Becoming Agile While Preserving Software Product Lines

An Agile Transformation Model For Large Companies

Jil Klünder
Leibniz Universität Hannover
Software Engineering Group
Germany
jil.kluender@inf.uni-hannover.de

Philipp Hohl
Daimler AG
Research and Development
Germany
philipp.hohl@daimler.com

Kurt Schneider
Leibniz Universität Hannover
Software Engineering Group
Germany
kurt.schneider@inf.uni-hannover.de

ABSTRACT

Software process improvement has always been an essential part of software projects. Current market trends and the rapid pace of changing requirements demand fast development and adaptability. Agile software development is a popular possibility to react on these trends. Implementing agile practices promises for example a shorter time-to-market, satisfied customers and increased software quality. Consequently, many companies strive for an integration of agile methods or for an agile transformation.

High-technological environments such as the automotive domain also want to benefit from the advantages promised by agile software development. Even more than smaller companies, these large ones have to deal with software systems getting more and more complex. One established solution facing this problem is an effective and managed way to reuse software at least partially. Software product lines provide an efficient way to manage software reuse and to handle the high complexity. Consequently, they are widely distributed in large and high-technological environments.

In most companies in the automotive domain, software product lines are already present and agile development methods should be introduced. Hence, there is a need for a transformation model preserving the benefits of software product lines. We conducted a literature review to achieve an overview and a better understanding of agile transformation models in large companies. In total, we analyzed 367 papers. None of them addresses the agile transformation in large companies with existing software product lines. In consideration of this research gap, we propose a transformation model preserving the benefits of already existing models and extending aspects which are important for existing software product lines.

CCS CONCEPTS

• **Software and its engineering → Software product lines**; *Embedded software*; **Agile software development**; *Collaboration in software development*;

KEYWORDS

Agile transformation, software product line engineering, transformation process

ICSSP '18, May 26–27, 2018, Gothenburg, Sweden
© 2018 Association for Computing Machinery.
This is the author's version of the work. It is posted here for your personal use. Not for redistribution. The definitive Version of Record was published in *ICSSP '18: International Conference on the Software and Systems Process 2018 (ICSSP '18), May 26–27, 2018, Gothenburg, Sweden*, https://doi.org/10.1145/3202710.3203146.

ACM Reference Format:
Jil Klünder, Philipp Hohl, and Kurt Schneider. 2018. Becoming Agile While Preserving Software Product Lines: An Agile Transformation Model For Large Companies. In *ICSSP '18: International Conference on the Software and Systems Process 2018 (ICSSP '18), May 26–27, 2018, Gothenburg, Sweden.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3202710.3203146

1 INTRODUCTION

Software development processes are recently in a disruptive change. The increasing rapid pace of changing requirements to satisfy market needs as well as to be within the framework of existing and changing norms and standards lead to an increasing need for rapid development and adaptability [41]. Hence, continuous software process improvement is a current topic many companies have to deal with [30].

There are various existing approaches of improving processes [30]. Nonetheless, the development processes which are nowadays in use are seen as too slow to keep pace in future [8]. Agile Software Development (ASD) addresses this problem. It promises a shorter time-to-market, faster feedback and an increased software quality [5, 14, 31].

Consequently, ASD is a great topic in the area of software process improvement. In a systematic mapping study, Kuhrmann et al. [30] analyzed papers facing the topic of software process improvement, thereof 73 papers dealing with the suitability of agile and lean approaches to improve the development process.

Due to the expected benefits, many companies want to change their development process towards an agile development process. Consequently, they aim at transforming their existing development process. Performing an agile transformation often goes along with fundamental changes. Introducing agile is not just the introduction of a development method, it is also about changing people by establishing a new mindset [28]. Hence, there is a need for an execution plan for the transformation process. There are many already existing models and approaches providing this plan and helping teams or companies to achieve their aim of "being agile" (c.f. [6, 52, 56]). Furthermore, the transformation is often supported by external coaches who help to transform into an agile organization [13].

High-technological large organizations such as companies in the automotive domain also face the already mentioned problems. In addition, these companies have to deal with another difficulty: The software systems are getting more and more complex which requires an effective and managed way to reuse software at least partially [41]. Software Product Lines (SPLs) provide an efficient way to manage software reuse and to handle the high complexity. Furthermore, Software Product Line Engineering allows the reuse

of program code and test cases as well as the adjustments to specific contexts. Hence, Software Product Line Engineering is widely distributed in the automotive domain.

Combining SPLs and agile development methods is not trivial [25]. Nonetheless, large companies strive for a combination of both methods. Hohl et al. [25] identified that Agile Software Product Line Engineering is driven by an assumed improvement of customer collaboration and the software development. It promises to deliver high-quality software at the required faster pace [25].

New trends require even shorter time-to-market, shorter release cycles as well as the introduction of development practices such as continuous integration. Companies assume that the development could benefit from both a working reuse strategy and an increased flexibility with an agile SPL [25]. This flexibility is necessary to react on customer needs and changing requirements during the development process.

In most cases, SPLs are already used, and the company wants to transform towards agile. Hence, there is a need for a transformation model preserving SPLs. In order to gain an overview of already existing approaches to transform a large company, we conducted a literature review. There are many transformation models facing different contexts such as small or large companies and having different focuses such as human factors or development methods. However, we did not find any transformation model preserving or supporting Software Product Line Engineering.

In this contribution, we figure out the possibilities and prerequisites for an agile transformation preserving the benefits of Software Product Line Engineering. We base our approach on the benefits and good characteristics of already existing transformation models but improve and extend the aspects which need to be adapted in order to preserve SPLs. The results are summarized in our transformation model providing a procedure for a successful transformation.

The contribution of our work is the following:

- Identifying and analyzing already existing transformation models for large companies.
- Defining an agile transformation model for large companies preserving SPLs.

The remainder of the paper is structured as follows: In Sec. 2, we present related work. Section 2.2 presents various approaches to combine Software Product Line Engineering and ASD in practice. The literature review is presented in Sec. 3. The results are combined in a transformation model presented in Sec. 4. We discuss our results in Sec. 5 and conclude our work in Sec. 6.

2 RELATED WORK

Our literature study addresses two issues: ASD in large companies and its combination with Software Product Line Engineering.

2.1 Agile Adoption in Large Settings

Rohunen et al. [51] present approaches to adopt agile in large settings. The authors compare the results of a literature analysis and an industrial inventory. They performed a study to figure out the currently used strategies in the software market to adopt agile methods. They identified 120 studies discussing the topic of agile adoption [51]. The authors selected the studies based on the publication date, the focus of the study and the outcomes. A study was only included if it contained a description of the experimental design including the context, data collection, and analysis, findings and conclusions as well as threats to validity. They included 38 studies in their research material. 13 of these publications were found as being relevant due to their focus on agile adoption. As second part of Rohunen et al.'s study [51], the authors considered the FLEXI project consortium. The consortium consists of 8 large-scale industry partners, 14 SMEs and 11 research or university partners with experiences in ASD. The authors figured out which strategies are used to adopt agile in the FLEXI project. In contrast to Rohunen et al. [51], we focus on agile transformation instead of agile adoption. Nonetheless, we included their results in our literature review.

Dyba and Dingsøyr [14] present a literature review on empirical studies within the context of ASD. They identified 1996 studies in the area of ASD since 2005. 36 of these studies are empirical ones [14]. During the analysis of these papers, the authors clustered them into four topics: (1) introduction and adoption, (2) human and social factors, (3) perceptions on agile methods and (4) comparative studies [14]. In our literature review, we focus on category (1). There are 7 papers in the main category. These 7 papers have also been included in our literature review. However, only two of them affect the transition of agile methods [14].

In contrast to the already existing literature studies facing the topic of ASD, we limit our study on agile transformation in large companies. We aim at finding transformation models which can be applied at a large scale.

We will present more papers facing the topic of ASD in large settings and suitable transformation models in Sec. 3.4.1.

2.2 Combining ASD and SPLs

McGregor [39] identifies that both ASD and Software Product Line Engineering are successful approaches to develop software-intensive products. He raises the question if SPLs can be effectively integrated with agile techniques [39]. Due to the existence of a wide variety of adaption strategies, the question cannot be easily answered [39]. Martini et al. [38] point out that ASD helps to reduce the time to market. However, it is necessary to consider reuse strategies such as Software Product Line Engineering as well. The Software Product Line Engineering is used for long-term productivity, efficiency, and profit [38]. The need and the perceived benefits of combining ASD and SPLs are identified in various publications during the last decades. Babar et al. [3] promote ASD and SPLs as means of reducing time to market, increasing productivity and an improving quality. Ghanam et al. [20], O'Leary et al. [42] and Díaz et al. [11] also point out the enhancement in quality and simultaneous reduction of costs and development efforts.

Several concrete methods and models that combine ASD and product line concepts are available. Each method comprises Software Product Line Engineering with selected agile approaches and techniques. Santos Jr. and Lucena Jr. [54] present the ScrumPL which supports iterative domain and application engineering based on Scrum.

The identified publications underline the need for an agile transformation while preserving existing SPLs. However, none of the publication focus on the transformation itself. The authors only present the perceived benefits after a successful transformation

and give combination models that combine Software Product Line Engineering and ASD.

Tian and Cooper [57] mention two possible approaches of an agile transformation. One approach is to take an existing SPL process and introduce agility, the other approach starts with an agile process and tailors it for SPLs [57]. Tian and Cooper [57] identify the way to end up with a combination of ASD and Software Product Line Engineering. However, they do not give any recommendations on how to reach this state. Weber [61] identifies the need of suitable tools and an appropriate infrastructure as the precondition for a successful agile transformation. In addition, Wiklund et al. [62] identify that impediments during early phases of the agile transformation were related to project management, coordination, and communication.

3 LITERATURE REVIEW

In order to gain an overview of already existing transformation models in large companies, we performed a literature review. The found papers help to identify the necessary steps during an agile transformation. The identified steps will serve as the basis for our model presented in Sec. 4.

3.1 Method

We based our approach on the research process proposed by Kitchenham et al. [27].

Definition of the search string. The search string resulted from combinations of the keywords, we wanted to cover with this study: *agile transformation*, *model* and *large scale*. Since many researchers and practitioners use transformation and transition as synonyms, we also included *agile transition* in the search string. Furthermore, we also considered German publications[1]. Hence, we also included the German string *agiles Transformationsmodell* in the search string. This resulted in the following search string:

("agile transformation" OR "agile transformation model" OR "agile transformation large scale" OR "agile transition" OR "agile transition model" OR "agile transition large scale" OR "agiles Transformationsmodell")

We adapted the search string for the selected databases, since search options differ and are specific for each search engine.

Definition of inclusion and exclusion criteria. We eliminated studies and publications which are not relevant for our research topic. To make this decision more objective, we defined the inclusion and exclusion criteria summarized in Tab. 1.

Selection of databases. We selected five scientific databases for our literature study: Science Direct, IEEE, ACM Digital Libraries, Springer Link, and Google Scholar. Based on this selection, we conducted a comprehensive search to avoid biases [47].

3.2 Execution

An overview of the process and the number of papers can be found in Fig. 1. The search resulted in 367 identified contributions. In the first step of the process, we applied the inclusion and exclusion criteria to the title of the distributions. We found 142 papers fitting the scope of our study according to the title, thereof 47 duplicates. This led to 95 remaining papers.

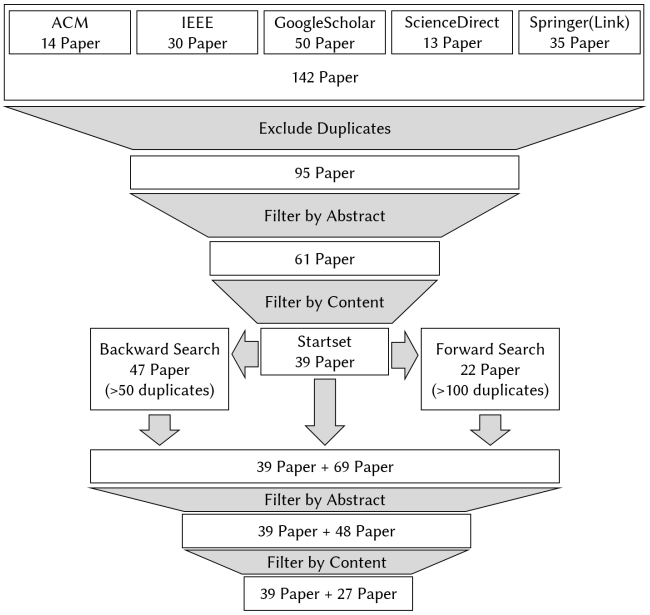

Figure 1: Search process and filtering steps.

In a second step, we filtered the papers with respect to the inclusion and exclusion criteria under consideration of the abstract and the keywords. This led to an exclusion of 34 papers. Considering the content of the papers resulted in a "starting set" of 39 papers.

We further conducted a forward and backward search for each paper in the starting set: As proposed by Wohlin et al. [64], we considered the references of the papers and the papers in which at least one of these 39 papers is cited. This led to 22 new papers using forward search and 47 papers using backward search without duplicates. In total, we found more than 100 duplicates during the forward search and at least 50 duplicates during the backward search. After this process, we had a set of 39 + 69 papers which might be interesting for our research question. Since we have already considered the 39 papers during the first steps, we applied the inclusion and exclusion criteria only to the new 69 papers. 21 papers were excluded after having screened the title and the abstracts in detail. Considering the content of the paper led to an exclusion of again 21 papers. The remaining 39 + 27 papers were considered as a basis for the transformation model.

3.3 Threats to Validity

The outcome of our literature study is biased by different factors. We will present the threats to validity according to the different steps of the literature review. Furthermore, we will categorize them according to Wohlin et al. [65] as *internal, external, construct* and *conclusion validity*.

[1]None of the found publications written in German has been relevant for the outcome of this paper.

Table 1: Inclusion and Exclusion Criteria applied to the study

	Criteria	Description
Inclusion	IC_1	The paper presents an agile transformation model in large scale companies.
	IC_2	The paper presents an overview of already existing transformation models or frameworks.
	IC_3	The paper is an experience report or a "success story" describing a successful transformation in a large company.
	IC_4	The paper is a peer-reviewed contribution to a conference or a journal or it is a master- or PhD-thesis.
Exclusion	EC_1	The paper has no accordance with at least two of the search keywords.
	EC_2	The paper describes in particular a method or a model for small companies.
	EC_3	The paper is not accessible.
	EC_4	The Paper occurred multiple times.
	EC_5	The paper is neither written in English nor in German.

Definition of the search string: We tried to consider synonyms and all related keywords in order to find the relevant publications. For example, we used "transformation" and "transition" because these two words are often used as synonyms describing the process from a plan-driven working company towards an agile working one (*construct validity*). Nonetheless, there may be other synonyms for the "transformation" or "transition". Some authors may use the words "adoption" or "introduction" in the same context. In our opinion, the adoption of agile methods assumes having performed an agile transformation and adopting the methods to the required context. Nonetheless, a different or extended search string would have led to different results. For our purpose of building a model, the accuracy of our results was sufficient.

Definition of inclusion and exclusion criteria: Specific criteria deciding on the inclusion or exclusion criteria can retain a certain objectivity of the results. In order to reduce biases due to subjective decisions (*internal validity*), we formulated criteria for inclusion and exclusion of a paper. Some of the criteria such as the accessibility of the paper are very objective, while others considering the content of the paper are still subjective.

Selection of databases: There are many databases publishing papers related to the area of Software Engineering. Some publications are listed in more than one database while others are not (*construct validity*). We mitigated the threat of missing publications by using five databases which commonly publish related work.

Execution: The study was mainly executed by one researcher which increases the amount of researcher bias *(internal validity)*. The decision on the inclusion or exclusion of a paper mainly depended on his opinion and hence was subjective. In case of doubts, both researchers decided about the inclusion or exclusion of the paper. The whole process was reviewed by the second researcher.

Results: Due to the construction of the study *(construct validity)*, we cannot guarantee that we have found all papers which are relevant for the topic of agile transformation in a large scale company *(external validity)*. Repeating the snowballing technique might have retrieved more relevant papers.

3.4 Results

To reach our goal of setting up a transformation model for large companies that preserves software product lines, we were mostly interested in gaining an overview of already existing agile transformation approaches in different domains and recommended steps during an information. The results do not necessarily base on approaches directly focusing on the automotive domain.

3.4.1 Agile Transformation Approaches. There are several publications presenting different approaches to transform a traditionally working large company into an agile one. The approaches have some points in common, but differ in many others. The biggest difference is the organizational strategy: bottom-up or top-down. Furthermore, there are mainly two different proceedings: step-by-step (cf. [56]) or *big bang* (cf. [52]).

Most of the found publications present case studies or experience reports (e.g. [16, 33, 37, 40, 44]).

Roman et al. [52] present the agile transformation at a large company with globally distributed projects. The authors report on a qualitative study based on interviews about the reasons of becoming agile and the main concerns of the management [52]. The transformation followed a top-down big-bang transformation which was initiated by a new CIO. After having decided on the agile transformation, he started promoting the idea together with a few other managers. After three and a half months, they announced the idea in the whole world-wide distributed company. This was the kick-off of the agile transformation. After 12 month, all projects had to be "acting agile" [52, p. 34]. So the big-bang transformation took one year and about four months from the decision to the kick-off.

Smits and Rilliet [56] report on the agile transformation at Cisco Voice Technology. The process was structured by Kotter's process "The 8-Step process of successful change" [29] and took more than half a year: It started with planning the change process, continued with formulating the change vision and strategy before realizing and executing the change. Unfortunately, it is not clear who initiated the process. The authors just state that "everybody was aware of the need to move away from the lengthy and rigid waterfall process" [56, p.275]. The authors recommend a step-by-step transformation starting with getting agile skills in the teams before striving for changing the whole development process.

Benefield [6] describes the agile transformation at Yahoo!. They started with a pilot project with four teams implementing Scrum and sharing their experiences afterward. They proposed to complete a comprehensive Scrum training, to work with external Scrum masters for some sprints, to use Schwaber's Scrum practices [55] and to complete at least one sprint. After this sprint, the teams were allowed to abandon Scrum if they are not satisfied with the process. However, they were asked to complete at least one sprint before the

decision. After two months, the feedback was very positive: The teams reported positively about the process and their experiences, and the management also noticed positive results. Other teams started being interested. The process started in February 2005, and at the end of 2005, there have already been 25 teams working with Scrum. The process was "initially championed by people at the top of the company as well as people lower in the ranks, the fact that Scrum was never mandated meant it had genuine bottom-up support" [6, p. 7].

3.4.2 Recommended Transformation Steps. Most of the papers present a transformation process consisting of several intermediate steps. Including these steps is very important for a successful transformation. Following, we describe recommendations steps that should be faced during the transformation process. The given order of the steps neither relies on literature nor indicates a chronological sequence. Nonetheless, there are steps which need to be fulfilled as a prerequisite for certain other steps. We considered these dependencies between the steps. Furthermore, we identified independent steps which need to be faced during an agile transformation but not at a specific point in time.

Organizational steps. Before starting the transformation, several publications recommend having a defined **objective** [2, 10, 19, 50, 53]. The team or organization needs a motivation in order to perform a transformation. Common goals are a shorter time-to-market, increased productivity, more satisfied customers or faster reacts to change requests.

Furthermore, there is a need for a **project plan** indicating a structure and giving an approximate timeline [18, 45, 48, 50, 56, 60]. The project plan must not be very detailed according to the idea of coarse planning in ASD.

In particular in large companies, the available **budget** should be estimated or defined [6, 49]. The management needs to be aware of arising costs. It is possible that the development output during the transformation is slightly less than before. The transformation takes some time and the developers also need to spend time on tasks supporting the transformation. It may also need some time before a team is able to increase its productivity after a transformation. The **risks** also need to be evaluated [4, 7, 32, 49]. An example for a possibly occurring risk is the fulfillment of contracts in spite of the transformation changes.

At a certain point during the transformation, it is necessary to get a **vertical commitment** for the change. The management [6, 10, 12, 22, 35, 58] as well as the involved development teams [2, 49, 58] need to support the transformation. Impediments against the introduction of ASD need to be identified and dissolved.

Development process. Before deciding on the development process, the teams and the management need to gain and share knowledge about ASD as well as about the variety of methods and practices [2, 10, 15, 66]. This knowledge is the base for decisions on the development process and the used tools. Hence, it should be shared with any team member involved in the decision process. The team members should also be aware of the meaning of the agile idea and the mindset. Despite the impossibility to enforce a cultural change, the developers should know about the agile values and principles – and of course about the agile practices. This allows the teams to decide in collaboration with the management on a suitable development process and appropriate development methods. This knowledge can be extended by **trainings**. It is important to have both theoretical and practical knowledge on ASD. Within these trainings, it is possible to internalize agile values and simultaneously apply the practices [2, 6, 10, 13, 17, 19, 23, 35, 49, 63, 66].

ASD is a very broad field and the context-specific selection of suitable agile methods is one of the biggest issues during the transformation process. The teams and the management have to define the **development process** with the underlying management framework such as Scrum or Kanban and the agile artifacts [7, 10, 13, 26]. The collaboration with the customer needs to be considered. Scrum and XP recommend implementing an on-site customer within the process. Often, this is not possible in a worldwide distributed development process. However, it is possible to manage the process with suitable tools to virtually integrate the customer. This can also be done in distributed teams to create the virtual environment of a collocated team. However, the synchronization of the teams in a worldwide distribution has to be considered because of different time zones and cultures. The length of the sprint or iteration and the frequency of increments are very important aspects, too. Afterwards, the development process can be fortified with **development methods** [7, 12, 19, 43, 48, 50, 53, 60, 66]. There are many possibilities and it does strongly depend on the team and the context, which ones are suitable. Practices from XP may be the first choice. Next to Scrum which is only a management framework, XP is the best known agile method, but there are many other ways to implement agile that should be taken into account.

After having decided on the development process and the practices, the technical and other **infrastructure** should be set up [2, 17, 19, 45, 49, 59]. The selection and the use of tools such as JIRA, Confluence or Trello strongly depend on the company's constraints and legal requirements. The workspace of the team needs to be prepared, too. In ASD, a Scrum or XP team mostly works collocated in one room – if possible.

Human Resources and Hierarchical Structures. New development processes may require a customized **distribution of tasks and responsibilities** [7]. According to the idea of an agile self-organized team, there may be one team member who is countable for instance for software quality. But the whole team is responsible for delivering a software with a good quality according to the company's guidelines. Furthermore, new tasks like writing story cards may occur. These tasks have to be done by one team member or another involved person. In the case of writing story cards, this can, for example, be done by the customer.

A new development process often also goes along with new **roles** like a Scrum master or an internal product owner. The management has to decide with the team how to react on this [10, 53].

Transformation. Many publications recommend starting the transformation with a **pilot team** [7, 12, 22, 34, 36]. There might be one team exploring ASD and the transformation. Afterwards, the team members of the pilot team can share the gained knowledge and help other teams during their transformation process.

The sharing of knowledge can be supported by **reviews** [17, 21, 49]. The procedure may be scrutinized during the process in order to adjust it and prevent problems or risks. This continuous review process enables continuous **feedback** [4, 6, 17, 21, 36, 45, 48, 49] which is important to improve the process.

Continuous **learning** as one of the agile ideas should be omnipresent during the whole process [2, 4, 6, 63]. Continuous **improvement** is also necessary in order to enable the transformation. Development teams may not need to face the same problems as the pilot team. If the team is not satisfied with the current status, it can improve it anytime. This is specific for ASD [6, 21, 40, 48, 59]. These are two aspects of the **cultural change**, which must not be enforced. Not only implementing the agile practices, but internalizing the agile mindset is the road to a successful agile company [2, 4, 10, 12, 23, 26, 45, 53, 60, 66].

Lastly, it is possible to expand the process across borders, i.e. in all teams, on all hierarchical levels and departments [13, 19, 26].

4 TRANSFORMATION MODEL

According to the results of our literature review, there does not exist any suitable transformation model preserving SPLs. Hence, there is a need for a transformation model taking SPLs into account. To fill this gap, we propose a transformation model maintaining the benefits of already existing SPL development. Our transformation model is based on the results of our literature review presented in Sec. 3.4.2.

4.1 Benefits of Software Product Lines

Due to the necessity of a transformation model for high-technological organizations such as companies in the automotive domain, we aim at building a transformation model that preserves SPLs which are widespread in this context.

Based on a GQM approach, we identified the benefits of SPLs that should be preserved during the transformation. We defined three goals to analyze the current state of agile SPLs in large companies. Two of the goals aimed at assessing the status quo of Software Product Line Engineering respectively ASD in the company. The third goal considered the improvement of the development process using agile practices while preserving SPLs. The three goals are the following:

- G1: The purpose is to *identify the status* quo of *SPLs* from the point of view of *software developers* in the context of an *ASD*.
- G2: The purpose is to *identify the status quo* of *ASD* from the point of view of *software developers* in the context of *SPLs*.
- G3: The purpose is to *enhance the software development in relation to predefined quality goals* by the use of *agile practices* from the point of view of *software developers* in the context of the *software development process which is using SPLs*.

According to GQM, an abstraction sheet was defined for each goal. The abstraction sheet usually consists of quality aspects, influence factors, dependencies and hypotheses on the influence of the factors on the quality aspects. For our approach of identifying the specific characteristics of Software Product Line Engineering which need to be preserved during the transformation, the quality aspects and influence factors are most important.

For Goal 1, i.e. the status quo of Software Product Line Engineering in a company, we identified the **amount of software reuse (in %)** and the **amount of reuse of testcases (in %)** as quality aspects. These aspects are influenced by the *amount of different variants (total number)*, the *amount of common software (in %)* and the *implementation of SPL management including aspects such as scoping or the use of tools*.

For Goal 2, i.e. the status quo of ASD in a company, we identified the **duration of a development iteration (in weeks/months)** and the **length of development cycles (in weeks)** as quality aspects. These aspects are influenced by the *time between two software integrations (in weeks/months)* and the *amount of changes in the (sprint) backlog during a sprint (total number or in %)*.

For Goal 3, i.e. the improvement of Software Product Line Engineering with ASD, we identified the **amount of software reuse and the corresponding test cases (in %)** as well as **short release cycles** as quality aspects. These are influenced by a *continuous integration and deployment*, an *established use of agile practices* and an *established SPL management*. There are also some dependencies to reach this goal: There is a need for an *adjustable and scalable test strategy* which is suitable for short iterations and for all variants. Furthermore, there is a need for a *virtual integration*.

According to our results of the GQM approach, the most important benefits of SPLs are the **high amount of software reuse** and the **reuse of test cases**. Both aspects and their improvement over time can be objectively measured.

To deliver complex software systems in more and more customer variations, it is necessary to manage reuse by SPLs [9, 41]. SPLs help to achieve an increased software quality, reduced costs and time, and reduced risks [41].

4.2 The transformation model

The identified benefits of SPLs which have been presented in the previous section should be considered while creating the model.

There are mainly three parties involved: the *Management*, the development *Teams* and the *Company*. Since there are some issues affecting the team and the management, we also included one layer for both parties. In the previous section, we have already defined necessary steps of an agile transformation. Each of the steps needs to be performed by at least one of the involved parties. Hence, we decided to assign each of these steps either to the management, the team or the whole organization. As already mentioned, some of these steps depend on others, while others can be performed simultaneously. The arrangement of the steps on the horizontal axis does not imply any temporarily ordering. Instead, there are some dependencies between the steps. For example, the team cannot decide on suitable development methods without fundamental knowledge in ASD.

Whenever a step considers specific SPL tasks, this step is marked with "SPL Extension".

An overview of the model is visualized in Fig. 2.

4.2.1 Management. Transformation models can be driven top-down, i.e. initiated by the management, or bottom-up, i.e. initiated by the developers. In large companies, the bottom-up approach is hard to realize. Hence, the development has to initiate the process. As a necessary part of the initiation process, the management needs to *define the objectives* and to develop a *project plan*. The management is also responsible for the *budget estimation*. The management needs to define the right trade-off on the combination of ASD and SPL with respect to the business value. It further considers that

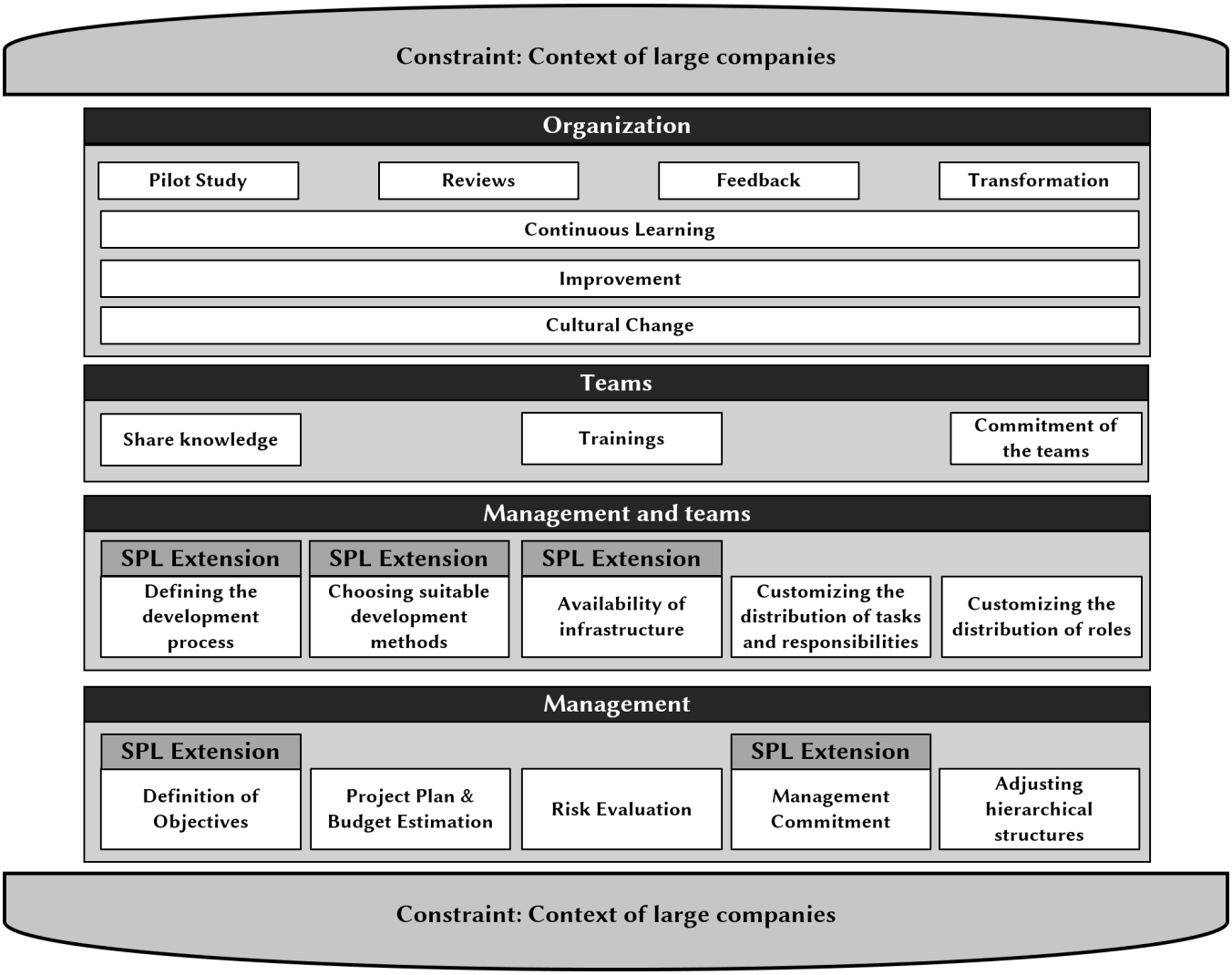

Figure 2: Transformation model based on the literature review. The layers are defined by the responsibilities of different roles in the company (teams, management and organization).

agile development is in budget, creates business value and new terms of supplier payments are introduced.

The *risks* also have to be evaluated by the management. Nonetheless, each involved person is responsible to utter doubts or possible risks. The *adjustment of hierarchical structures*, if necessary, is also a step of the management as well as the *management commitment*.

The management commitment comprises the involvement of the parties which are directly involved in the development process. It is important to develop a common understanding for the SPL and that all affected parties are aware of the development within the SPL. Furthermore, the management commitment helps to overcome impediments given by the organizational hierarchies. The management shall ensure the communication is not blocked by the hierarchy and every participant can speak to every other directly. This will lead to a good and open-minded collaboration within the team and also with suppliers.

4.2.2 Management and Teams. The *development process* and the chosen *development methods* mostly affect the development team. Nonetheless, compulsory organizational structures or legal requirements have to be taken into account and the management needs to be integrated into the process. For maintaining a software reuse strategy, the definition of the development process is important. Large companies often work with SPLs. Software development with SPLs is usually divided into Domain and Application Engineering. These two processes need to be maintained in order to ensure an appropriate reuse strategy. Nonetheless, integrating agile requires a better synchronization of both processes. Managing this synchronization leads to an increased effort to maintain the benefits of SPLs. One possible solution to combine these processes at low effort might be a sprint for Domain Engineering and downstream Application Engineering sprints. Another possibility is the division

of teams into either Domain or Application Engineering with an increased communication.

The development process has to ensure that only one SPL is used for the worldwide development of a product. Furthermore, it is important that the development is synchronized with the surrounding processes and with all affected parties. Synchronizing development tasks and choosing the right development methods must, therefore, support the software reuse. This can be ensured by additional time frames to conduct refactoring of the SPL architecture.

Furthermore, the integration of a development team into the large company affects possibilities and limitations, for example in the choice of suitable tools. These tools shall provide traceability and consistency between software requirements, software units, and the software architecture. Especially maintaining benefits resulting from the software reuse demand for tool support. Hence, both management and teams need to decide on the development process and the chosen methods.

These limitations also affect the *availability of infrastructure*. According to the agile idea of self-organizing and self-deciding teams, they are allowed to decide on the chosen infrastructure. But for example in the automotive domain, there are many conditions which need to be preserved during the transformation such as tools which are requested by the company council.

The *roles, tasks, and responsibilities* need to be adjusted. Since this is both a part of human resources but also of the team, both parties should be involved.

4.2.3 Teams. The idea of self-organizing agile teams is widely distributed. Hence, the team is responsible for gaining *knowledge* required for adequate decisions. The team is also responsible to participate in the agile transformation process. For an appropriate participation, *trainings* facing agile development techniques might be necessary. Furthermore, the team is responsible for the *commitment*. A missing vertical commitment can hinder the agile transformation. Hence, the teams' commitment is mandatory.

4.2.4 Organization. The *pilot study* is primarily performed by one of the teams, but it affects the whole organization. Each team may be the pilot team. So everybody is responsible to find an appropriate team. Furthermore, everybody is able to give *feedback* and to *review* the process. The pilot team may be more responsible than other teams, but nonetheless, everybody is responsible to report on observations, doubts, potential for improvement etc.

In the layer of the organization, the steps *Continuous Learning*, *Improvement* and *Cultural Change* are present throughout the whole process. This indicates that these steps are ongoing and support the transformation from start till the end. The cultural change cannot be forced [24], but will evolve during the transformation process. An improvement is based on the willingness of all participants to learn continuously and accept changes.

The final step, the *transformation* across teams and borders, also affects the whole organization.

4.3 Use of the Model

The presented model can be used as a guideline for large companies throughout an agile transformation process. Some steps depend on each other while others do not. For example, the commitment of the teams and the management is a prerequisite for starting the transformation.

After having identified the interdependencies between the steps, it is possible to define packages of tasks which can be performed simultaneously. For example, the team members can gain and share knowledge about ASD whereas the management develops a project plan and estimates the budget. In the best case, each person group, i.e. management, teams, and organization, is involved at each time.

The selection and assigning of the priority for each task must be considered regarding the context. Furthermore, the time process to conduct each task may vary from organization to organization. However, it is important to focus on all layers and to address not only one part of the organization.

4.4 Rebuilding and Adapting the Model

The separation of the steps into different layers given by the involved groups and the possible parallelism of some steps reminds of a hamburger which is eaten bite per bite. And each bite contains elements of the different layers.

Generating a model based on the idea of a "hamburger" is not new in the area of Software Engineering. Adzic and Evans [1] presented the idea of a hamburger to improve user stories. It provides a proceeding to build the model, to refine and adapt it. Hence, it provides a possibility of using our model. The technique is inspired by Patton and Economy's [46] work on user story mapping and resulted from the fact that their story maps have often been misapplied. Therefore, problems such as too big user stories or missing starting stories arise. The hamburger method deals with these problems by presenting a technique for splitting user stories in an appropriate way.

The hamburger method comprises six steps to create "vertical slices" to identify smaller deliverables.[2] It helps to generate the model in Fig. 2 by identifying and defining the different slices (in our case "management", "team" and "organization") and the tasks contained in each of the slices.

Step 1: Identify tasks. The first step helps to identify the necessary technical steps that need to be done for implementing a story on a high level. A splitting of big stories is done in this step.

For building the transformation model, this step aims at identifying the involved persons and necessary steps during a transformation. The results from the literature review presented in Sec. 3.4.2 can be seen as the results of this step towards a model. We identified important parts of a transformation which need to be covered by our model. Large transformation tasks are clustered and roughly classified.

Step 2: Identify options for tasks. In this step, the quality of each task is defined by smaller teams. Therefore, each team documents several possible implementations and analyses the possible effects on the overall quality.

In the transformation model, this step aims at dividing the large steps into smaller ones. The steps are evaluated by the teams to avoid unnecessary risks that could influence the transformation negatively.

Step 3: Combine results. This step helps to combine the previously done work of the smaller groups. Therefore, representatives

[2] https://gojko.net/2012/01/23/splitting-user-stories-the-hamburger-method/

from each team briefly introduce their opinion about the effects on the quality. This step further helps to identify duplicates and to align tasks according to the level of quality.

In the transformation model, this step aims at identifying duplicates to avoid redundant transformation tasks. This step further requires a collaboration over team levels in the organization and a management commitment to perform the transformation.

Step 4: Trim the hamburger. In Step 4, the tasks are evaluated and the identified quality is related to the difficulty to implement each task. Tasks with a high influence on the quality are getting a higher priority to be implemented. Items without a significant impact on the quality are either sorted out or not considered in the implementation.

In the transformation model, this step aims at sorting the identified tasks into a prioritized order. This helps to focus on the most important tasks, which are promising the most benefit when implementing them at the beginning.

Step 5 and Step 6: Take the first bite and continue. These steps are slicing the tasks vertically for every iteration and define the minimum acceptable level of quality for each step. Any further bite (selection of tasks) should provide more quality to the product, regardless of what is added.

This step is to conduct the transformation process, by an incremental execution of the identified transformation tasks.

5 DISCUSSION

Agile transformation of large companies is a current topic. We identified several publications (cf. Sec. 3.4.1) presenting different approaches to transform a traditional working large company into an agile one. All presented approaches have some points in common. However, according to the results of our literature review, there does not exist any suitable transformation model preserving SPLs.

To fill this research gap, we implemented a transformation model for large companies maintaining the benefits of Software Product Line Engineering. Our transformation model is based on the results of our literature review presented in Sec. 3.4.2. Our transformation model does not claim to be comprehensive to fit for every organization and in every context. Context-specific adoptions may be necessary to identify the most useful task for the organization. The coordination of the steps also needs to be adjusted to the organization's context. It depends on the involved parties and the capability to focus on the transformation. If prerequisites such as the fundamental knowledge about agile practices are missing it interferes with the process of transformation.

The transformation model has not been validated yet. Therefore, we cannot guarantee its applicability. In future, we plan to evaluate the model in a large company. Therefore we will conduct a case study to evaluate the acceptance of the hamburger approach and the possibility to maintain the SPL during transformation. Nonetheless, parts of our model have already been validated, since it strongly focuses on already established approaches from literature. Only the new part – the SPL extensions – has not been validated yet. Furthermore, the structure of our model – predefined by the hamburger model – strives for short iterations and continuous adjustments. This facilitates the applicability of the model. In addition, we are actively involved in SPL development in a large company. This knowledge also influenced our model. Furthermore, the model was reviewed by several SPL experts from industry.

In order to preserve the benefits of SPLs, maintaining Domain and Application Engineering is necessary. We cannot give a concrete recommendation to which extent both processes need to be preserved, but we propose to provide sprints for both. Nonetheless, the applicability strongly depends on the company's culture and may need to be adjusted to the specific context.

We recommend using our model as a starting point for a successful transformation that should be adjusted to the specific needs. A procedure for the adjustments is provided in Sec. 4.4.

6 CONCLUSION

Many large companies, for example in the automotive domain, already manage their development by using SPLs. Nonetheless, they realize the need for transforming towards an agile organization to benefit from the perceived advantages. Currently, many companies are starting an agile transformation. The transformation is often done using existing transformation models and external coaches. However, none of the examined transformation models focus on the maintenance of SPLs. Consequently, there is a need for a transformation model for large companies that want to preserve the SPLs and their benefits.

We conducted a literature review to gain an overview and a better understanding of agile transformation models in large companies. In total, we analyzed 367 papers. None of them addresses the agile transformation in large settings with existing SPLs. We close this research gap by presenting a transformation model preserving the benefits of already existing models and extending and adapting the aspects which are important for Software Product Line Engineering. The transformation model bases on recommendations from the literature and on experiences from a large company. These issues are combined in a hamburger which presents an incremental approach of an agile transformation.

REFERENCES

[1] G. Adzic and D. Evans. 2014. *Fifty quick ideas to improve your user stories.* Neuri Consulting LLP.

[2] A.L. Asnawi, A.M. Gravell, and G.B. Wills. 2012. Factor analysis: Investigating important aspects for agile adoption in Malaysia. In *Proc. of the AGILE India Conference.* IEEE, 60–63.

[3] M.A. Babar, T. Ihme, and M. Pikkarainen. 2009. An Industrial Case of Exploiting Product Line Architecturesin Agile Software Development. In *Proc. of SPLC'09*, 171–179.

[4] T.J. Bang. 2007. Introducing agile methods into a project organisation. In *Proc. of XP'07.* Springer, 203–207.

[5] A. Begel and N. Nagappan. 2007. Usage and perceptions of agile software development in an industrial context: An exploratory study. In *Proc. of ESEM'07.* IEEE, 255–264.

[6] G. Benefield. 2008. Rolling out agile in a large enterprise. In *Proc. of Hawaii International Conference on System Sciences.* IEEE, 461–461.

[7] B. Boehm and R. Turner. 2005. Management challenges to implementing agile processes in traditional development organizations. *IEEE Software* 22, 5 (2005), 30–39.

[8] J. Bosch and P.M. Bosch-Sijtsema. 2011. Introducing agile customer-centered development in a legacy software product line. *Software: Practice and Experience* 41, 8 (2011), 871–882. https://doi.org/10.1002/spe.1063

[9] J. Bosch, G. Florijn, D. Greefhorst, J. Kuusela, J.H. Obbink, and K. Pohl. 2002. Variability Issues in Software Product Lines. In *Software Product-Family Engineering*, LNCS, Vol. 2290. Springer-Verlag Berlin Heidelberg, 13–21. https://doi.org/10.1007/3-540-47833-7_3

[10] A.S. Campanelli, D. Bassi, and F.S. Parreiras. 2017. Agile Transformation Success Factors: A Practitioner's Survey. In *International Conference on Advanced Information Systems Engineering.* Springer, 364–379.

[11] J. Díaz, J. Pérez, and J. Garbajosa. 2014. Agile product-line architecting in practice: A case study in smart grids. *Information and Software Technology* 56, 7 (2014), 727–748. https://doi.org/10.1016/j.infsof.2014.01.014

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

[13] A. Druckman. 2011. Agile Transformation Strategy: What does it take to become Agile? (2011).

[14] Tore Dybå and Torgeir Dingsøyr. 2008. Empirical studies of agile software development: A systematic review. *Information and Software Technology* 50, 9-10 (2008), 833–859. https://doi.org/10.1016/j.infsof.2008.01.006

[15] A. Elshamy and A. Elssamadisy. 2006. Divide after you conquer: an agile software development practice for large projects. In *Proc. of XP'06*. Springer, 164–168.

[16] C. Fry and S. Greene. 2007. Large scale agile transformation in an on-demand world. In *Proc. of AGILE'07*. IEEE, 136–142.

[17] T.J. Gandomani and M.Z. Nafchi. 2015. An empirically-developed framework for Agile transition and adoption. *Journal of Systems and Software* 107, C (2015), 204–219.

[18] T.J. Gandomani, H. Zulzalil, A.A.A. Ghani, A.B. Sultan, and K.Y. Sharif. 2014. An Exploratory Study on Managing Agile Transition and Adoption.. In *Proc. of IC2IT*. 177–188.

[19] T.J. Gandomani, H. Zulzalil, and M.Z. Nafchi. 2014. Agile Transformation: What is it about?. In *Proceedings of Malaysian Software Engineering Conference*. IEEE, 240–245.

[20] Y. Ghanam, S. Park, and F. Maurer. 2008. A Test-Driven Approach to Establishing & Managing Agile Product Lines. In *Proc. of SPLC'08*, Lero Int. Science Centre, University of Limerick, Ireland, 151–156.

[21] D.L. Giudice, H. Kisker, and N. Angel. 2014. How Can You Scale Your Agile Adoption? Technical Report. *Forrester Research Inc* . (2014).

[22] H. Hajjdiab and A.S. Taleb. 2011. Agile adoption experience: A case study in the UAE. In *Proc. of International Conference on Software Engineering and Service Science*. IEEE, 31–34.

[23] R. Hoda and J. Noble. 2017. Becoming agile: A grounded theory of agile transitions in practice. In *Proc. of ICSE'17*. IEEE Press, 141–151.

[24] P. Hohl, J. Münch, K. Schneider, and M. Stupperich. 2016. Forces that Prevent Agile Adoption in the Automotive Domain. In *Proc. of PROFES'16*, Vol. 10027. Springer International Publishing, 468–476. https://doi.org/10.1007/978-3-319-49094-6_32

[25] P. Hohl, J. Münch, K. Schneider, and M. Stupperich. 2017. Real-Life Challenges on Agile Software Product Lines in Automotive. In *Proc. of PROFES'17*, Springer International Publishing, 28–36. https://doi.org/10.1007/978-3-319-69926-4_3

[26] S. Kim, H. Lee, Y. Kwon, M. Yu, and H. Jo. 2016. Our Journey to Becoming Agile: Experiences with Agile Transformation in Samsung Electronics. In *Proc. of Asia-Pacific Software Engineering Conference*. IEEE, 377–380.

[27] B. Kitchenham, O.P. Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman. 2009. Systematic literature reviews in software engineering–a systematic literature review. *Information and software technology* 51, 1 (2009), 7–15.

[28] J. Klünder, A. Schmitt, P. Hohl, and K. Schneider. 2017. Fake News: Simply Agile. *Proc. of Projektmanagement und Vorgehensmodelle* (2017).

[29] J. Kotter and H. Rathgeber. 2009. The 8-Step Process of Successful Change. *Viitattu* 10 (2009).

[30] M. Kuhrmann, P. Diebold, and J. Münch. 2016. Software process improvement: A systematic mapping study on the state of the art. *PeerJ Computer Science* 2 (2016), e62.

[31] M. Laanti, O. Salo, and P. Abrahamsson. 2011. Agile methods rapidly replacing traditional methods at Nokia: A survey of opinions on agile transformation. *Information and Software Technology* 53, 3 (2011), 276–290.

[32] E.C. Lee. 2008. Forming to performing: Transitioning large-scale project into agile. In *Proc. of AGILE'08*. IEEE, 106–111.

[33] G. Lifshitz, A. Kroskin, and Y. Dubinsky. 2008. The Story of Transition to Agile Software Development. *Agile Processes in Software Engineering and Extreme Programming* (2008), 212–214.

[34] M. Lindvall, D. Muthig, A. Dagnino, C. Wallin, M. Stupperich, D. Kiefer, J. May, and T. Kahkonen. 2004. Agile software development in large organizations. *Computer* 37, 12 (2004), 26–34.

[35] J.A. Livermore. 2007. Factors that impact implementing an agile software development methodology. In *Proc. of SoutheastCon*. IEEE, 82–86.

[36] A. Mahanti. 2006. Challenges in enterprise adoption of agile methods - A survey. *Journal of Computing and Information Technology* 14, 3 (2006), 197–206.

[37] C. Maples. 2009. Enterprise agile transformation: the two-year wall. In *Proc. of AGILE'09*. IEEE, 90–95.

[38] A. Martini, L. Pareto, and J. Bosch. 2013. Communication factors for speed and reuse in large-scale agile software development. In *Proc. of the SPLC'13*, 42. https://doi.org/10.1145/2491627.2491642

[39] J.D. McGregor. 2008. Agile Software Product Lines - A Working Session. In *Proc. of SPLC'08*. 364. https://doi.org/10.1109/SPLC.2008.60

[40] M.E. Moreira. 2013. Three Case Studies in Adopting Agile. In *Being Agile*. Springer, 233–248.

[41] J. Münch, K. Schmid, and H.D. Rombach. 2013. *Perspectives on the future of software engineering: Essays in honor of Dieter Rombach*. Springer.

[42] P. O'Leary, F. McCaffery, S. Thiel, and I. Richardson. 2012. An agile process model for product derivation in software product line engineering. *Journal of Software: Evolution and Process* 24, 5 (2012), 561–571. https://doi.org/10.1002/smr.498

[43] H. Holmström Olsson and J. Bosch. 2014. Towards agile and beyond: An empirical account on the challenges involved when advancing software development practices. In *Proc. of XP'2014*. Springer, 327–335.

[44] G. Papadopoulos. 2015. Moving from Traditional to Agile Software Development Methodologies – Also on Large, Distributed Projects. *Proc. of Social and Behavioral Sciences* 175 (2015), 455–463.

[45] R.M. Parizi, T.J. Gandomani, and M.Z. Nafchi. 2014. Hidden facilitators of agile transition: Agile coaches and agile champions. In *Proc. of the Malaysian Software Engineering Conference*. IEEE, 246–250.

[46] J. Patton and P. Economy. 2014. *User story mapping: Discover the whole story, build the right product*. O'Reilly Media, Inc..

[47] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson. 2008. Systematic Mapping Studies in Software Engineering. In *Proc. of EASE'08*, 68–77. http://www.bcs.org/upload/pdf/ewic_ea08_paper8.pdf

[48] M. Pikkarainen, O. Salo, and J. Still. 2005. Deploying agile practices in organizations: A case study. *Software Process Improvement* (2005), 16–27.

[49] R. Popli, N. Chauhan, et al. 2013. A mapping model for transforming traditional software development methods to agile methodology. *International Journal of Software Engineering & Applications* 4, 4 (2013), 53.

[50] A. Rasnacis and S. Berzisa. 2017. Method for Adaptation and Implementation of Agile Project Management Methodology. *Proc of Computer Science* 104 (2017), 43–50.

[51] A. Rohunen, P. Rodriguez, P. Kuvaja, L. Krzanik, and J. Markkula. 2010. Approaches to agile adoption in large settings: A comparison of the results from a literature analysis and an industrial inventory. *Proc. of PROFES'10* (2010), 77–91.

[52] G. Roman, S. Marczak, A. Dutra, and R. Prikladnicki. 2015. On the Agile Transformation in a Large-Complex Globally Distributed Company: Why Boarding this Journey, Steps Taken and Main Foreseen Concerns. In *Proc. of the Brazilian Workshop on Agile Methods*. IEEE, 32–39.

[53] M. Sahota, J. Appelo, and H. Kniberg. 2012. *An agile adoption and transformation survival guide: Working with organizational culture*. InfoQ.

[54] A. Jr. Santos and V. Jr. Lucena. 2010. SCRUMPL - Software Product Line Engineering with SCRUM. In *Proc. of ENASE'10*. 239–244. https://doi.org/10.5220/0003038302390244

[55] K. Schwaber. 2004. *Agile project management with Scrum*. Microsoft Press. http://proquestcombo.safaribooksonline.com/book/software-engineering-and-development/agile-development/9780735619937

[56] H. Smits and K. Rilliet. 2011. Agile experience report: Transition and complexity at cisco voice technology group. In *Proc. of AGILE'11*. IEEE, 274–278.

[57] K. Tian and K. Cooper. 2006. Agile and Software Product Line Methods: Are They So Different? (2006).

[58] A. Tomasini and M. Kearns. 2012. Agile Transition – What You Need to Know Before Starting. *InfoQueue Enterprise Software Development Series* (2012).

[59] R. Valade. 2008. The Big Projects Always Fail: Taking an Enterprise Agile. In *Proc. of AGILE'08*. IEEE, 148–153.

[60] K. Vilkki. 2008. Juggling with the Paradoxes of Agile Transformation or How to survive in a large scale agile transformation. *An article in FLEXI Newsletter (February 2008)* (2008).

[61] S. Weber. April 2015. Agile in Automotive âĂŞ State of Practice 2015. (April 2015). www.kuglermaag.com/agile2015

[62] K. Wiklund, D. Sundmark, S. Eldh, and K. Lundqvist. 2013. Impediments in Agile Software Development: An Empirical Investigation. In *Proc. of PROFES'13*. Vol. 7983. Springer Berlin Heidelberg, 35–49. https://doi.org/10.1007/978-3-642-39259-7_6

[63] D. Wildt and R. Prikladnicki. 2010. Transitioning from Distributed and Traditional to Distributed and Agile: An Experience Report. In *Agility Across Time and Space*. Springer, 31–46.

[64] C. Wohlin. 2016. Second-generation systematic literature studies using snowballing. In *Proc. of EASE'16*. ACM, 15.

[65] C. Wohlin, M. Höst, and K. Henningsson. 2003. Empirical Research Methods in Software Engineering. In *Empirical Methods and Studies in Software Engineering: Experiences from ESERNET*. Springer Berlin Heidelberg, 7–23. https://doi.org/10.1007/978-3-540-45143-3_2

[66] F.A. Zainal Abidin. 2017. Agile Transition Model Based on Human Factors. *International Journal of Innovative Computing* 7, 1 (2017).