An Agile Process Model for Product Derivation in Software Product Line Engineering

Pádraig O'Leary[1], Fergal McCaffery[2&1], Steffen Thiel[3], Ita Richardson[1]

[1]Lero, the Irish Software Engineering Research Centre, University of Limerick, Ireland
[2]Dundalk Institute of Technology, Dundalk, Ireland
[3]Department of Computer Science, Furtwangen University of Applied Sciences, Germany

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

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 process model 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 Process Model for Product Derivation (A-Pro-PD). A-Pro-PD was developed as part of Pro-PD (Process reference model for Product Derivation). Pro-PD was developed at Lero (the Irish Software Engineering Research Centre) with the goal of defining a generic process reference model for product derivation. Pro-PD can be used as a foundation for situation-specific process approaches to product derivation. Pro-PD, its development, and its validation are also described in [5]. As part of the development of Pro-PD, we adapted it through Use-by-Specialisation [6] to incorporate Agile principles. This adapted version is called A-Pro-PD.

We decided to concentrate on product derivation as it is considered one of the most important and challenging SPL "activities" [7], 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, 8-10], much of this has been related to identifying the opportunities and challenges. There has been little research conducted on the use of Agile approaches in the product derivation process while no Agile approach for product derivation have been developed. This is a first attempt at bridging this gap.

A-Pro-PD is a lightweight approach to product derivation, minimising the amount of up-front investment required and therefore making SPL more accessible to small organisations with limited resources. For larger organisations, A-Pro-PD could bring 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 [8].

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 A-Pro-PD. In Section 5, we discuss in detail the Agile aspects of the A-Pro-PD. 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 [8]. 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 [10]. 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 [4, 11-13]. Hotz *et al.* [13] 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" [12].

2.2 Agile Practices

Agile Methods 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. In the 2000s, interest in agile methods increased dramatically[14]. Agile methods have been adopted in different types of software projects and in various application domains [15]. Researchers have shown that the use of agile methods can assist product manageability, visibility and team communication [16] as well as ensuring frequent feedback from the customer [17]. Researchers and practitioners have proposed several software development approaches based on the principles of the Agile manifesto [18]. The Agile manifesto defines four basic core values:

- Individuals and interactions over processes and tools;
- Working software over comprehensive documentation;
- Customer collaboration over contract negotiation;
- Responding to change over following a plan.

In addition to these four values, the Agile manifesto defines a set of principles that form guidelines for development. The Agile manifesto provides a foundation for all of the Agile methods including: eXtreme Programming (XP) [19] and Scrum [20].

XP evolved from the problems caused by the long development cycles of traditional development models [21].

The individual practices of XP are not new however those 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 [22]. In some instances selected sets of XP practices have even been used in the field

of safety critical systems [23]. In such cases, the use of XP practices has reportedly improved quality by 53% compared to the plan-driven software development project.

Scrum provides a project management framework that focuses development into 30-day Sprint cycles in which a specified set of Backlog features are delivered [20]. 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. One of the central principles of the Agile approach is the focus on people. People, coupled with effectiveness and manoeuvrability, are considered the primary drivers of a project's success [24].

2.3 Agile Practices and SPL

Recently there has been growing interest in exploring the possibilities of integrating Agile approaches into the SPL development process. In conjunction with the 2006 Software Product Line Conference (SPLC) a new workshop was arranged called Agile Product Line Engineering (APLE). From the workshop it was recognized that both Agile approaches and SPL share several common goals [1, 8, 10] such as:

- Increasing the productivity of teams;
- Reducing the product's time to market;
- Reducing the development costs;
- Improving customer satisfaction.

Both approaches assume that requirement changes during the development will occur and Agile based testing could benefit SPL development [1]. Furthermore both development paradigms are being promoted as a means of reducing time to market, increasing productivity, and gaining cost effectiveness and efficiency of software development efforts [1]. These goals (shared by Agile and SPL) open the possibilities of introducing Agile practices into SPL activities.

While a motivation for combining Agile and SPL approaches can be found there is also recognition of the potential challenges of any integration effort due to certain differences that exist in the philosophies of both approaches such as design and change management strategies [1, 2].

Both approaches have different design strategies. Agile focuses on a simple design while SPL has an emphasis on rigorously designing and systematically maintaining SPL architectures [2]. Agile approaches do not develop flexible artefacts for reuse [2, 3]. SPL tends toward long-lived life cycles which means that the maintaining of software is necessary [3]. SPL targets satisfying the needs of several customers rather than satisfying individual needs [3]. SPL and Agile approaches include different strategies for change management: in Agile it is incremental development and in SPL the focus is on platform artefacts [1]. SPL architecture needs to be flexible to handle requirements of several customers [25], Agile does not.

Despite having similar goals, both Agile and SPL have different mechanisms and logistics for achieving their respective goals. SPL aims to fulfil the requirements of several customers at the same time rather than concentrating on meeting the individual needs of a particular customer [2]. Customer involvement is much more intensive in Agile approaches where developers work closely with customers on a daily basis. Agile and SPL paradigms also differ in terms of role and importance of software design. Agile approaches do not emphasise the importance of rigorous design and documentation; rather Agile approaches advocate implementing required functionality and then documenting the prepared code by reconstruction and refactoring. On the other hand, design and documentation are very vital activities in SPL as platform assets need to be appropriately documented to support their reusability [1].

As a result of these challenges, there are a number of risks associated with an Agile SPL approach. If an SPL based architecture is tailored to be more Agile there is a danger that valuable architecture that supports other products in the family may be damaged [1]. Traceability management as well as maintaining of components, in Agile approaches can be difficult without explicit knowledge [10] and no tool support for Agile SPL approach exists [2].

Despite the documented challenges and risks mentioned, there are some suggested approaches for integrating Agile practices in SPL. Some suggestions include the Planning Game [3] from the XP methodology can increase agility in SPL organisations, particularly for gathering and negotiating the product requirements. The use of Agile practices during application engineering activities is suggested [2], in particular to perform product tailoring [9] and to support the collaborative working of stakeholders [1].

An Agile SPL approach may have the potential of supporting the effective and efficient development of much larger families of products than a traditional SPL approach. Moreover, the Agile project management approach Scrum can be used to manage the product derivation process and development practices like XP or test-driven development can be used for developing platform and/or product components.

3. Research Approach

A-Pro-PD was developed, as part of Pro-PD, with the goal of defining a generic Agile process model for product derivation. 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. The output of this four month iterative development stage was version one of Pro-PD [26].

We extended version one through case study research with Robert Bosch GmbH[1]. Here we investigated product derivation practices within an automotive systems sub-

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

unit. The systems produced consist of both hardware (such as processors, sensors, connectors, and housing) and software. Data collection involved studying internal company documentation, an onsite visit to their headquarters and a two-day workshop with key employees. 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 research was a technical report [27] where we documented our recommendations on the use of Agile practices within Bosch automotive business units. By generalising and discussing our observations we revised version one and developed version two of Pro-PD [28].

The research was further developed through two research collaborations. The first was a six month visit to LASSY2; where Pro-PD and FIDJI [29] were mapped. The second was a collaboration project with Doppler Laboratory where we investigated the application of their DOPLER^{UCon} [12] tool within the Pro-PD. The output of this phase of the research was [30].

We performed further validation of Pro-PD through an inter-model evaluation with the SEI Product Line Practice Framework [31].

We used the Eclipse Process Framework (EPF) [The Eclipse Foundation, 32] to model A-Pro-PD. By enabling inbuilt process variability within EPF we can select, tailor, or remove content from our process in order to strike the right balance for a particular situation. Moreover, a documented process model is a good starting point for the integration of non-standard techniques such as agile practices, at appropriate times in the development process [26, 33, 34].

4 A-Pro-PD

A-Pro-PD is intended to offer process guidance to small, co-located teams and support the adoption of Agile practices in product derivation. The instantiation of A-Pro-PD is intended to:

- Encourage the use of iterative development cycles in product derivation
- Minimise the risk associated with 'big bang' releases where large products are assembled just before product delivery
- Increase customer involvement in the derivation process.

Product derivation approaches in the literature [27] and industry practice observed through this research (c.f. Section 3), 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 A-Pro-PD. Through our research into Agile methods we have applied iterative and incremental approaches within this phased lifecycle.

There are two layers to A-Pro-PD (see Fig. 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 e.g. configuring platform components. The iterative lifecycle layer structures these phase increments to deliver stable builds of

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

the product that incrementally progress towards the iteration objectives. These iterations result in regular product releases.

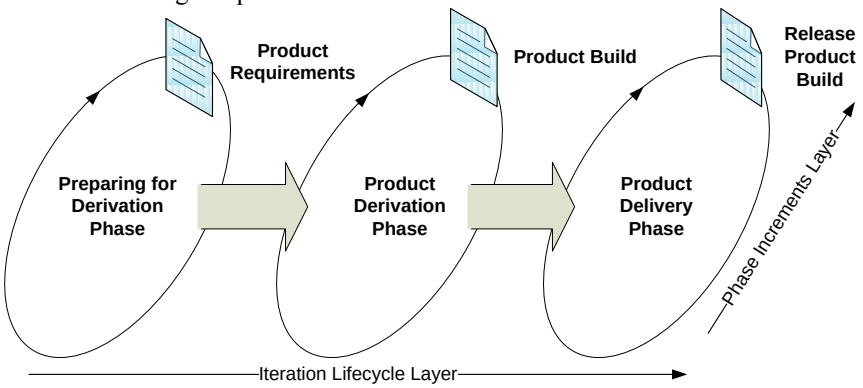

Fig. 1. Agile Process Model for Product Derivation

The three principal phases, consisting of essential activities required during any product derivation project, within the A-Pro-PD are:

- Preparing for Derivation
- Product Configuration
- Product Development and Testing.

4.1 Preparing for Derivation

Preparing for Derivation phase determines the product requirements and performs the project management tasks for a specific iteration. 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. The purpose of this phase is to achieve agreement among all the stakeholders on the *Product Requirements*. On subsequent iterations, this phase is used to plan and manage product iterations. The main objectives of this phase are as follows:

- Define the product requirements
- Scope the Product
- Define guidance for decision makers
- Allocate requirements for implementation in a specific iteration
- Plan the project
- Identify role and task structures

4.2 Product Configuration

The goal of the *Product Configuration* phase is to create a partial product configuration that maximises the benefits of the platform artefacts and minimises the amount of product specific development required. The partial product configuration is derived based on the product specific requirements and by using the available assets.

Theoretically at this stage the partial product configuration could satisfy the product requirements for a specific iteration and testing should begin. However, this is the ideal case and assumes all the product requirements are covered by the platform. In most cases some additional development will be required. This additional development occurs in the next phase.

The main objectives of this phase are as follows:

- Create the Product Build
- Make maximum use of the platform artefacts

4.3 Product Development and Testing

This phase focuses on satisfying requirements which could not be satisfied through reuse of platform assets through product specific development and transitioning the software into the customers environment. Transitioning the software involves achieving customer agreement that the product is complete. The purpose of this phase is to ensure the product is ready for delivery to users.

The number of iterations in the phase varies from one iteration (for a simple system requiring primarily minor bug fixing) to many iterations (for a complex system involving adding features and performing activities) to make the business transition from using the old system to using the new system.

When the *Product Requirements* have been met, the project is in a position to be closed. For some products, the end of the current project lifecycle may coincide with the beginning of the next lifecycle, leading to the next generation of the same product. The main objectives of this phase are as follows:

- Test to product to ensure it meets the product requirements
- Correct defects
- Modify the software if unforeseen problems arise
- Provide feedback to platform team on usage of platform

4.4 Phase Milestones

Phase milestones are used as gateways for phase endings. If a milestone is reached the project is ready to move to the next phase. Each iteration of the project provides an increment in functionality or design.

Each phase has a specific focus and objective. At the end of the *Preparing for Derivation* phase is the first milestone, the **Product Requirements Milestone**. At this point the product requirements have been defined and the allocated for the iteration.

At the end of the Product Configuration phase is the **Product Build Milestone**. At this point, a *Product Build* is constructed.

At the end of the *Product Development and Testing* Phase is the **Product Release Milestone**. At this point, a decision is made on whether the Product Requirements allocated to this iteration have been met.

5. Increasing Agile in Product Derivation

In this section, we discuss the following Agile elements of the A-Pro-PD:

- 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 A-Pro-PD 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 [8] 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 [8]. 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 [18], 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} [2] tool could be used within the A-Pro-PD. 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 [27].

In a technical report to Bosch [35], 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 A-Pro-PD. Based on the principles of integration testing suggested by Muccini [13], 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

In this paper, the results of research into the adoption of Agile practices in product derivation was presented. The research into Agile practices in product derivation is motivated by the fact that despite the widespread adoption of SPL within industry, product derivation remains an expensive and error-prone activity [36]. The adoption of Agile practices could improve the product derivation process. A-Pro-PD provides a means of supporting this adoption.

The development of the Agile approach to product derivation is a response to calls from industry for research into this area [8]. The integrated Agile process model could solve many of the problems associated with product derivation's complex and cumbersome nature.

A-Pro-PD 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 A-Pro-PD 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 .

Our future work includes an ongoing investigation into the benefits of combining Agile and SPL approaches and the validation of A-Pro-PD, particularly with respect to the expected return on investment. Another area of future work is the combination of Lean Software Development [37] and SPL. As Lean Software Development is based on production methods from product lines at Toyota[3], the integration of lean principles in SPL could have interesting results.

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

[3] http://www.toyota.com/

4. Deelstra, S., Sinnema, M., and Bosch, J., Product Derivation in Software Product Families: A Case Study. *Journal of Systems and Software*, 2005. **74**(2): p. 173-194.
5. O'Leary, P., *Towards a Product Derivation Process Reference Model for Software Product Line Organisations*, in *Department of Computer Science and Information Systems*. 2009, University of Limerick: Limerick. p. 241.
6. Braun, R., Esswein, W., Gehlert, A., and Weller, J., *Coniguration Management for Reference Models*, in *Reference Modeling for Business Systems Analysis*, P. Fettke and P. Loos, Editors. 2006, IGI. p. 331-357.
7. Griss, M.L., *Implementing Product-Line Features with Component Reuse*. ICSR-6: Proceedings of the 6th International Conerence on Software Reuse. 2000, London, UK: Springer-Verlag. 137--152.
8. Kurmann, R. Agile SPL-SCM Agile Software Product Line Configuration and Release Management. in *1st International Workshop on Agile Product Line Engineering (APLE'06)*. 2006. Maryland, USA.
9. Noor, M., Rabiser, R., and Grünbacher, P., *A Collaborative Approach for Reengineering-based Product Line Scoping*, in *1st International Workshop on Agile Product Line Engineering (APLE'06)*. 2006: Maryland, USA.
10. Trinidad, P., Benavides, D., Ruiz-Cortes, A., and Segura, S. Explanations for Agile Feature Models. in *1st International Workshop on Agile Product Line Engineering (APLE'06)*. 2006. Maryland, USA.
11. O'Leary, P., Richardson, I., McCaffery, F., and Thiel, S. Preparing for Product Derivation: Activities and Issues. in *4th International Conference on Software and Data Technologies - ICSOFT 2009*. 2009. Sofia, Bulgaria: INSTICC - Institute for Systems and Technologies of Information, Control and Communication.
12. Rabiser, R., Grünbacher, P., and Dhungana, D., *Supporting Product Derivation by Adapting and Augmenting Variability Models*, in *11th International Software Product Line Conference*. 2007: Kyoto, Japan.
13. Hotz, L., Gunter, A., and Krebs, T., *A Knowledge-based Product Derivation Process and some Ideas how to Integrate Product Development*, in *Proc. of Software Variability Management Workshop*. 2003: Groningen, The Netherlands.
14. Mikael, L., Muthig, D., Dagnino , A., Wallin, C., Stupperich, M., Kiefer, D., May, J., and Kahkonen, T., Agile Software Development in Large Organizations. *Computing Practices*, 2004. **37**(12): p. 38-46.
15. Karlström, D., *Introducing Extreme Programming - An Experience Report*, in *3rd International Conference on eXtreme Programming and Agile Processes in Software Engineering (XP2002)*. 2002: Sardinia, Italy.
16. Larman, C., *Agile and Iterative Development*. 2003, Boston: Addison-Wesley Professional. 342.
17. Rising, L. and Janoff, N.S., The Scrum Software Development Process for Small Teams. *IEEE Software*, 2000. **17**(4): p. 26-32.
18. Beck, K., Beedle, M., Bennekum, A.v., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R.C., Mellor, S., Schwaber, K., Sutherland, J., and Thomas, D. *Manifesto for Agile Software Development*. 2001 1/3/2006 [cited 10/09/2008]; Available from: http://agilemanifesto.org/.
19. Beck, K. and Andres, C., *Extreme Programming Explained : Embrace Change (2nd Edition)*. 2004: Addison-Wesley Professional.
20. Schwaber, K. and Beedle, M., *Agile Software Development with Scrum*. 2001: Prentice Hall PTR.
21. Beck, K., *Embracing Change with Extreme Programming*, in *IEEE Computer*. 1999. p. 70 - 77.

22. Beck, K., *Extreme Programming Explained: Embrace Change*. 1999, Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc. 190.
23. Grenning, J., *Using XP in a Big Process Company: A Report from the Field*, in *XP Universe*. 2002: Raleigh, NC.
24. Highsmith, J. and Cockburn, A., *Agile Software Development: The Business of Innovation*, in *IEEE Computer*. 2001. p. 120-122.
25. Clements, P. and Northrop, L., *Software Product Lines: Practices and Patterns*. 2001, Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.
26. O'Leary, P., Ali Babar, M., Thiel, S., and Richardson, I. Product Derivation Process and Agile Approaches: Exploring the Integration Potential. in *Proceedings of 2nd IFIP Central and East European Conference on Software Engineering Techniques*. 2007. Poznań, Poland: Wydawnictwo NAKOM.
27. O'Leary, P., Thiel, S., Botterweck, G., and Richardson, I. (2008). *Bosch Technical Report*: LERO - The Irish Software Engineering Research Centre.
28. O'Leary, P., Thiel, S., Botterweck, G., and Richardson, I. Towards a Product Derivation Process Framework. in *3rd IFIP TC2 Central and East European Conference on Software Engineering Techniques CEE-SET 2008*. 2008. Brno (Czech Republic).
29. Perrouin, G., Klein, J., Guelfi, N., and Jezequel, J.M. Reconciling Automation and Flexibility in Product Derivation. in *12th International Software Product Line Conference (SPLC)*. 2008.
30. O'Leary, P., Rabiser, R., Richardson, I., and Thiel, S. Important Issues and Key Activities in Product Derivation: Experiences from Two Independent Research Projects. in *Software Product Line Conference*. 2009. San Francisco, CA, USA: Proc. of the 13th International Software Product Line Conference (SPLC 2009).
31. Northrop, L. and Clements, P. *A Framework for Software Product Line Practice, version 5.0*. 2007 30th April 2008 [cited 12/09/2008]; Available from: http://www.sei.cmu.edu/productlines/framework.html.
32. *Eclipse Process Framework Composer (EPF)*. 2007, IBM.
33. O'Leary, P., Ali Babar, M., Thiel, S., and Richardson, I. Towards Agile Product Derivation in Software Product Line Engineering. in *4th International Workshop on Rapid Integration of Software Engineering techniques (RISE)*. 2007. Luxembourg.
34. O'Leary, P., McCaffery, F., Richardson, I., and Thiel, S. Towards Agile Product Derivaiton in Software Product Line Engineering. in *16th European Conference on Software Process Improvement (EuroSPI 2009)*. 2009. Madrid, Spain.
35. Muccini, H. and van der Hoek, A., *Towards Testing Product Line Architectures*, in *European Joint Conferences on Theory and Practice of Software (ETAPS)*. 2003, In Electronic Notes of Theoretical Computer Science: Warsaw, Poland.
36. Bosch (2006). *Presentation: Product Derivation - Research Interests within BOSCH*, Lero,Lero, University of Limerick,Dec, 2006.
37. Poppendieck, M. and Poppendieck, T., *Implementing Lean Software Development: From Concept to Cash*. 1st ed. The Addison-Wesley Signature Series. 2006: Addison-Wesley Professional. 304.