]. Another success story of enhanced software quality has been achieved with 50% bug reduction as reported by Walter et al. [27]. Two experimental groups for measuring defect rate had been observed during the experiment by Perera and Fernando [16]. One group adopted the combination of lean and agile, the other one just pure agile methodology. The pure agile group experienced a lower amount of defects at the beginning. However, this situation swapped at later stages and the lean-agile sample had a minimal defects rate [16]. A higher number of defects for the agile group was probably due to unfixed hidden defects at the early stages.

B. Challenges (RQ2)

In general, it was difficult to extract information about the challenges from the papers (RQ2, Fig. 3), as controversial/negative aspects might be omitted from papers.

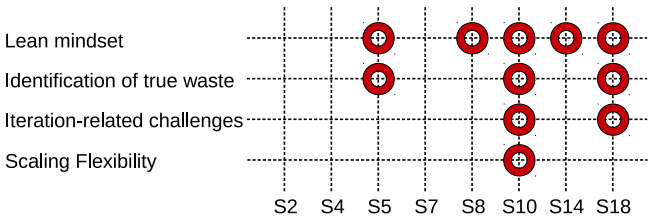

Fig. 3. Challenges mapped to papers. O = metric reported in the paper

An ”agile-to-lean” transformation brings a series of challenges. Maintaining process visibility, managing sustainability, and communication among teams are seen often as key challenges [5]. The improvement of the testing process is also reported as a key challenge, as the driving principle is perfectionism and identification of root-causes for software defects [5]. As such, placing lean on top of agile brings even more importance to the testing process. Acquiring the proper mindset can be seen as a challenge in lean adoption [14], [20], as lean requires a different mindset compared to the application of pure agile practices. Another challenge of ”agile-to-lean” is to integrate the concepts of waste minimization and quality improvements that are part of the lean philosophy, bringing to a more complete development and management process [16]. Lean on top of agile brings more the focus on the end-to-end value flow of the whole development process, thus putting lots of emphasis on different tools and their support, like value stream mapping or Kanban [17]. Constant management of flow is also a relevant issue [27], together with building a lean mindset towards defects reduction [23]. Scaling flexibility, business management involvement and waste reduction were found as challenges, with scaling flexibility problematic due to the management of the whole value streaming, making flexibility more difficult to reach [19].

1) Adapting to the Lean Mindset: In general, resistance to change is a common problem when trying to adopt new ways of working. This was the case in the reviewed studies [14], [17], [19], [23], [27]. Therefore, getting the commitment to this new paradigm is required from the longtime perspective as it is a continuous process that needs to be enforced. Misaghi and Bosnic [14] state that defining the criteria to implement the *”lean mindset”* into the organization is a main challenge. On the other hand, even though there are some tough challenges at the beginning to get the team to think end-to-end and work in a new way, once the team starts to see the added value, such way becomes naturally accepted within the team [23].

2) Identification of ”true waste”: Waste reduction is the key principle to maintain when working with a lean mindset. Rodríguez et al. [19], Walter et al. [27] found that it can be hard to identify *”what a true waste is”* within the organization and it may be even more challenging to eliminate such waste. Although setting-up teams and establishing self-organization within teams have not been hard to achieve, scaling flexibility and involvement of business management tasks were much more challenging in the lean way of working [14]. Aspects like people multitasking, which may seem at first appropriate to be more efficient, can also be seen as a waste as they might be the major cause slowing down productivity [14].

3) Iteration-related challenges: Studies have faced challenges also with coaching, estimates, pair programming, all during the run of development iterations [27]. In Walter et al. [27], team coaches had difficulties to ensure that tasks were delivered on time and with quality. For this reason, the coaching process adopted some more visible indicators (physical flags). Formation of pair programming couples and iteration estimations were adapted to the needs of the process [27]. In Rodríguez et al. [19], teams complained about too long feedback loops, caused by the involvement of business management in the whole value stream mapping process.

4) Scaling Flexibility: Scaling flexibility was defined as the easiness of performing changes during the software development process and was found as one of the key challenges in Rodríguez et al. [19]. The issue in "agile-to-lean" is that flexibility needs to pass through the whole value stream, making the process more complex than in pure agile contexts [19].

C. Metrics (RQ3)

To answer our third research question (RQ3, Fig. 4), we have reviewed and categorized findings of metrics that have been used to measure the lean transformation.

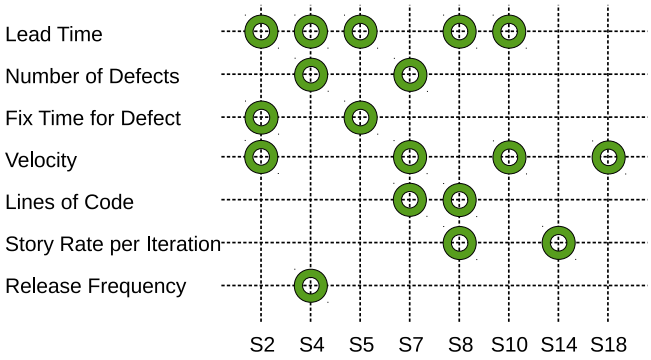

Fig. 4. Metrics mapped to papers. O = metric reported in the paper

1) Lead time: Five studies [2], [12], [14], [17], [19] have used lead time as one of the metrics to measure the progress of their organizational transformation. As we have described earlier (Section III-A about the benefits), lead time represents the duration from the customer request until the time the product is shipped to the customer. Lead time is usually measured in working days and it is stopped when user acceptance testing is complete and the product is ready for release [2]. Petersen and Wohlin [17] achieved higher responsiveness to customer needs by means of reduced lead time, managing to decrease time for delivery. Petersen and Wohlin [17] claim that this metric is highly important as the customer often needs frequent changes. The ability to respond to these changes quickly is a powerful competitive advantage. Along with this metric, Middleton and Joyce [2] reported using also cycle time, that can be considered as a sub-value of lead time. Cycle time is the time from actual initiation of the feature development (start of the working on the item) until the work on the feature is completed.

2) Number of defects: The number of defects per given time period was used as a metric in two studies [2], [16]. The main focus of Perera and Fernando [16] was on minimizing the number of defects. However, this study points out that higher number of defects at the early stages is expected, as described in section III-A4. Therefore, number of defects need to be measured with a long time perspective in mind. Middleton and Joyce [2] were measuring the number of defects per week for a twelve-month time period. The mean numbers of bugs open each week of their issue tracking system declined [2].

3) Fix time for defect: Similarly to the previous metric, two other studies [12], [14] examined defects from the fixing time perspective. Misaghi and Bosnic [14] have observed that, as the time spent on developing new features increased, the time spent on bug fixing decreased. The overall quality of the product improved as the releases contained fewer defects [14]. Measurements have been observed for the period of one year, that shows the positive long-term effect of lean.

4) Velocity: We have found this metric in four studies [12], [16], [19], [27]. Velocity can be measured by dividing the expected time of task completion by the actual time the task has been closed. Walter et al. [27] have improved their velocity by categorizing the time estimates using different sizes (extra-small, small, medium, big, extra-big). The number of hours was estimated based on their historical values and added to each category [27]. In Jakobsen and Poppendieck [12], velocity was measured as a sub-flow for story implementation, ensuring that the stories were developed in a smooth flow, eliminating the waste associated with context shifts and handovers. To verify whether the project is achieving the goals related to its schedule, expected work and actual work levels were also used by Perera and Fernando [16]. However, this time the metric was slightly modified by dividing the divergence between actual and expected work level with the expected work level [16].

5) Lines of code: Two studies [16], [17] have been measuring lean performance by evaluating the number of Lines of Code (LoC) developed. The outcome of the study by Perera and Fernando [16] was that the hybrid lean-agile approach produced more LoCs. Perera and Fernando [16] evaluated this metric from various perspectives such as new LoCs, removed LoCs and changed LoCs. On the other hand, Petersen and Wohlin [17] used this metric in a slightly different way, by measuring value efficiency: the difference between the value of output and value of input within a given time-frame [17].

6) Story rate per iteration: Customer requirements are captured in the form of user stories, which are afterwards estimated and prioritized [28]. For visualizing the long-term effect with this metric, data have been displayed mostly in story flow diagrams and cumulative story flow diagrams to measure continuous improvement from the longtime perspective, which also helped to identify the piling up of inventory [23]. Depending on the time dedicated to development, each story was given a story point value discussed and sorted out by the developers of the team. Usually, developers were estimating the story size as small and even, to better structure

their work. Swaminathan and Jain [23] measured story rate per iteration as the total number of story points approved and closed by the customer in the given iteration. The flow of requirements through the software development life-cycles was the key topic also in Petersen and Wohlin [17]. However, this study was measuring hand-overs within the stories as well as the variance, to better predict the development cycle [17].

7) Release frequency: The only study which used the number of releases was Middleton and Joyce [2], defining it as the number of items released to customers. Time-frame for measuring the frequency of releases was set to one month. Even though this metric does not reveal how much value is being delivered to the customers, it showed an eight-fold increase in releases for a two years period [2]. This is indicating an improvement in configuration management discipline and capability [2], as the more frequent releases are reducing technical and market risks, as the customer can evaluate a real product in smaller increments rather than just seeing temporary results from progress reports.

IV. Conclusion

The goal of this paper was to better understand the "agile-to-lean" transformation process regarding benefits, challenges, and metrics that primary studies reported in transformations within companies. To reach the goal, we conducted a Systematic Literature Review (SLR) [6]. The most represented benefits were reduced lead time, improved flow, continuous improvement and improved defect fix rate. Seeking the challenges faced, the common problems were the adaptation to the lean mindset, teaching and maintaining the "lean mindset", maintaining development flexibility, and the identification of the concept of waste. The most used metric to measure a lean transformation was lead time.

References

[1] X. Wang, "The combination of agile and lean in software development: An experience report analysis," in *2011 Agile Conference*, Conference Proceedings. doi: 10.1109/AGILE.2011.36 pp. 1–9.

[2] P. Middleton and D. Joyce, "Lean software management: Bbc worldwide case study," *IEEE Transactions on Engineering Management*, vol. 59, no. 1, pp. 20–32, 2012. doi: 10.1109/TEM.2010.2081675

[3] M. Kalenda, P. Hyna, and B. Rossi, "Scaling agile in large organizations: Practices, challenges, and success factors," *Journal of Software: Evolution and Process*, p. e1954. doi: 10.1002/smr.1954

[4] M. Poppendieck, T. Poppendieck, and T. Poppendieck, *Lean Software Development: An Agile Toolkit*, ser. The agile software development series. Addison-Wesley, 2003. ISBN 9780321150783

[5] A. Shalloway, G. Beaver, and J. R. Trott, *Lean-agile software development: achieving enterprise agility*. Pearson Education, 2009.

[6] B. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering," 2007.

[7] P. Middleton, "Lean software development: two case studies," *Software Quality Journal*, vol. 9, no. 4, pp. 241–252, 2001. doi: https://doi.org/10.1023/A:1013754402981

[8] K. B. Stone, "Four decades of lean: a systematic literature review," *International Journal of Lean Six Sigma*, vol. 3, no. 2, pp. 112–132, 2012. doi: https://doi.org/10.1108/20401461211243702

[9] E. Kupiainen, M. V. Mäntylä, and J. Itkonen, "Using metrics in agile and lean software development–a systematic literature review of industrial studies," *Information and Software Technology*, vol. 62, pp. 143–163, 2015. doi: https://doi.org/10.1016/j.infsof.2015.02.005

[10] X. F. Wang, K. Conboy, and O. Cawley, ""leagile" software development: An experience report analysis of the application of lean approaches in agile software development," *Journal of Systems and Software*, vol. 85, no. 6, pp. 1287–1299, 2012. doi: 10.1016/j.jss.2012.01.061

[11] T. Hayata, J. Han, and M. Beheshti, "Facilitating agile software development with lean architecture in the dci paradigm," in *2012 Ninth International Conference on Information Technology - New Generations*. doi: 10.1109/ITNG.2012.157 pp. 343–348.

[12] C. R. Jakobsen and T. Poppendieck, "Lean as a scrum troubleshooter," in *2011 Agile Conference*, Conference Proceedings. doi: 10.1109/AGILE.2011.11 pp. 168–174.

[13] R. Kuusela and M. Koivuluoma, "Lean transformation framework for software intensive companies: Responding to challenges created by the cloud," in *2011 37th EUROMICRO Conference on Software Engineering and Advanced Applications*, Conference Proceedings. doi: 10.1109/SEAA.2011.74. ISBN 1089-6503 pp. 378–382.

[14] M. Misaghi and I. Bosnic, "Lean mindset in software engineering: A case study in a software house in brazilian state of santa catarina," vol. 466, pp. 697–707, 2014. doi: https://doi.org/10.1007/978-3-319-11854-3_60

[15] M. Paasivaara, C. Lassenius, V. T. Heikkila, K. Dikert, and C. Engblom, "Integrating global sites into the lean and agile transformation at ericsson," *2013 Ieee 8th Int. Conference on Global Software Engineering (Icgse 2013)*, pp. 134–143, 2013. doi: 10.1109/icgse.2013.25

[16] G. I. U. S. Perera and M. S. D. Fernando, "Enhanced agile software development - hybrid paradigm with lean practice," in *2007 Int. Conference on Industrial and Information Systems*, Conference Proceedings. doi: 10.1109/ICIINFS.2007.4579181. ISBN 2164-7011 pp. 239–244.

[17] K. Petersen and C. Wohlin, "Measuring the flow in lean software development," *Software-Practice and Experience*, vol. 41, no. 9, pp. 975–996, 2011. doi: 10.1002/spe.975

[18] P. Rodríguez, J. Markkula, M. Oivo, and K. Turula, "Survey on agile and lean usage in finnish software industry," in *Proceedings of the ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, ser. ESEM '12. New York, NY, USA: ACM, 2012. doi: 10.1145/2372251.2372275. ISBN 978-1-4503-1056-7 pp. 139–148.

[19] P. Rodríguez, J. Partanen, P. Kuvaja, and M. Oivo, "Combining lean thinking and agile methods for software development: A case study of a finnish provider of wireless embedded systems detailed," in *2014 47th Hawaii International Conference on System Sciences*, Conference Proceedings. doi: 10.1109/HICSS.2014.586. ISBN 1530-1605 pp. 4770–4779.

[20] U. Samanta, V. S. Mani, and Ieee, "Successfully transforming to lean by changing the mindset in a global product development team," pp. 135–139, 2015. doi: 10.1109/icgse.2015.17

[21] J. Schnitter and O. Mackert, "Large-scale agile software development at sap ag," vol. 230, pp. 209–220, 2011. doi: https://doi.org/10.1007/978-3-642-23391-3_15

[22] D. I. K. Sjøberg, A. Johnsen, and J. Solberg, "Quantifying the effect of using kanban versus scrum: A case study," *IEEE Software*, vol. 29, no. 5, pp. 47–53, 2012. doi: 10.1109/MS.2012.110

[23] B. Swaminathan and K. Jain, "Implementing the lean concepts of continuous improvement and flow on an agile software development project: An industrial case study," in *2012 Agile India*, Conf. Proceedings. doi: 10.1109/AgileIndia.2012.12. ISBN 2326-6007 pp. 10–19.

[24] J. Trimble and C. Webster, "From traditional, to lean, to agile development: Finding the optimal software engineering cycle," in *2013 46th Hawaii Int. Conference on System Sciences*, Conference Proceedings. doi: 10.1109/HICSS.2013.237. ISBN 1530-1605 pp. 4826–4833.

[25] K. Vilkki, *When Agile Is Not Enough*, ser. Lecture Notes in Business Information Processing, 2010, vol. 65, pp. 44–47. ISBN 978-3-642-16415-6

[26] U. Viswanath, "Lean transformation: How lean helped to achieve quality, cost and schedule: Case study in a multi location product development team," in *2014 IEEE 9th Int. Conference on Global Software Engineering*, Conf. Proceedings. doi: 10.1109/ICGSE.2014.13 pp. 95–99.

[27] M. Walter, R. Tramontini, R. M. Fontana, S. Reinehr, and A. Malucelli, *From Sprints to Lean Flow: Management Strategies for Agile Improvement*, ser. Lecture Notes in Business Information Processing, 2015, vol. 212, pp. 310–318.

[28] M. Pergher and B. Rossi, "Requirements prioritization in software engineering: a systematic mapping study," in *Empirical Requirements Engineering (EmpiRE), 2013 IEEE Third International Workshop on*. IEEE, 2013. doi: 10.1109/EmpiRE.2013.6615215 pp. 40–44.