ARE AGILE AND SCALED AGILE FRAMEWORKS REALLY ADDRESSING SOFTWARE DEVELOPMENT?

1. Introduction

In the recent years, digital transformation (DT) has emerged as an important phenomenon (Bharadwaj, El Sawy, Omar, Pavlou, Paul, & Venkatraman, 2013). Digital transformation is defined as "the use of new digital technologies (social media, mobile, analytics or embedded devices) to enable major business improvements (such as enhancing customer experience, streamlining operations or creating new business models)" (Fitzgerald, Kruschwitz, Bonnet, & Welch, 2013). Put differently, digital transformation is about adopting disruptive technologies to increase productivity, value creation, and social welfare (Ebert and Duarte 2018). As digital transformation impacts increasing complexity and scale of technological solutions and emphasizes time to market, quality, and affordability (Ebert and Duarte 2018), effective software development methods, techniques, and tools are needed to address these issues of IT systems delivery. Original Agile methods, designed to be used in small, single team projects (Boehm and Turner 2005), have stopped being sufficient. This has resulted in a birth of Scaled Agile methods that are nowadays broadly adopted according to global surveys (CollabNet VersionOne 2019; Dingsøyr and Moe 2014).

Software development encompasses both a project management viewpoint and engineering viewpoint. Agile methods, especially the most popular ones such as Scrum and Kanban, cover mostly the management of software projects and do not address the engineering practices. From this viewpoint, is the case of Scaled Agile methods different? The aim of this paper is to analyze Agile and Scaled Agile methods and answer the research question: To what extent do Agile and Scaled Agile methods cover the engineering activities? To answer this research question, a mapping of individual methods to the ISO/IEC/IEEE 12207:2017 is provided.

The rest of the paper is organized as follows. Following the Introduction, Section 2 describes the background, i.e. a brief introduction to Agile and Scaled Agile methods and international standards for Software Process Improvement. Next, Section 3 describes the research approach. Section 4 then presents the results of the mapping of selected Agile and Scaled Agile methods to the ISO/IEC/IEEE 12207 standard. Finally, concluding remarks are given in Section 5.

2. Background

2.1. Agile and Scaled Agile Methods

While the traditional plan-driven software development methods, known as "Waterfall", do not scale to the challenges brought by digital transformation, the Agile and Lean approaches are a major step in that direction. Agile methods have now become the mainstream for software development worldwide. Formally, they were introduced through a set of four core values and 12 principles laid out in the Agile Manifesto (Beck et al., 2001). The benefits of Agile methods lie in reducing the risk of a project failure as software increments are delivered regularly based on prioritized requirements and frequent user feedback.

In order to use the Agile approach also for larger projects and larger companies handling an inter-team coordination and interfacing with other organizational units, such as human resources,

marketing and sales, and product management is needed (Dikert, Paasivaara, & Lassenius, 2016). Therefore, a number of Scaled Agile methods and frameworks have emerged and are used in the industry such as the Discipline Agile Delivery (DAD), Large-scale Scrum (LeSS), Scaled Agile Framework (SAFe), Scrum@Scale, and Nexus (Kalenda, Hyna, & Rossi, 2018).

The state of Agile software development methods adoption has been surveyed both by scientists and practitioners since 2006. However, some surveys were focused only on specific geographic territories, e.g. Finland (Dikert et al., 2016) or Brazil (Campanelli, Camilo, & Parreiras, 2018). In addition, researchers have tried to reach English-speaking populations across the globe by offering them survey instruments in English (Kuhrmann et al., 2018; Kurapati, Manyam, & Petersen, 2012). In the world of business practice, the "State of Agile" survey with a global reach has been conducted by VersionOne (later CollabNet VersionOne) annually since 2006. This well-known practitioner survey has also added a part focused on Scaled Agile methods since 2013. The recent (13th) edition (CollabNet VersionOne, 2019) was carried out between August and December 2018. Based on this survey, the most used Agile methods for the analysis were selected, i.e. Scrum with a 54% usage, then the combination of Scrum and Extreme Programming with a 10% usage, quickly increasing Kanban with 5% and the combination of Scrum and Kanban with 8%. The Scaled Agile Framework (SAFe) with a 30% usage, Disciplined Agile Delivery (DAD) which an increasing usage reaching 7% and quite popular Large-scale Scrum (LeSS) framework with 3% were selected among Scaled Agile methods. These methods and frameworks are briefly introduced in next sections.

2.1.1. Scrum

Scrum is a development framework in which cross-functional teams develop products or projects in an iterative, incremental manner. It structures development in cycles of work called Sprints that are timeboxed. At the beginning of each Sprint, a cross-functional Team (of about seven people) selects items (customer requirements) from a prioritized list. The Team agrees on a collective target of what they believe they can deliver by the end of the Sprint. The Team gathers every day briefly to inspect its progress, and adjust the next steps needed to complete the work remaining. At the end of the Sprint, the Team reviews the Sprint with stakeholders, and demonstrates what has been built. This way, teammates obtain feedback that can be incorporated in the next Sprint. Scrum emphasizes a working product at the end of the Sprint that is really "done"; in the case of software, this means a system that is integrated, fully tested, end-user documented, and potentially shippable (Deemer, Benefield, Larman, & Vodde, 2012).

2.1.2. Extreme Programming

Extreme Programming(XP) is a discipline of software development based on values of simplicity, communication, feedback, and courage (Beck, 1999). It works by using simple practices, with enough feedback to enable the team to see where they are and to tune the practices to their unique situation (Lindstrom & Jeffries, 2004). In XP, every contributor to the project is a member of the "Whole Team". The "Customer" is central to the team and works with them on a daily basis. The team produces the software in a series of small, fully integrated releases that pass all the tests that the Customer has defined. Extreme Programmers work together in pairs and as a group, with a simple design and obsessively tested code, improving the design continually to keep it always just right for the current needs. The XP team keeps the system integrated and running all the time. The programmers write all production code in pairs, and all work together all the time. They code in a consistent style so that everyone can understand and improve all the code as needed (Lindstrom & Jeffries, 2004). The core XP practices are the Whole Team, Small Releases, Acceptance Tests, Pair

Programming, Simple Design, Test-Driven Development, Refactoring, Continuous Integration, Team Code Ownership, and Coding Standards. Some of the XP practices such as testing, refactoring and continuous integration are heavily used within the Software Engineering discipline. However, the XP method stresses a synergy among its practices, which complete each other (Abrahamsson, Warsta, Siponen, & Ronkainen, 2003).

2.1.3. Kanban

Kanban is the most popular among Lean techniques which have their roots in Lean manufacturing and Toyota Production System. In 2004, Kanban was then firstly used for software development (Anderson, David, 2010). Kanban has five core principles: visualize workflow, limit work in progress, measure and manage flow, make process policies explicit, and use models to recognize improvement and opportunities (Anderson, David, 2010). The motivation to use Kanban was to visualize work, limit work in progress (WIP), and identify process constraints to achieve the flow and focus on a single item at a given time (Ahmad, Dennehy, Conboy, & Oivo, 2018). Various studies (Ahmad et al., 2018; Ahmad, Markkula, & Oivo, 2013; dos Santos, Beltrão, de Souza, & Travassos, 2018) have reported the benefits of using Kanban in software development, for example, a better visibility and understanding of the entire process, improved transparency of work and communication, better control of flow and WIP, improved team communication and coordination with other stakeholders, and increased customer satisfaction.

2.1.4. Scaled Agile Framework

The Scaled Agile Framework (SAFe) is a freely revealed knowledge base of proven, integrated patterns for enterprise-scale Lean-Agile development (Scaled Agile, 2018). The SAFe was created by Dean Leffingwell in 2012 and since then it has continually evolved to a current 5.0 version. The SAFe website (Scaled Agile, 2019) provides a guidance for scaling agile development across the Portfolio, Value Stream, Program, and Team levels that are part of the Big Picture, i.e. a visual overview of the Framework. The SAFe's practices are grounded on nine fundamental principles that have evolved from the Agile principles and methods, Lean product development, systems thinking, and observation of successful enterprises. The heart of the SAFe is the Program level, which revolves around an organization called the Agile Release Train (ART). Each ART aligns teams to a common mission and vision via a single program backlog and produces valuable and evaluable system-level solutions every two weeks (Scaled Agile, 2018).

2.1.5. Disciplined Agile Delivery

The Disciplined Agile Delivery (DAD) framework is a hybrid of existing methods such as Scrum, Kanban, Agile Modelling, SAFe, Extreme Programming, Agile Data, Unified Process and many others. DAD provides the flexibility to use various approaches and plugs the gaps not addressed by mainstream agile methods (Ambler & Lines, 2016). The main characteristics of this framework are that it: is a people first, learning oriented hybrid Agile/Lean approach; has a risk value delivery lifecycle; is goal-driven; is enterprise aware; is tactically scalable at the team level; and strategically scalable across all of the enterprise (PMI, 2019).

2.1.6. Large-scale Scrum

The Large-scale Scrum (LeSS) framework was created by Bas Vodde and Craig Larman in 2013 based on their experiences working with large-scale product development. As both authors state in (Larman & Vodde, 2016), scaling Scrum starts with understanding and being able to adopt standard

one-team Scrum. Large-scale Scrum requires examining the purpose of single-team Scrum elements and figuring out how to reach the same purpose while staying within the constraints of the standard Scrum rules. LeSS provides two different large-scale Scrum frameworks, i.e. the basic LeSS applicable up to eight teams (of eight people each) and the LeSS Huge that introduces additional scaling elements for development up to hundreds of developers.

2.2. International Standards for Software Process Improvement

The traditional way towards more successful software projects is represented by a Software Process Improvement (SPI) initiative. Software Process Improvement aims to improve software processes (Humphrey, Watts, 1989) and comprises a variety of tasks, such as scoping, assessment, design and realization, and continuous improvement, etc. In this field, a number of SPI models have been developed like the Capability Maturity Model Integration (CMMI) (CMMI Institute, 2019) or international standards ISO/IEC/IEEE 12207 (ISO/IEC/IEEE, 2017) or ISO/IEC 15504 series that has been converting into the revised content associated with the ISO/IEC 33000 series. For small companies is then intended the ISO/IEC 29110 series (SC7, 2016). In 2017, a new version of the ISO/IEC/IEEE 12207 standard was published (ISO/IEC/IEEE, 2017) which establishes a common framework for software life cycle processes. It contains processes, activities, and tasks that are to be applied during the acquisition of a software system, product or service and during the supply, development, operation, maintenance and disposal of software products. It serves as the Process Reference Model for software life cycle processes.

3. Research Method

To answer the research question and identify the coverage of the engineering activities among Agile and Scaled Agile methods, the analysis of its extent among selected Agile and Scaled Agile methods is performed. The ISO/IEC/IEEE 12207:2017 processes were used as the basis for the analysis. Selected methods for the analysis, as stated in Section 2.1, were Scrum, Extreme programming (XP)(Beck, 1999), Kanban, Scaled Agile Framework (SAFe), Disciplined Agile Delivery (DAD) (Scaled Agile, 2018) and Large-scale Scrum (LeSS). The analysis was conducted in the form of the mapping of individual methods, factually their specific practices, onto the ISO/IEC/IEEE 12207:2017 processes. The level of coverage was expressed by the scale of (i) not covered; (ii) partially covered; (iii) largely covered; (iv) fully covered. The adherence to the ISO/IEC/IEEE 12207:2017 processes was assessed based on the process outcomes and activities and tasks. The results of the analysis are presented in the form of graphical figures depicting all 12207 processes and indicating their level of adherence by color.

4. Mapping of Selected Methods to ISO/IEC/IEEE 12207 Standard

Figure 1 depicts the mapping of the Scrum method to the ISO/IEC/IEEE 12207:2017 processes. It is apparent that only technical management processes are covered by Scrum. Among technical processes, only Requirements definition processes are covered. As Scrum is oriented only on one project, no processes at the organizational level are covered. These results support the well-known classification of the Scrum method being a project management framework. In this sense, Scrum definitely does not lead software development sufficiently and other engineering methods and technics are needed to be used with the Scrum method.

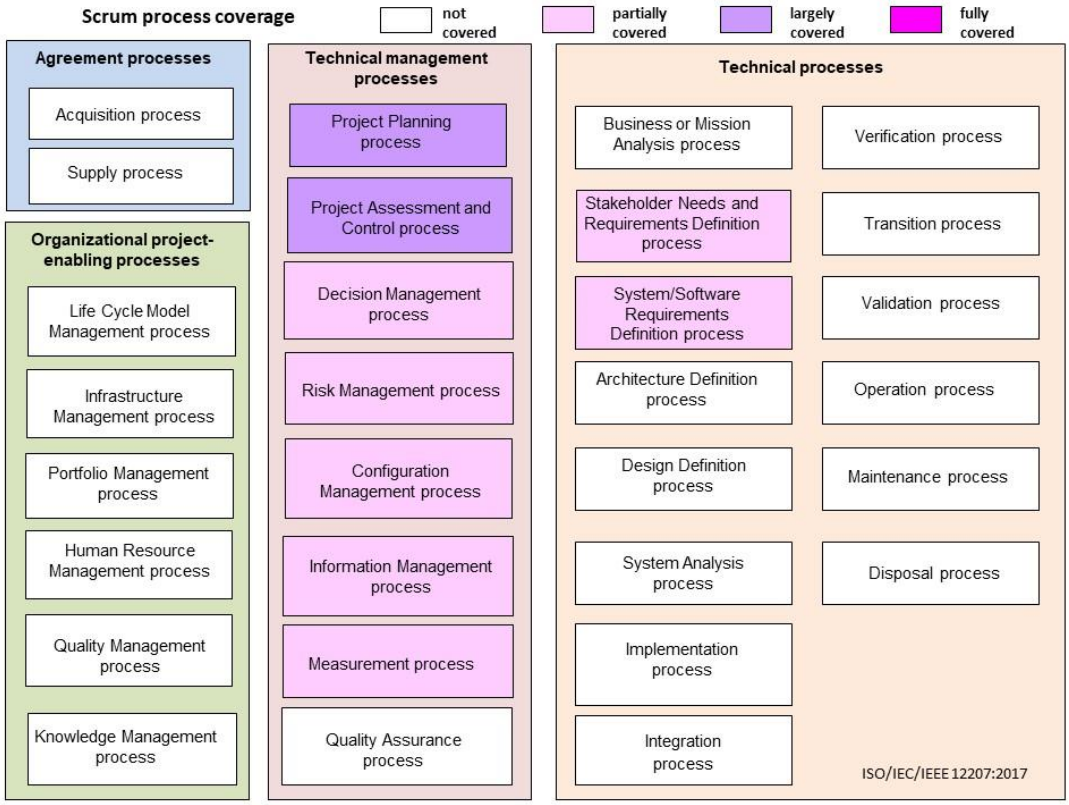

Figure 1. Mapping of the Scrum method to the ISO/IEC/IEEE 12207:2017

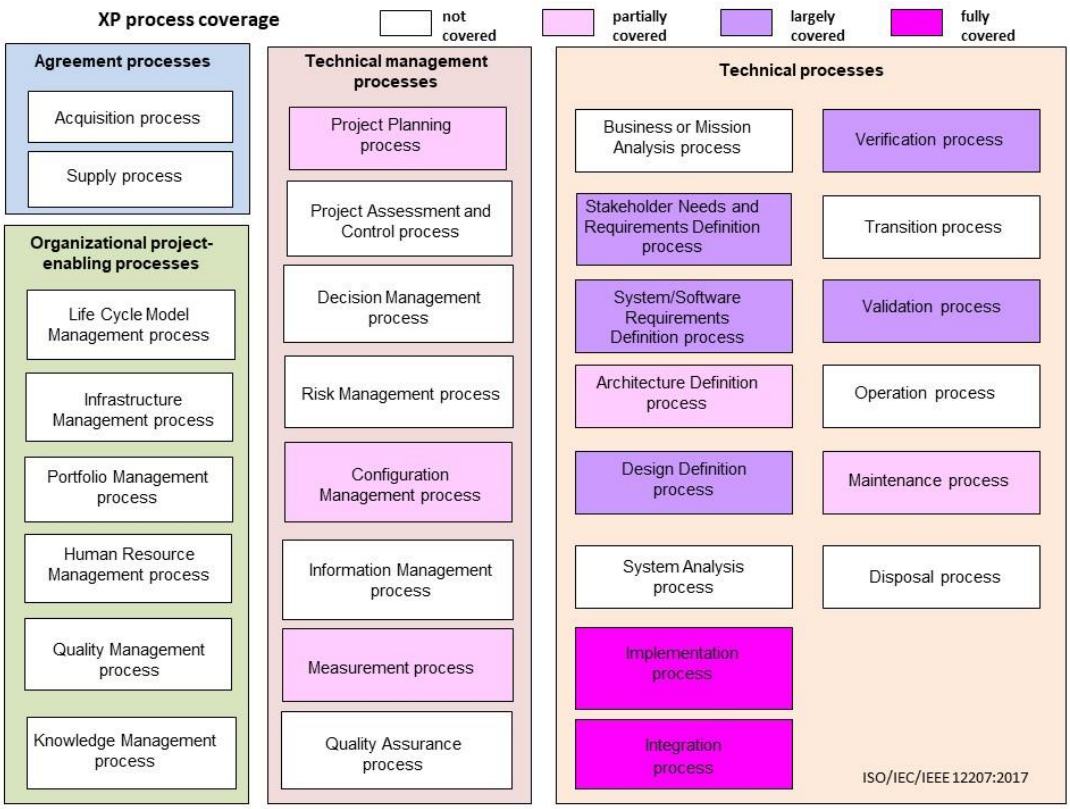

Figure 2. Mapping of the XP method to the ISO/IEC/IEEE 12207:2017

Figure 2 shows mapping of the Extreme Programming (XP) method to the ISO/IEC/IEEE 12207:2017 processes The Extreme Programming has been selected for the analysis as it is one of a few Agile methods that is specifically focused on software engineering. As seen from Figure 2, the results of the mapping confirm this statement. XP is a very disciplined approach based on the engineering practices such as Pair Programming, Simple Design, Test-Driven Development, Refactoring, Continuous Integration, Acceptance Tests and Coding Standards. On the other hand, XP does not cover the management processes, with the exception of a partial coverage of the Project Planning, Configuration Management and Measurement processes. This is why XP is combined with Scrum from its very beginning and this combined method still keeps a relatively high share (10%) among methods being used for software development (CollabNet VersionOne, 2019)

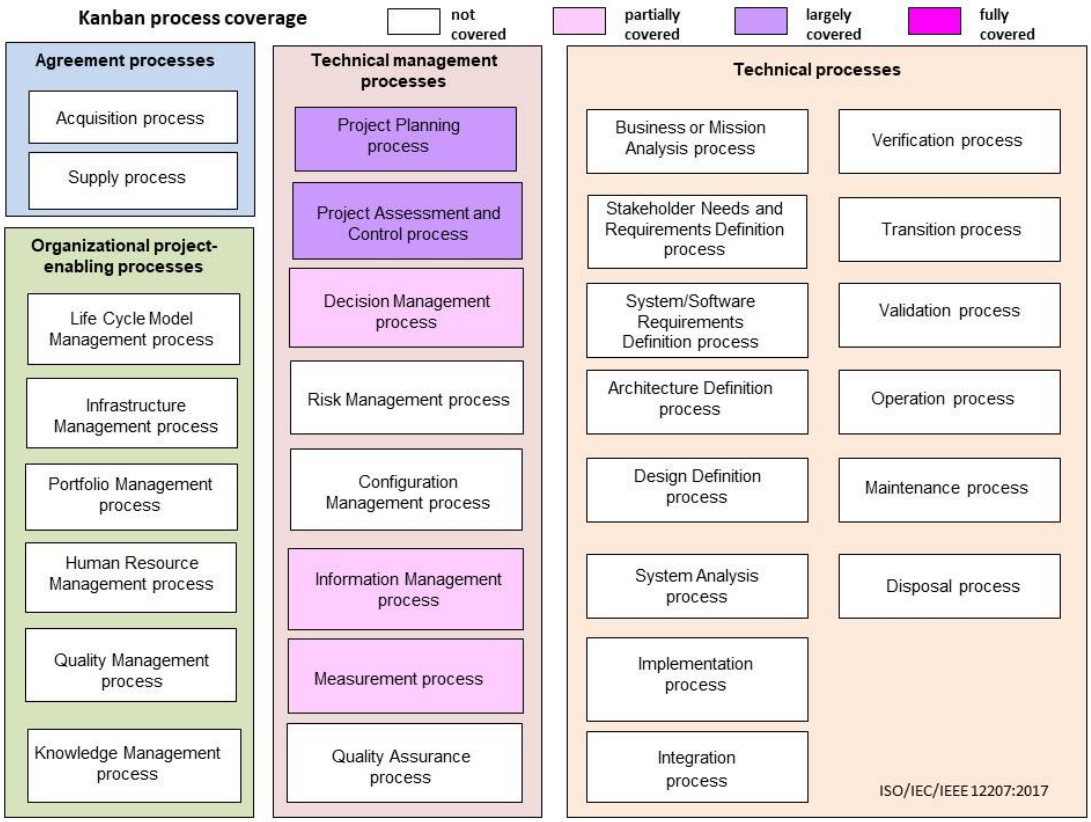

Figure 3. Mapping of the Kanban method to the ISO/IEC/IEEE 12207:2017

Recently, the Kanban method has been gaining an increasing popularity. It is used either separately (5%) or as a combination with Scrum which is named Scrumban and is used by 8% of the latest State of Agile survey respondents (CollabNet VersionOne, 2019). Figure 3 depicts the Kanban´s mapping to the ISO/IEC/IEEE 12207:2017 processes. Similar as Scrum, Kanban is focused especially on project management and is even more lightweight compared to Scrum. It defines only 5 principles that lead the process. No engineering practices are defined within Kanban, not even for requirements definition. The quite popular combination of Scrum and Kanban, i.e. Scrumban, however does not improve the coverage extent of the engineering practices. Also, in the case of Kanban and Scrumban, other engineering practices have to be incorporated to support software development.

Figures 4 to 6 show the mapping of selected Scaled Agile methods to the ISO/IEC/IEEE 12207:2017 processes.

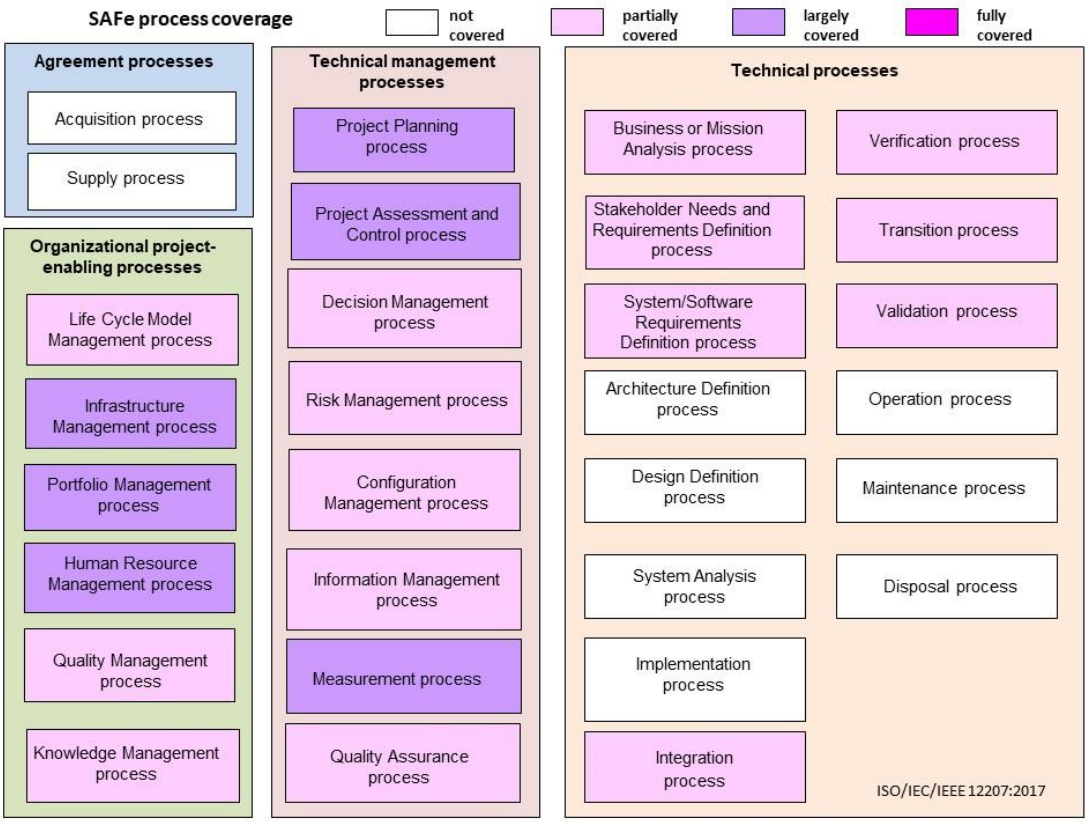

Figure 4. Mapping of the SAFe method to the ISO/IEC/IEEE 12207:2017

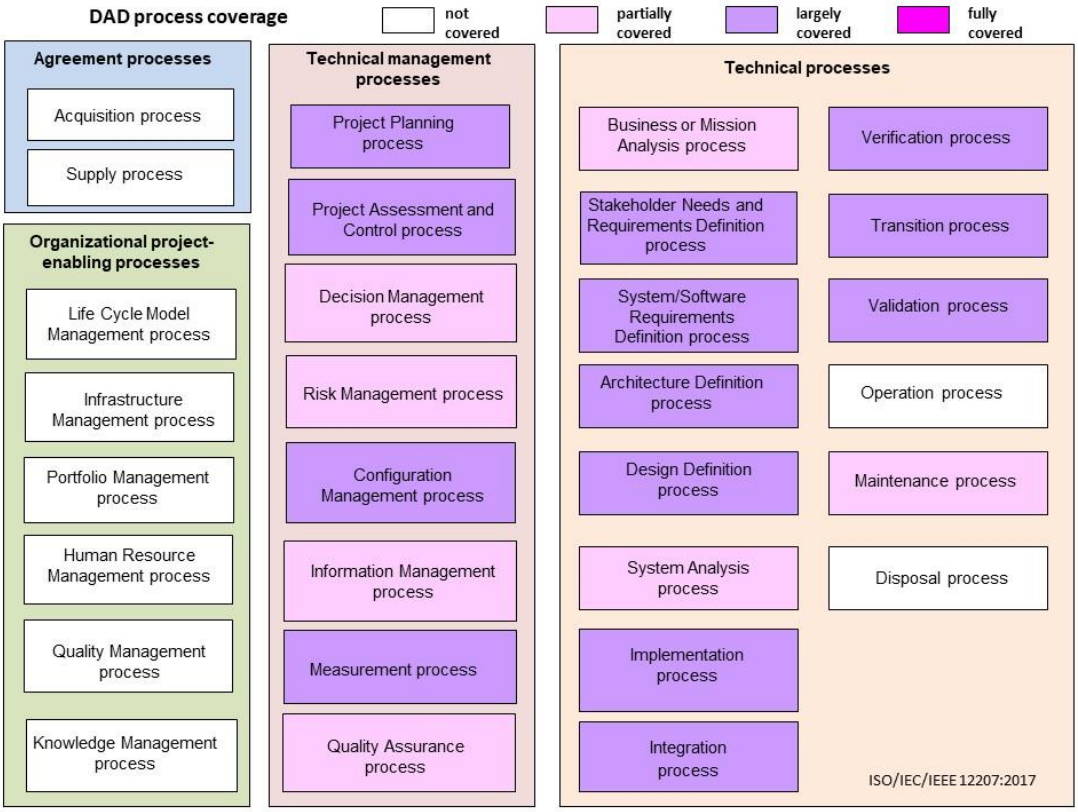

Figure 5. Mapping of the DAD method to the ISO/IEC/IEEE 12207:2017

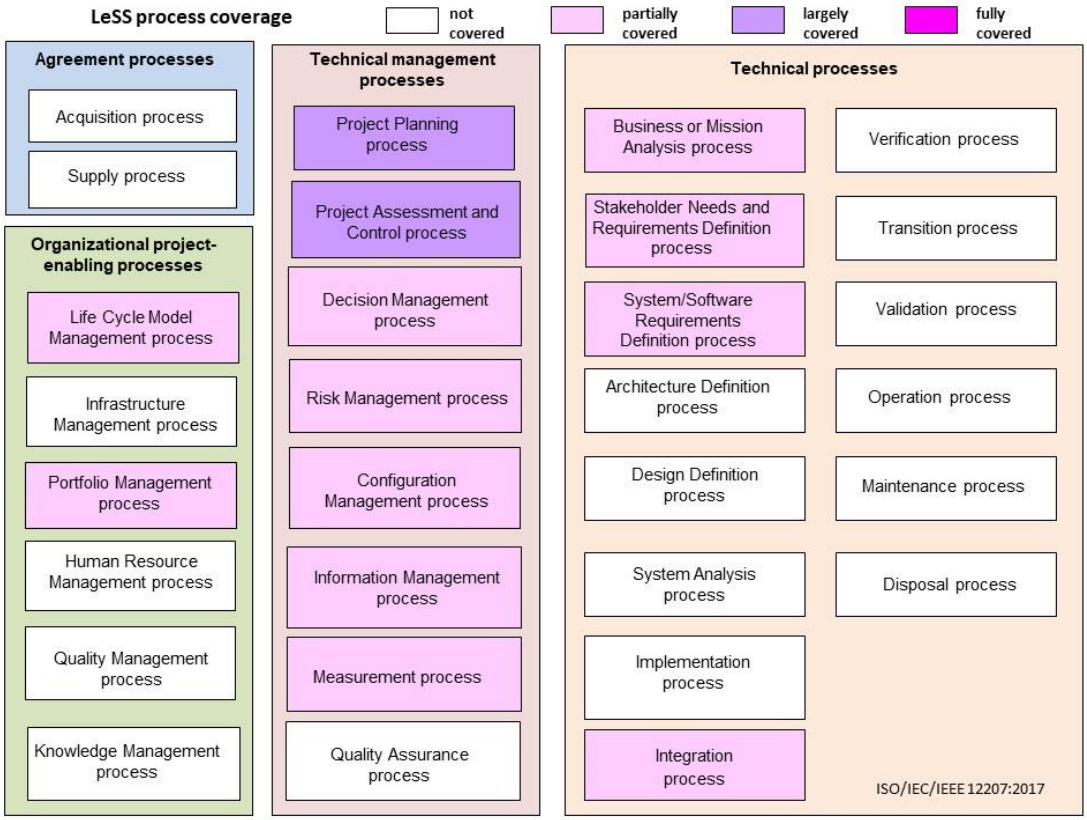

Figure 6. Mapping of the LeSS method to the ISO/IEC/IEEE 12207:2017

The mapping of the Scaled Agile Framework (SAFe), which is the mostly used Scaled Agile method nowadays, is depicted in Figure 4. Compared to other methods, much more 12207 processes are covered by the SAFe. All the technical management processes are covered largely or partially. Worth noticing is the coverage of the Organizational project enabling processes. The SAFe as an example of Scaled Agile methods is focused on the enterprise level and covers largely the Infrastructure Management, Portfolio Management and Human Resource Management processes. Other Organizational project enabling processes are covered partially. However, the SAFe does not cover some engineering processes like the Architecture Definition, Design Definition, System Analysis and Implementation processes.

The Disciplined Agile Delivery (DAD) framework is the only method/framework which covers both technical management processes and technical processes as apparent from Figure 5. In this case, the processes are covered mostly largely. This high level of coverage is due to an effective combination of various proven software development methods like Rational Unified Process, Extreme Programming, Agile modelling, Scrum, Lean and others. This broad coverage of most software development processes may have been the cause of a growing popularity of this framework recently. Disciplined Agile Delivery is a part of the whole Disciplined Agile toolkit that encompasses four levels: Disciplined Agile Delivery, Disciplined DevOps, Disciplined Agile IT and Disciplined Agile Enterprise. Even though DAD does not cover the Organizational project-enabling processes, other parts of the Disciplined Agile toolkit do.

Figure 6 shows process coverage of the Large-scale Scrum (LeSS) framework. On the contrary to SAFe, LeSS covers much less Organizational project enabling processes. Technical management processes are covered mostly, but only partially. Ouality assurance process is not covered. Similar to SAFe key technical processes are not covered in LeSS, except for requirements management and integration processes.

5. Conclusion

In this paper, the analysis of selected Agile and Scaled Agile methods was presented determining to what extent individual methods cover software development processes. The most used Agile and Scaled Agile methods were selected, i.e. Scrum, Extreme Programming, Kanban, Scaled Agile Framework (SAFe), Disciplined Agile Delivery (DAD) and Large-scale Scrum (LeSS). The analysis was performed through the mapping of individual methods to the processes of the ISO/IEC/IEEE 12207:2017 international standard.

The results of the mapping show that Scrum covers only the technical management processes and thus other engineering methods and technics have to be used together with Scrum to support software development completely. On the opposite side lies Extreme Programming which is specifically focused on software engineering and does not cover project management sufficiently. That is why XP is usually combined with Scrum and this combined method still keeps a relatively high share in method usage. Similar to Scrum, Kanban and Scrumban are mainly focused on the project management practices.

The Disciplined Agile Delivery (DAD) framework is the only method which covers both technical management processes and technical processes. It is a real hybrid framework that effectively combines several methods like Scrum, Kanban, Agile Modelling, Extreme Programming, Unified Process and others.

The SAFe and LeSS are popular Scaled Agile methods and as such they cover the Organizational project enabling processes, i.e. the processes at an enterprise level. However, they do not cover key engineering processes like the Architecture Definition, Design Definition, System Analysis and Implementation processes.

Thus, to answer the research question, we can summarize that the most popular and most used Agile and Scaled Agile methods such as Scrum and SAFe do not cover software engineering processes sufficiently and have to be supplemented by other engineering practices.

6. References

Abrahamsson, P., Warsta, J., Siponen, M. T., & Ronkainen, J. (2003). New directions on agile methods: A comparative analysis. *Proceedings - International Conference on Software Engineering*, *6*, 244–254. https://doi.org/10.1109/icse.2003.1201204

Ahmad, M. O., Dennehy, D., Conboy, K., & Oivo, M. (2018). Kanban in software engineering: A systematic mapping study. *Journal of Systems and Software*, *137*, 96–113. https://doi.org/10.1016/j.jss.2017.11.045

Ahmad, M. O., Lwakatare, L. E., Kuvaja, P., Oivo, M., & Markkula, J. (2017). An empirical study of portfolio management and Kanban in agile and lean software companies. *Journal of Software: Evolution and Process*, *29*(6). https://doi.org/10.1002/smr.1834

Ahmad, M. O., Markkula, J., & Oivo, M. (2013). Kanban in software development: A systematic literature review. *Proceedings - 39th Euromicro Conference Series on Software Engineering and Advanced Applications, SEAA 2013*, 9–16. https://doi.org/10.1109/SEAA.2013.28

Ambler, S. W., & Lines, M. (2016). *The Disciplined Agile Process Decision Framework*. https://doi.org/10.1007/978-3-319-27033-3_1

Anderson, David, J. (2010). *Kanban: Successful Evolutionary Change for Your Technology Business*. Blue Hole Press.

Beck, K. (1999). *Extreme Programming Explained : Embrace Change*. Addison-Wesley Professional.

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., … Thomas, D. (2001). Manifesto for Agile Software Development. Retrieved March 10, 2019, from https://agilemanifesto.org/

Bharadwaj, A., El Sawy, Omar, A., Pavlou, Paul, A., & Venkatraman, N. (2013). Digital Business Strategy: Toward a Next Generation of Insights. *MIS Quarterly*, *37*(20), 471–482.

Campanelli, A. S., Camilo, R. D., & Parreiras, F. S. (2018). The impact of tailoring criteria on agile practices adoption: A survey with novice agile practitioners in Brazil. *Journal of Systems and Software*, *137*. https://doi.org/10.1016/j.jss.2017.12.012

CMMI Institute. (2019). CMMI. Retrieved August 10, 2019, from https://cmmiinstitute.com/

CollabNet VersionOne. (2019). The 13th annual STATE OF AGILE Report - 2018. In *CollabNet | VersionOne* (Vol. 13).

Deemer, P., Benefield, G., Larman, C., & Vodde, B. (2012). *A Lightweight Guide to the Theory and Practice of Scrum*.

Dikert, K., Paasivaara, M., & Lassenius, C. (2016). Challenges and success factors for large-scale agile transformations : A systematic literature review. *The Journal of Systems & Software*, *119*, 87–108.

dos Santos, P. S. M., Beltrão, A. C., de Souza, B. P., & Travassos, G. H. (2018). On the benefits and challenges of using kanban in software engineering: a structured synthesis study. *Journal of Software Engineering Research and Development*, *6*(1), 1–30. https://doi.org/10.1186/s40411-018-0057-1

Fitzgerald, M., Kruschwitz, N., Bonnet, D., & Welch, M. (2013). Embracing Digital Technology: A New Strategic Imperative | Capgemini Consulting Worldwide. In *MIT Sloan Management Review* (Vol. 55). Retrieved from https://www.capgemini-consulting.com/SMR

Humphrey, Watts, S. (1989). *Managing the software process*. Boston: Addison-Wesley.

ISO/IEC/IEEE. (2017). *ISO/IEC 12207:2017 Systems and Software Engineering -- Software Life Cycle Processes*.

Kalenda, M., Hyna, P., & Rossi, B. (2018). Scaling agile in large organizations: Practices, challenges, and success factors. *Journal of Software: Evolution and Process*, *30*(10), e1954.

Kuhrmann, M., Diebold, P., Munch, J., Tell, P., Trektere, K., Mc Caffery, F., … Prause, C. (2018). Hybrid Software Development Approaches in Practice: A European Perspective. *IEEE Software*. https://doi.org/10.1109/MS.2018.110161245

Kurapati, N., Manyam, V. S. C., & Petersen, K. (2012). Agile software development practice adoption survey. *International Conference on Agile Software Development*. Springer, Berlin, Heidelberg.

Larman, C., & Vodde, B. (2016). *Large-Scale Scrum: More with LeSS*. Addison-Wesley Professional.

Lindstrom, L., & Jeffries, R. (2004). Extreme programming and agile software development methodologies. *Information Systems Management*, *21*(3), 41–52. https://doi.org/10.1201/1078/44432.21.3.20040601/82476.7

PMI. (2019). Disciplined Agile. Retrieved March 10, 2019, from https://www.disciplinedagiledelivery.com/

SC7. (2016). *ISO/IEC TR 29110-1 Systems and software engineering -- Lifecycle profiles for Very Small Entities (VSEs) -- Part 1: Overview*.

Scaled Agile. (2018). *SAFe 4.6 Introduction*. https://doi.org/10.1017/S1431927615014221

Scaled Agile. (2019). Scaled Agile Framework. Retrieved from https://www.scaledagileframework.com/