Towards Agile Product Derivation in Software Product Line Engineering

Padraig O'Leary[1], Fergal McCaffery[2], Ita Richardson[1], Steffen Thiel[1]

[1]Lero, the Irish Software Engineering Research Centre, University of Limerick, Ireland
[2]Dundalk Institute of Technology, Dundalk, Ireland

{padraig.oleary, ita.richardson, steffen.thiel}@lero.ie
fergal.mccaffery@dkit.ie

Abstract. Software Product Lines (SPL) and Agile practices have emerged as new paradigms for developing software. Both approaches share common goals; such as improving productivity, reducing time to market, decreasing development costs and increasing customer satisfaction. These common goals provide the motivation for this research. We believe that integrating Agile practices into SPL can bring a balance between agility and formalism. However, there has been little research on such integration. We have been researching the potential of integrating Agile approaches in one of the key SPL process areas, product derivation. In this paper we present an outline of our Agile framework for product derivation that was developed through industry based case study research.

Keywords: Software Product Lines, Product Derivation, Agile Approaches

1 Introduction

Both Agile and Software Product Lines (SPL) development paradigms are being promoted as means of reducing time to market, increasing productivity, and gaining cost effectiveness and efficiency of software development efforts [1]. Furthermore, both approaches assume that requirement changes will occur and can be managed effectively [1]. These goals (shared by Agile and SPL) open the possibilities of introducing Agile practices into SPL activities. There are, however, several challenges involved in integrating Agile approaches in SPL development due to certain differences that exist in the philosophies of both approaches such as design and change management strategies [1, 2]. Moreover, Agile approaches do not purpose to develop flexible artefacts for reuse [2, 3] or develop and maintain rigorous and extensive documentation as required by SPL [3].

Our research in SPL is aimed at improving the Product Derivation (PD) process, which purports to develop new products by utilizing core assets of a SPL such as

feature models, architecture models, and code artefacts [4], through the adoption of Agile practices.

In this paper we present our research results on the development of an Agile Framework for Product Derivation (AFPD). We decided to concentrate on product derivation as it is considered one of the most important and challenging SPL "activities" [5], and the activity which has the most to gain from the successful implementation of agile practices. We believe that any successful effort to introduce Agile practices in the product derivation process can make SPL significantly more effective and efficient. While some research in the area of Agile SPL has been reported [1-3, 6-8], there has been little research conducted on the use of Agile approaches in the product derivation process.

The remainder of this paper is organised as follows: Section 2 describes the key concepts of SPL and Agile practices. Section 3 discusses the research methodology. Section 4 presents an overview of our Agile Product Derivation Framework. In Section 5, we discuss in detail the Agile aspects of the AFPD. The paper concludes in Section 6 with a summary and an outlook of future work.

2 Background and Motivation

In the following section, we discuss the main concepts of Agile and SPL that underpins our proposal for integrating the two.

2.1 Software Product Lines

A SPL is a set of software-intensive systems that share a common, managed set of features satisfying the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way [6]. The SPL approach makes a distinction between domain engineering, where a common platform for an arbitrary number of products is designed and implemented, and application engineering, where a product is derived based on the platform components [8]. It is during application engineering that the individual products within a product line are constructed. The process of creating these individual products using the shared artefacts is known as the product derivation process [4].

The underlying assumption of product derivation is that "the investments required for building the reusable assets during domain engineering are outweighed by the benefits of rapid derivation of individual products" [4]. This assumption might not hold if inefficient derivation practices diminish the expected gains.

A number of publications discuss the difficulties associated with product derivation. Hotz *et al.* [9] describe the process as "slow and error prone even if no new development is involved". Deelstra *et al.* [4] observe that the derivation of individual products from shared software assets is still a time-consuming and expensive activity in many organisations. The authors state that "there is a lack of methodological support for application engineering and, consequently, organizations fail to exploit the full benefits of software product families." "Guidance and support

are needed to increase efficiency and to deal with the complexity of product derivation" [10].

2.2 Agile Practices

Agile practices have recently gained popularity among large numbers of companies as a mechanism for reducing costs and increasing ability to handle change in dynamic market conditions. Researchers and practitioners have proposed several software development approaches based on the principles of the Agile manifesto [11, 12]. Two of these approaches are: eXtreme Programming (XP) [13] and Scrum [14].

XP evolved from the problems caused by the long development cycles of traditional development models [15]. The individual practices of XP are not new, however, the practices have been collected and lined up to function with each other in a novel way. The term 'extreme' comes from taking these commonsense principles and practices to extreme levels [16].

Scrum provides a project management framework that focuses development into 30-day Sprint cycles in which a specified set of Backlog features are delivered [14]. The core practice in Scrum is the use of daily 15-minute team meetings for coordination and integration. Scrum does not define any specific software development techniques. Scrum concentrates on how team members should function in order to produce good quality code and maintain flexibility in a changing environment.

Although XP and Scrum are based on a common guideline defined by the Agile manifesto, they vary in focus and presentation. XP emphasises technical elements of the development lifecycle, while Scrum concentrates on the project management.

3. Research Approach

The preparatory stage of this research was conducted as an extensive literature review. The research aimed to identify the fundamental practices of product derivation and Agile approaches. The initial results were further developed and assessed through a series of iterative workshops over a four month period. Evidence and feedback from SPL and Agile experts was collected from these organised workshops.

We conducted case study research with Robert Bosch GmbH [1]. We collected data on the product derivation practices of their automotive systems. The systems produced consisted of both hardware (such as processors, sensors, connectors, and housing) and software. Many of the requirements were derived from market segments, such as low cost or high cost customers or from regulatory requirements.

Based on knowledge garnered on the derivation practices within the company, we identified areas with potential for the integration of Agile methods. The output of this

[1] http://www.bosch.com

research was a technical report [17] where we documented our recommendations on the use of Agile practices within Bosch automotive business units.

The research was further developed through two research collaborations. The first was a six month visit to LASSY[2]; where AFPD and FIDJI [18] were mapped. The second was a collaboration project with Doppler Laboratory where we investigated the application of their DOPLER^{UCon} [10] tool within the AFPD

4 Agile Framework for Product Derivation

Product derivation approaches in the literature [4, 19-21] and industry practice observed through this research (c.f. Section 3), typically follow a phased structure. These phases are broadly speaking requirements analysis, product configuration and artefact reuse, and finally product specific development and testing. These phases are reflected in the structure of the AFPD. Through our research into Agile methods we have applied iterative and incremental approaches within this phased lifecycle.

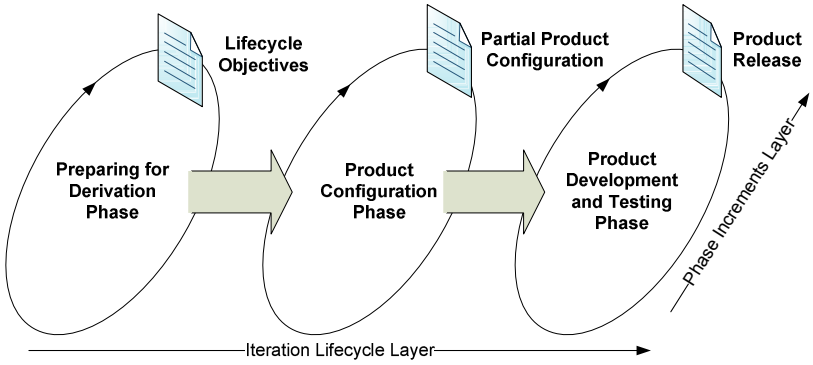

Fig. 1. Agile Framework for Product Derivation

The three principal phases, consisting of essential activities required during any product derivation project, within the AFPD are: Preparing for Derivation, Product Configuration and Product Development and Testing. Figure 1 provides an overview of these phases, including the main milestone for each phase.

Preparing for Derivation Phase determines the objectives and manages the project. The phase forms the product-specific requirements based on customer requirements and negotiation with the platform team. Requirements are prioritized and assigned to development iterations.

Product Configuration purports to create a partial product configuration based on the product-specific requirements and by using the available core assets. The aim of this phase is to maximize reuse of the platform assets.

[2] Laboratory of Advanced Software Systems (LASSY), University of Luxembourg

During *Product Development and Testing*, product specific development is undertaken. The product is tested to ensure it satisfies customer expectations.

There two major layers to the AFPD (c.f. Figure 1). These are the phase increments layer and iteration lifecycle layer. Phase increments are short units of work on a particular aspect of the derivation process i.e. configuring platform components. The iterative lifecycle layer structures these phase increments to deliver stable builds of the product that incrementally progress towards the iteration objectives. These iterations result in regular product releases.

The next section discusses expands on the Agile aspects of the AFPD.

5. Increasing Agile in Product Derivation

In this section, we discuss the following Agile elements of the AFPD:

- Adoption of Early and Continuous Delivery Strategy;
- Automation of Product Derivation;
- Product Derivation Iterations;
- Agile Testing Techniques.

We describe how these elements were identified and the benefits that they can bring to product derivation.

5.1 Adoption of Early and Continuous Delivery Strategy

Typically, implementing product specific features can be time consuming. Firstly, product construction can be substantially delayed due to the Change Control Board (CCB). The CCB scopes new development to gauge the reusability of a requested feature within the product line. Secondly, development is further delayed if the Product Team defers implementing a feature until the platform team implement the requested platform changes at the product level.

In the AFPD we adopt the Agile principle of "early and continuous delivery of valuable software". The product team implement changes at product level. The Platform Team subsequently mine any changes from the product if there is reuse potential.

In Bosch we observed this Agile principle in action. To facilitate early and continuous delivery of software, the product team would not wait for scoping decisions from the CCB. Rather, the product team would negotiate a new platform interface containing required extensions to facilitate new product components before proceeding to develop in parallel against the platform team. When the platform extensions had been implemented and the new platform was released, the product team would check for compatibility issues with newly developed components.

We recommended [17] the adoption of the Agile practice of pair programming for customer specific components. Pair programming is suitable for implementing and reviewing any changes at the product level [6]. This helps to produce better quality product code and consequently, improved code for any features that are mined for the platform.

5.2 Automation of Product Derivation

Automated support for product derivation is a necessity for managing the complexity and variability inherent in software product lines and according to Kurmann [6], automation is the most important aspect of an Agile software product line. Automated development approaches facilitate the Agile Principle "*Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.*" [12], as automated development techniques allow product teams to implement changing customer requirements late in the development lifecycle and automation enables these changes to be implemented quickly.

However current process models and tools for automation do not integrate well. All the stakeholders involved in product derivation are supported in their tasks by different approaches and different automation tools. Because of the difficulty of integrating these different approaches and tools, product derivation can quickly become an error-prone and tedious task.

In our research collaboration with Dopler Laboratory (c.f. Section 3) we investigated how DOPLER^{UCon} [10] tool could be used within the AFPD. We were particularly interested in its ability to facilitate Agile approaches. For instance, we observed that while the DOPLER^{UCon} tool does not directly support *iterative development* cycles by defining additional attributes for requirements it could be used to allocate specific requirements to specific iterations.

5.3 Product Derivation Iterations

The identification of product derivation iterations is a key aspect of deriving high quality, customer satisfying products. According to Carbon et al. [2] when adopting a SPL approach, an organisation is capable of producing a first version of a product for a specific customer, including the core functionality, quicker than other software development methods. Because of the approved quality of the reusable assets, the customer can get a high quality product that can be used and evaluated to give feedback. In further iterations, new functionality can be added to the scope of the product line or product specific features can be implemented [2].

In a technical report to Bosch [17], we recommended that they could benefit from applying the planning game practice from the XP methodology for the management of their product iterations during the *Preparing for Derivation* phase. This would assist them in gathering and negotiating product specific requirements. During customer negotiation requirements are prioritised and allocated to specific iterations based on priority.

5.4 Agile Testing Techniques

Agile methods propose that testing is carried out frequently, as this helps Agile developers keep their code as error free as possible. We have adopted a phased testing approach in the AFPD. Based on the principles of integration testing suggested by Muccini [22], the structure and nature of the elements in a product line are

leveraged. Firstly, integrate the partial configuration and use a traditional approach to integration testing. Then, based on the observation that at least the partial product configuration works properly, we can incorporate the other product elements. Product construction continues in a phased assembly test approach. For systems testing of partial or fully assembled products traditional system testing techniques can be utilized as no SPL specific methods exist.

6. Conclusion and Future Work

Our research is motivated by the fact that despite the widespread adoption of SPL within industry, product derivation remains an expensive and error-prone activity [23, 24]. We believe that the adoption of Agile practices can improve the product derivation process. The Agile Framework for Product Derivation provides a means of supporting this adoption.

The development of the framework is a response to calls from industry for research into this area [25]. The integrated Agile framework could solve many of the problems associated with product derivation's complex and cumbersome nature.

The framework is a lightweight approach to product derivation, minimising the amount of up-front investment required making SPL more accessible to small organisations with limited resources. The framework may benefit larger organisations by bringing a balance between formalism and agility, helping individual product teams deliver products with the best possible quality. A combination of Agile and SPL is expected to create a leaner but more disciplined product derivation process [6].

Our future work includes an ongoing investigation into the benefits of combining Agile and SPL approaches and the validation of our framework, particularly with respect to the expected return on investment.

7. Acknowledgements

This work is partially funded by IRCSET under grant no. RS/06/167 and by Science Foundation Ireland under grant no.s 03/CE2/I303_1 and 07/SK/I1299.

References

1. Tian, K., Cooper, K.: Agile and Software Product Line Methods: Are They So Different? : 1st International Workshop on Agile Product Line Engineering, Maryland, USA (2006)
2. Carbon, R., Lindvall, M., Muthig, D., Costa, P.: Integrating Product Line Engineering and Agile Methods: Flexible Design Up-front vs. Incremental Design. 1st International Workshop on Agile Product Line Engineering (APLE'06), Maryland, USA (2006)
3. Navarrete, F., Botella, P., Xavier, F.: An Approach to Reconcile the Agile and CMMI Context in Product Line Development. 1st International Workshop on Agile Product Line Engineering (APLE'06) Maryland, USA (2006)

4. Deelstra, S., Sinnema, M., Bosch, J.: Product Derivation in Software Product Families: A Case Study. J. Syst. Softw., Vol. 74. Elsevier Science Inc., New York, NY, USA (2005) 173-194
5. Griss, M.L.: Implementing Product-Line Features with Component Reuse. Springer-Verlag, London, UK (2000)
6. Kurmann, R.: Agile SPL-SCM Agile Software Product Line Configuration and Release Management. 1st International Workshop on Agile Product Line Engineering (APLE'06). Maryland, USA (2006)
7. Noor, M., Rabiser, R., Grünbacher, P.: A Collaborative Approach for Reengineering-based Product Line Scoping. 1st International Workshop on Agile Product Line Engineering (APLE'06), Maryland, USA (2006)
8. Trinidad, P., Benavides, D., Ruiz-Cortes, A., Segura, S.: Explanations for Agile Feature Models. 1st International Workshop on Agile Product Line Engineering (APLE'06), Maryland, USA (2006)
9. Hotz, L., Gunter, A., Krebs, T.: A Knowledge-based Product Derivation Process and some Ideas how to Integrate Product Development. Proc. of Software Variability Management Workshop, Groningen, The Netherlands (2003)
10. Rabiser, R., Grünbacher, P., Dhungana, D.: Supporting Product Derivation by Adapting and Augmenting Variability Models. 11th International SPLC, Kyoto, Japan (2007)
11. Abrahamsson, P., Warsta, J., Siponen, M.T., Ronkainen, J.: New Directions on Agile Methods: A Comparative Analysis. ICSE 2003, Portland, USA (2003)
12. Manifesto for Agile Software Development. http://agilemanifesto.org/
13. Beck, K., Andres, C.: Extreme Programming Explained : Embrace Change (2nd Edition). Addison-Wesley Professional (2004)
14. Schwaber, K., Beedle, M.: Agile Software Development with Scrum. Prentice Hall PTR (2001)
15. Beck, K.: Embracing Change with Extreme Programming. IEEE Computer Society Press, Vol. 32 (1999) 70 - 77
16. Beck, K.: Extreme Programming Explained: Embrace Change. Addison-Wesley Longman Publishing Co., Inc, Boston, MA, USA (1999)
17. O'Leary, P., Thiel, S., Botterweck, G., Richardson, I.: Bosch AB10 Technical Report. LERO The Irish Software Engineering Research Centre (2008) 26
18. Perrouin, G., Klein, J., Guelfi, N., Jezequel, J.M.: Reconciling Automation and Flexibility in Product Derivation. Software Product Line Conference, 2008. SPLC '08. 12th International (2008) 339-348
19. Bayer, J., Gacek, C., Muthig, D., Widen, T.: PuLSE-I: Deriving instances from a product line infrastructure. Engineering of Computer Based Systems, 2000. (ECBS 2000) Proceedings. Seventh IEEE International Conference and Workshopon the (2000) 237-245
20. Czarnecki, K., Helson, S., Eisenecker, U.W.: Staged configuration using feature models. Proc. of the 3rd International Software Product Line Conference (SPLC 2004). Springer Berlin Heidelberg, Boston, MA, USA (2004) 266-283
21. Halmans, G., Pohl, K.: Communicating the Variability of a Software-Product Family to Customers. Informatik - Forschung und Entwicklung **18** (2003) 113-131
22. Muccini, H., van der Hoek, A.: Towards Testing Product Line Architectures. ETAPS2003. In Electronic Notes of Theoretical Computer Science, Warsaw, Poland (2003)
23. Deelstra, S., Sinnema, M., Bosch, J.: Experiences in Software Product Families: Problems and Issues During Product Derivation. Software Product Lines, Third International Conference. Springer, Boston, MA, USA (2004)
24. Hotz, L., Wolter, K., Krebs, T.: Configuration in Industrial Product Families: The ConIPF Methodology. IOS Press, Inc. (2006)
25. BOSCH: Presentation: Product Derivation - Research Interests within BOSCH. In: LERO (ed.): (2006)