Mapping Agility to Automotive Software Product Line Concerns

Philipp Hohl[1(✉)], Sven Theobald[2], Martin Becker[2], Michael Stupperich[1], and Jürgen Münch[3]

[1] Daimler AG, Research and Development, Ulm, Germany
{philipp.hohl,michael.stupperich}@daimler.com

[2] Fraunhofer Institute for Experimental Software Engineering IESE, Kaiserslautern, Germany
{sven.theobald,martin.becker}@iese.fraunhofer.de

[3] Reutlingen University, Reutlingen, Germany
juergen.muench@reutlingen-university.de

Abstract. Context: Software product lines are widely used in automotive embedded software development. This software paradigm improves the quality of software variants by reuse. The combination of agile software development practices with software product lines promises a faster delivery of high quality software. However, the set up of an agile software product line is still challenging, especially in the automotive domain. **Goal:** This publication aims to evaluate to what extend agility fits to automotive product line engineering. **Method:** Based on previous work and two workshops, agility is mapped to software product line concerns. **Results:** This publication presents important principles of software product lines, and examines how agile approaches fit to those principles. Additionally, the principles are related to one of the four major concerns of software product line engineering: Business, Architecture, Process, and Organization. **Conclusion:** Agile software product line engineering is promising and can add value to existing development approaches. The identified commonalities and hindering factors need to be considered when defining a combined agile product line engineering approach.

Keywords: Software product line · Agile · Agility
Automotive software development · Commonalities · Hindering factors

1 Introduction

The automotive domain is recently in a disruptive change. Vehicles are no longer isolated, but connected to the environment [1]. The uprising complexity has to be addressed for the development of new car generations. A lot of innovation is nowadays addressed in software. According to Oliveira [2], 85% of the functionalities in cars are realized with software running on electronic control units. The amount of software has evolved from zero to tens of millions of lines of code

[3], distributed on more than 70 electronic control units [4,5]. In 2006, Broy [6] already identified the increasing amount of software. He stated that the amount of software in the car has increased exponentially in the last decades [6]. McCaffery et al. [7] expect that the quantity of software grows continuously. They further emphasize that software must be developed faster and in a more cost effective way, but still in a high quality [7].

The reuse of software parts is necessary to keep pace with the high amount of different variants, while simultaneously maintaining the quality of the software. Software product lines are a paradigm for systematic software reuse [8]. This paradigm is particularly important for the automotive domain to meet different requirements across multiple markets [9]. According to Wozniak et al. [10], the automotive domain is the most challenging environment for systems and software product line engineering. Millions of different software variants exist, whereby each one of them comprises a large complexity [10]. The complexity is based on the large number of variation points within the product [10,11]. There exist several variant management approaches, methods and processes to handle variants [12]. However, it is important to ensure that the approach is suitable for the automotive domain. The consistency and reliability of systems has to be ensured at all times [8] and must be compliant to valid norms and restrictions given by the law.

Traditional working structures make it difficult to deliver the required high amount of software fast enough [6], because the development processes are too slow to keep pace with the fast changing market [13,14]. Since 2001, agile software development methods promise a fast delivery of high-quality products. Furthermore it is possible to react dynamically on changing requirements from customers and market demands, due to close customer collaboration and incremental development. The adoption of agile practices within the automotive domain is therefore a possible way to keep pace with fast changing market demands [15].

However, the combination of agile software development and software product lines for software development is assumed to be difficult [15]. Agile methods and practices are primarily designed for short development cycles in small development teams. For larger teams, agile methods are scaled up to approaches for large organizations, such as the Scaled Agile Framework or Scrum of Scrums. In the automotive domain, software product lines are used to manage the software development, by "intra-organisational reuse through the explicitly planned exploitation of similarities between related products" [9, p. 531].

Various combinations of agile software development and plan-driven development approaches are already present within parts of the automotive domain. These hybrid approaches are often very specific to the context they are used and rarely consider a strategic software reuse [16]. In order to ensure the quality of a agile software product lines processes, the aim of this paper is to identify competing and aligned goals for the combination of agile elements and software product lines. Considering the goals, it is possible to define a development process to maintain long-term productivity, efficiency, and profit (cf. [17])

The remainder of this paper is structured as follows. Section 2 discusses related work. Section 3 describes the research approach. Section 4 presents the key findings and Sect. 5 discusses these findings. Finally, Sect. 6 summarizes the paper and gives an outlook on future work.

2 Related Work

This section investigates publications for a combination of software product lines and agile software development in the automotive domain.

2.1 Agile Software Product Lines in Automotive

The published literature does not provide significant information or approaches on how to adapt existing agile software development approaches to software product line development in the automotive domain.

Thiel et al. [9] analyze the combination of agile and plan-driven processes. This combination could be seen as a typical characteristic of current automotive software development. They emphasize that a combination is beneficial under certain conditions, such as rigid quality and safety requirements [9]. They further suggest to introduce selected agile practices and methods to automotive systems engineering [9]. This suggestion follows as well from the "Agile in Automotive Survey" from Kugler and Maag [18]. The survey identifies that many companies only introduce single agile elements into existing and proven development cycles. In addition, the use of Scrum as defined in [19] is often not applicable due to the context factors and the development environment. In most cases, Scrum is tailored to the specific context [20].

Schlosser et al. [21] define an imminent challenge for automotive software development. They mention the necessity of shorter development time for multiple software variants by considering constraints of a high cost pressure. They further argue that more incremental software deliveries in a shorter time are necessary. With the incremental approach, a quick response from customers could be achieved, which leads to a shorter development time and saves money [15,21].

In addition, the agile element of *Continuous Integration* is seen as a key component for agile automotive development. Continuous Integration provides a quick response with respect to functional aspects [21–23].

In order to integrate further agile elements into the automotive domain, different models and processes are introduced, such as the *Feedback Loop Model* and the *Mega Scale Software Product Line Engineering*.

The concurrent *Feedback Loop Model* introduces feedback loops for different organizations to enable cooperation [24]. The cooperation (including communication) between different organizations is managed by a new architect role to shorten the development time. The reduction of time is validated in a case study. Furthermore, the *System Architecture Virtual Integration* (SAVI) initiative helps to improve the agile development. In SAVI, Continuous Integration is concurrently operated within the software development process. Integration starts with the earliest available system models into a virtual integration environment. With a virtual integration strategy vendors are more closely tied to the project [1].

In order to handle the complexity, the *Mega Scale Software Product Line Engineering* (MS-SPLE) approach is introduced. This approach is applied for large product sets with complex products and complex feature variations like the automotive domain. MS-SPLE is a possibility to manage, e.g., calibration parameters for the software variation mechanism and the complexity management. However, agile software development is not taken into account within MS-SPLE [10].

In total, four different trends could be identified.

– Adopting agile software development in the automotive typically concentrates only on selected agile practices such as Continuous Integration or Pair Programming. The published literature does not show any recommendations to use a comprehensive set of agile elements and practices together in the automotive domain.
– The majority of the published literature suggests that agile models and processes should get customized to the specifics of the automotive domain before they are implemented in practice.
– Agile models and processes that are already customized to the specifics of the automotive domain are proposed in the published literature. An example is the *Feedback Loop Model* that especially considers the collaboration between different organizations (such as OEMs and suppliers).
– Combination approaches include interesting new concepts such as virtual integration on the system level.

3 Research Approach

The research presented in this publication is guided by the question:
Which commonalities and hindering factors need to be considered when combining software product lines and agile development? This question identifies how well both approaches can be combined. It further examines incompatibilities which need special attention for setting up a combination.

3.1 Research Design

Hohl et al. conducted a qualitative interview study [25], a literature review [26] and gave some recommendations on the combination of agile development and software product lines in the automotive domain. Our research is based on these publications and on two workshops. These expert workshops aimed at achieving a common understanding how to combine agile software development and software product lines in the automotive domain. We combined the experience of the authors due to the active involvement of Researcher 1 (P. Hohl) and Researcher 4 (M. Stupperich) in the automotive software development and the experience of Researcher 2 (S. Theobald) regarding agile practices. Furthermore, we included the experience of Researcher 3 (M. Becker) in software product lines and the knowledge of Researcher 5 (J. Münch) in software engineering. The workshops were held at the "Fraunhofer Institute for Experimental Software Engineering" (IESE) in Kaiserslautern, Germany. In both workshops, researchers from Fraunhofer IESE attended the workshop as additional participants. Each participant was invited by Researcher 1 and Researcher 3 to participate in the workshop due to their experience in agile development practices and the use of software product lines.

The first workshop took place in September 2017. This one day workshop was divided into two sessions of 3 h each. Industrial participation was given by Researcher 1 and Researcher 4 working at the Daimler AG. Furthermore, four participants (including Researcher 2 and Researcher 3) from the departments of embedded systems engineering and process engineering from the Fraunhofer IESE participated in the workshop. In the first session, each of the six participants gave a short presentation about their experience on the topic. Researcher 1 presented his thoughts on assessment models for agile software product line engineering.

The second session was a discussion round to identify common ground, to interconnect the knowledge of the participants and link agile practices and software product line techniques together. Within the workshop, the "Business (B), Architecture (A), Process (P) and Organization (O) (BAPO)" Model was identified as the basis for a discussion to evaluate agile principles according to the major concerns of software product line engineering [27]. The end of the session was the identification of topics for discussion in the follow-up meeting.

Researcher 3 prepared a list of product line engineering principles and categorized them according to the BAPO Model. Researcher 1 and Researcher 2 reviewed the categorization and completeness of the principles prior to the follow-up meeting. Researcher 3 prioritized the principles from the software product line view and Researcher 2 rated the importance of those principles for supporting the agile way of working. Researcher 1 reviewed the combined results.

The follow-up workshop took place in January 2018 with 4 participants from the first workshop (including Researcher 1, Researcher 2 and Researcher 3). It consists of two sessions of 3 hours each. In the first session the results which were generated between the workshops were presented and discussed. In the second session, the results were presented to an independent researcher. The subsequent discussion provided feedback to the categorization.

3.2 Threats to Validity

Threats to the validity are described and a mitigation strategy for each threat is given.

Bias may be introduced by the researchers, which were also the workshops participants. The completeness and correctness of the mapping could be influenced by the experience of the workshop participants. However, we limit this threat by including researchers from different fields and several review iterations.

Furthermore, the categorization was reviewed by an independent researcher. With this approach, a balanced view on the combination was possible.

The categorization into the four major concerns of software product line engineering: Business (B), Architecture (A), Process (P) and Organization (O) [27] could influence the result for the identification of commonalities and hindering factors. We addressed this threat by a review process and a discussion with an independent and unbiased researcher from the Fraunhofer IESE in the second session of the follow-up workshop.

4 Results and Analysis

Which commonalities and hindering factors need to be considered when combining software product lines and agile development?

To identify the potential combination we introduced a list with a comparison of agile and software product line approaches. Each list item is related to one of the four major concerns of software product line engineering: Business (B), Architecture (A), Process (P) and Organization (O) [27] and organized in two categories. The categorization is based on the Family Evaluation Framework (FEF) proposed by van der Linden [27,28], which has been developed in the CAFÉ/Families EU projects. It provides the four dimensional evaluation scheme to assess the readiness or maturity of product line adoption within an organization. Commonalities and hindering factors are identified (cf. Fig. 1).

The mappings are denoted as *"X-Description"*, whereas *"X"* defines one of the four dimensions (B,A,P,O) and *"Description"* names the identified principle.

4.1 Aligned Goals

This subsection presents the aligned goals. The principles were denoted as aligned if both approaches show the same trend (cf. Fig. 1).

B-Know Feature Cost, B-Know Feature Value. Agile development focuses on individual single-system products. The costs for each features are estimated. A prediction of the overall costs cannot be done precisely at the start. The economic justification for introducing product line engineering is the reduction of costs [29]. The concept of software reuse is simple. Reuse already implemented software parts saves the cost of designing, writing and testing new code [30]. Budget steering is essential to balance reuse on the one hand and to foster customer orientation on the other hand [31]. Benefits of a reuse program must always be considered in the long term [30].

B-Know Company Goals, B-Aligned Strategies, B-Plan Feature Roadmap/Evolution. In agile development, it is helpful to consider the company goals and align with company strategies, but not always needed, e.g., a single team can develop a product for a customer independently of the rest of the organization. In software product line engineering the company goals are defined for the long term and considered to direct the evolution of the software product line.

P-Predictable Quality. In product line engineering, all features are extensively tested and a higher quality is achieved by using the core assets in multiple products. In agile development, short iterations and continuous testing help to avoid a big-bang integration. The learning loop about the maturity of the software is shorter and failures in the software are detected earlier in the development process. High qualitative software is mandatory for automotive software. Long lasting software tests and tests with real cars help to achieve a high software quality and to be compliant with the restrictions given by the law [25].

O-Share Domain Knowledge. Agile methods and Product line engineering aim at including experts to share their domain knowledge. The ideal agile team follows the one-room principle to center the cross-functional team. In product line engineering, domain experts are systematically involved to identify common parts in the scoping process. The development processes in the automotive domain are organized in a hierarchical structure. Coordination of the worldwide distributed software development requires to share domain knowledge to avoid a diverge of the common software baseline [25].

4.2 Competing Goals

This subsection presents the identified competing goals based on the BAPO model for product line engineering [27]. The principles were denoted as competing if one approach fosters (Helpful, Needed) and the other one hinders (Hampering, Impending) an agile software product line (cf. Fig. 1).

A-Ease Customization. Agile development is customer-oriented. The use of agile methods enables the development to react on customers' needs and changing requirements [32]. Customer oriented development focuses on a specific problem within a standalone solution. The idea of the software product line is to develop different products from one core asset base (also called a platform) [33]. Pohl et al. [29] mention the platforms used for mass-customization in their definition of software product line engineering. For software customization, customer-specific product variants can be generated using parts from the platform and customize it to fulfill customers' demands [31].

A-Share Architecture, A-Share Assets, P-Manage Interdependencies. In product line engineering, it is important to manage the traceability between requirements, architecture, code and tests [31]. Small changes in core assets can have an effect on multiple systems. The complex interdependencies between variants in a product line need special attention and require strategic long-term

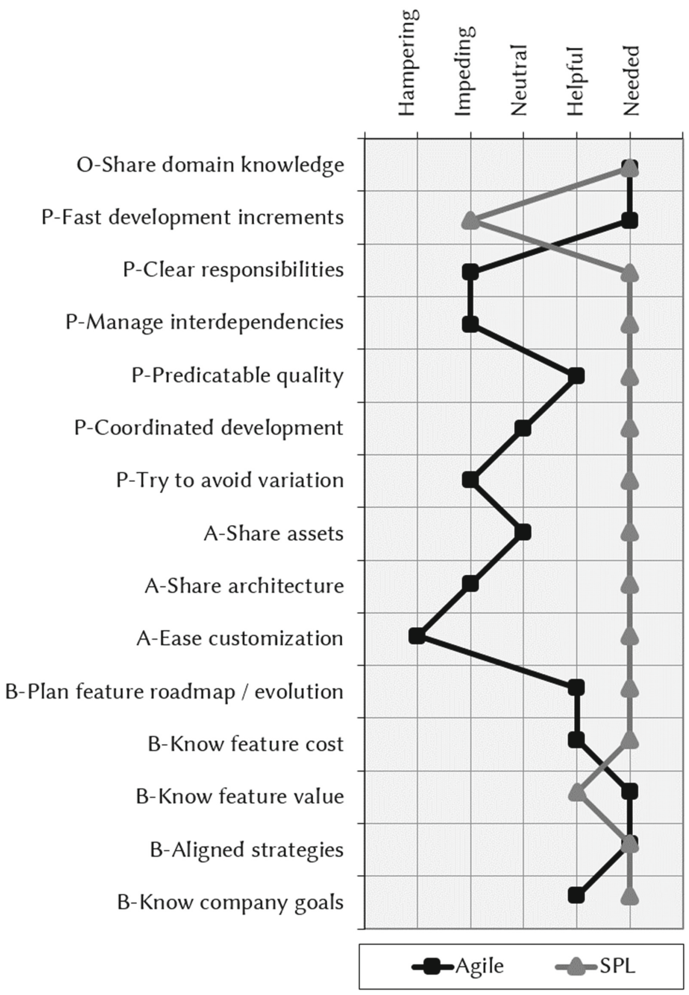

Fig. 1. Commonalities and hindering factors for the combination of agile and software product line approaches according to the four major concerns of software product line engineering: Business (B), Architecture (A), Process (P) and Organization (O) [27].

planning [34]. In contrast, agile development focuses on single product development and changes in the code or architecture only affect the single product. The software development in the automotive domain requires a coordination that addresses the interdependencies with other parties such as suppliers, in-house development, mechanical and hardware development.

P-Try to Avoid Variation. Agile methods and practices do not explicitly address the development of several similar products at the same time. Therefore, a structured reuse of software between several products for different customers is not considered. The development often follows a so called "clone and own" approach. In contrast, the software product line supports the structured development of different software variants. Members of a product family, addressing the same domain, are developed together. The reusability of domain assets is one success factors of the software product line [34]. The reusable assets are assembled into customer-specific software systems [35]. However, the software product line requires strategic long-term planning [34]. This is competing with the flexible planning fostered by agile methods. In the automotive domain, it is still challenging to plan in the long term, but equally important to be able to react on changing market demands in the short term.

P-Clear Responsibilities, P-Coordinated Development. Blau and Hildenbrand [31] point out that it is necessary to manage the development process and to identify possible reusable software parts, called core assets, to avoid redundant development [31]. Complex interdependencies between individual products in the product line require coordination and defined responsibilities for the development process. In agile methods, the concept of collective code ownership gives the responsibility to the team. However, current organizational hierarchy in the automotive domain likely keeps responsibility in higher management levels.

P-Fast Development Increments. Agile development focuses on customer demands and allows for small iterations where product increments are produced. Changes in requirements are welcomed and thus changes happen regularly based on the feedback collected at the end of each iteration. In product line engineering, new requirements and changes in the planned core assets cannot be addressed ad-hoc, since the impact to the whole product line has to be evaluated first.

5 Discussion

The categorization into the four major concerns of software product line engineering: Business (B), Architecture (A), Process (P) and Organization (O) [27] resulted in commonalities and hindering factors.

On a high-level view (cf. Fig. 1), our comparison shows alignment on business (B) and organizational (O) level. The aligned goals show that a combination is possible with a low effort. In the comparison, both approaches (agile software development and software product line engineering) show the same trend to address issues such as feature cost, feature value, and software quality. This confirms our understanding that both approaches follow similar goals.

However, guidelines are needed to combine both approaches in a structured way. This is even more important for the competing goals, as they are impeding a combination. Differences occur on the level of architecture (A) and processes (P). Maintaining a shared architecture to address the large amount of different software variants is challenging. The interdependencies of variants need special attention regarding the development process. The competing goals need to be considered carefully to combine agile development and software product lines successfully.

An assessment model is expected to be beneficial to guide the combination of agile software development with software product line engineering in order to shorten time to market and maintain long-term productivity, efficiency, and profit [17]. Hohl et al. [36] present an assessment model that could support the combination, by comprising the identified compatibility of both approaches. Furthermore, such an assessment model shall be aligned to existing standards, such as ASPICE [37] and comprises the major concerns of software product line engineering.

Existing literature [21–23] introduce agile elements into plan-driven processes. However, none of the mentioned publications provides a recording of the actual situation concerning product lines and agile development processes.

6 Conclusion

This publication identifies whether agility fits to the presented product line engineering principles. We identify commonalities and hindering factors of agile elements and software product lines. Therefore, we categorized the important characteristics to the four major concerns of software product line engineering: Business (B), Architecture (A), Process (P) and Organization (O) [27].

The results show that both approaches can be successfully combined. Aligned goals support a combination with low effort, because both approaches follow the same goal. Competing goals must be considered carefully.

The results motivate the development of an adjusted assessment model to support a successful introduction. An in-depth process knowledge and management of software variants can make a difference for success in the case of reuse based software development. An assessment model can foster the combination by reinforcing the implementation of the aligned goals and supporting an effective management program by explicitly including important areas to consider. Other domains that apply software product lines may benefit from the assessment model, while assessing the possibilities of combining agile software development with the existing software product line.

For future work, we plan to extend and validate the factors for a successful combination, by including other standards of software product line engineering. Further, the mapping of both approaches will be refined in a more granular way, to get a better understanding of the conflicts and synergies. Especially, the agile view will be considered more extensively.

References

1. Taiber, J., McGregor, J.D.: Efficient engineering of safety-critical, software-intensive systems. In: 2014 International Conference on Connected Vehicles and Expo (ICCVE), pp. 836–841
2. Oliveira, P., Ferreira, A.L., Dias, D., Pereira, T., Monteiro, P., Machado, R.J.: An analysis of the commonality and differences between ASPICE and ISO26262 in the context of software development. In: Stolfa, J., Stolfa, S., O'Connor, R.V., Messnarz, R. (eds.) EuroSPI 2017. CCIS, vol. 748, pp. 216–227. Springer, Cham (2017)
3. Pretschner, A., Salzmann, C., Schätz, B., Staudner, T.: Fourth International Workshop on Software Engineering for Automotive Systems (SEAS 2007): Proceedings, ICSE 2007 Workshops, 20–26 May 2007, Minneapolis ICSE 2007. IEEE, Piscataway (2007)
4. Contag, M., et al.: How they did it: an analysis of emission defeat devices in modern automobiles. In: 2017 IEEE Symposium on Security and Privacy (SP), pp. 231–250. IEEE (2017)
5. Münch, J., Schmid, K., Rombach, H.D.: Perspectives on the Future of Software Engineering: Essays in Honor of Dieter Rombach. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-37395-4
6. Broy, M.: Challenges in automotive software engineering. In: Proceedings of the 28th International Conference on Software Engineering, ICSE 2006, pp. 33–42. ACM (2006)
7. McCaffery, F., Pikkarainen, M., Richardson, I.: Ahaa-agile, hybrid assessment method for automotive, safety critical smes. In: Schäfer, W. (ed.) Companion of the 30th International Conference on Software Engineering, p. 551. ACM (2008)
8. Leitner, A., Mader, R., Kreiner, C., Steger, C., Weiß, R.: A development methodology for variant-rich automotive software architectures. e & i Elektrotechnik und Informationstechnik **128**(6), 222–227 (2011)
9. Thiel, S., Babar, M.A., Botterweck, G., O'Brien, L.: Software product lines in automotive systems engineering. SAE Int. J. Passeng. Cars Electron. Electr. Syst. **1**(1), 531–543 (2009)
10. Wozniak, L., Clements, P.: How automotive engineering is taking product line engineering to the extreme. In: Schmidt, D.C. (ed.) Proceedings of the 19th International Conference on Software Product Line, pp. 327–336. ACM (2015)
11. Galster, M., Avgeriou, P.: Supporting variability through agility to achieve adaptable architectures. In: Agile Software Architecture, pp. 139–159. Elsevier (2014)
12. Hohl, P., Münch, J., Stupperich, M.: Forces that support agile adoption in the automotive domain. In: Software Engineering (2017)
13. Bosch, J., Bosch-Sijtsema, P.M.: Introducing agile customer-centered development in a legacy software product line. Softw. Pract. Exp. **41**(8), 871–882 (2011)
14. Eliasson, U., Heldal, R., Lantz, J., Berger, C.: Agile model-driven engineering in mechatronic systems - an industrial case study. In: Dingel, J., Schulte, W., Ramos, I., Abrahão, S., Insfran, E. (eds.) MODELS 2014. LNCS, vol. 8767, pp. 433–449. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-11653-2_27
15. Katumba, B., Knauss, E.: Agile development in automotive software development: challenges and opportunities. In: Jedlitschka, A., Kuvaja, P., Kuhrmann, M., Männistö, T., Münch, J., Raatikainen, M. (eds.) PROFES 2014. LNCS, vol. 8892, pp. 33–47. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-13835-0_3

16. Hohl, P., Stupperich, M., Münch, J., Schneider, K.: Die variantenvielfalt agil managen: agile software-produktlinien im automobilsegment. In: Tagungsband - Embedded Software Engineering Kongress 2016 : 28. November bis 2. Dezember 2016, Sindelfingen, pp. 427–433 (2016)
17. Martini, A., Pareto, L., Bosch, J.: Communication factors for speed and reuse in large-scale agile software development. In: Jarzabek, S. (ed.) Proceedings of the 17th International Software Product Line Conference, p. 42. ACM, New York (2013)
18. KUGLER MAAG CIE GmbH: Agile in automotive - state of practice 2015, April 2015
19. Schwaber, K., Sutherland, J.: The Scrum Guide (2001)
20. Diebold, P., Ostberg, J.-P., Wagner, S., Zendler, U.: What do practitioners vary in using scrum? In: Lassenius, C., Dingsøyr, T., Paasivaara, M. (eds.) XP 2015. LNBIP, vol. 212, pp. 40–51. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-18612-2_4
21. Schloßer, A., Schnitzler, J., Sentis, T., Richenhagen, J.: Agile processes in automotive industry – efficiency and quality in software development. In: Bargende, M., Reuss, H.C., Wiedemann, J. (eds.) 16. Internationales Stuttgarter Symposium. Proceedings, pp. 489–503. Springer, Wiesbaden (2016). https://doi.org/10.1007/978-3-658-13255-2_35
22. Valade, R.: The big projects always fail: Taking an enterprise agile. In: Agile 2008 Conference, pp. 148–153 (2008)
23. Antinyan, V., et al.: Identifying risky areas of software code in agile/lean software development: an industrial experience report. In: 2014 Software Evolution Week - IEEE Conference on Software Maintenance, Reengineering and Reverse Engineering (CSMR-WCRE), pp. 154–163 (2014)
24. Hayashi, K., Aoyama, M., Kobata, K.: A concurrent feedback development method and its application to automotive software development. In: 2015 Asia-Pacific Software Engineering Conference (APSEC), pp. 362–369 (2015)
25. Hohl, P., Münch, J., Schneider, K., Stupperich, M.: Forces that prevent agile adoption in the automotive domain. In: Abrahamsson, P., Jedlitschka, A., Nguyen Duc, A., Felderer, M., Amasaki, S., Mikkonen, T. (eds.) PROFES 2016. LNCS, vol. 10027, pp. 468–476. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-49094-6_32
26. Hohl, P., Ghofrani, J., Münch, J., Stupperich, M., Schneider, K.: Searching for common ground: existing literature on automotive agile software product lines. In: Bendraou, R., Raffo, D., LiGuo, H., Maggi, F.M. (eds.) Proceedings of the 2017 International Conference on Software and System Process - ICSSP 2017, pp. 70–79. ACM Press (2017)
27. van der Linden, F.: Family evaluation framework overview & introduction, 29 August 2005
28. van der Linden, F., Schmid, K., Rommes, E.: Software Product Lines in Action. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-71437-8
29. Pohl, K., Böckle, G., Linden, F.: Software Product Line Engineering: Foundations, Principles, and Techniques. Springer, Heidelberg (2005). https://doi.org/10.1007/3-540-28901-1
30. Wentzel, K.D.: Software reuse&mdash;facts and myths. In: Proceedings of the 16th International Conference on Software Engineering, ICSE 1994, pp. 267–268. IEEE Computer Society Press (1994)

31. Blau, B., Hildenbrand, T.: Product line engineering in large-scale lean and agile software product development environments - towards a hybrid approach to decentral control and managed reuse. In: 2011 Sixth International Conference on Availability, Reliability and Security (ARES), pp. 404–408 (2011)
32. Black, S., Boca, P.P., Bowen, J.P., Gorman, J., Hinchey, M.: Formal versus agile: survival of the fittest. Computer **42**(9), 37–45 (2009)
33. Buckle, G., Clements, P.C., McGregor, J.D., Muthig, D., Schmid, K.: Calculating roi for software product lines. IEEE Softw. **21**(3), 23–31 (2004)
34. International Organization for Standardization: Software and systems engineering – reference model for product line engineering and management, 01 December 2015
35. Tian, K.: Adding more agility to software product line methods: a feasibility study on its customization using agile practices. Int. J. Knowl. Syst. Sci. **5**(4), 17–34 (2014)
36. Hohl, P., Stupperich, M., Münch, J., Schneider, K.: An assessment model to foster the adoption of agile software product lines in the automotive domain
37. VDA QMC Working Group 13/Automotive SIG: Automotive spice process assessment/reference model, 01 November 2017