Preparing for Product Derivation: Activities and Issues

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

[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) advocates the development of applications by reusing shared software assets across a set of related products. Current approaches to the derivation of products from a product line focuses on handling the commonalities and variabilities of the shared software assets. These approaches have failed to consider the early phases of product derivation. In this paper we report on how we compared both industrial and academic approaches to the establishment of a product derivation project. Based on this research and our experiences, we have identified key activities and important issues that should be considered when establishing a product derivation project.

Keywords: Software Product Lines, Product Derivation, Process Engineering.

1. Introduction

1.1 Software Product Lines

"A Software Product Line (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" [1]. 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 [2]. The separation of SPL into domain engineering and application engineering allows the development of software artefacts which are shared among all the products within that domain. These shared artefacts become separate entities in their own right, subscribing to providing shared functionality across multiple products.

During application engineering, individual products are constructed from the product line to fulfil the requirements of a particular customer or market. The products are built (re-)using a number of shared software artefacts – often called core assets – created during domain engineering. The process of creating these individual products using the platform artefacts is known as product derivation.

1.2 Product Derivation

Product Derivation is the process of constructing a product from a Software Product Line (SPL) [3]. 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" [3]. 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.* [4] describe the process as "slow and error prone even if no new development is involved". Griss [5] identifies the inherent complexity and the coordination required in the derivation process by stating that "…as a product is defined by selecting a group of features, a carefully coordinated and complicated mixture of parts of different components are involved". Therefore, as Deelstra *et al.* [3] point out: 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" [6]. As a means of addressing this imbalance, we are investigating the practices and issues surrounding the initial stage of the product derivation process, a stage we refer to as pre-derivation.

1.3 Contribution

Comparing existing product derivation approaches that consider pre-derivation allows the definition of important issues to be addressed and key activities that should be supported. The observations, which are reported in this paper, should be of interest to both researchers and industry practitioners alike.

The remainder of this paper is organised as follows: Section 2 discusses related work. Section 3, describes our research approach. In Section 4, based on our experiences we define key activities for product derivation preparation. In Section 5 we present important issues to be considered when initiating a product derivation project. We conclude the paper with a summary and an outlook on future work in Section 6.

2. Background

Several approaches with pre-derivation facets have been proposed. Deelstra *et al.* [3] present a product derivation approach developed based on two industrial case studies. This work presents a framework of terminology and concepts for product derivation. The framework focuses on product configuration and is a high level attempt at providing the methodological support that Deelstra *et al.* [7] agree is required for product derivation. Deelstra's approach suggests that requirements which cannot be accommodated by existing assets are handled by product-specific adaptation or reactive evolution. Parts of the derivation framework have been implemented in a research tool called COVAMOF [8], a variability modelling framework which purports to solve the product derivation problems associated with dependencies.

McGregor [9] introduces the *production plan*, which prescribes how products are produced from platform assets. The product plan facilitates the passing of knowledge between the platform developers and the product developers. McGregor [10] also provides an overview of technologies and approaches to automate product derivation.

Rabiser *et al.* [6] present an approach for supporting product derivation using feature specifications. The approach emphasises supporting the requirements acquisition and management mechanism through the use of variability models.

However, despite the above approaches, comparably few publications focus on the early stages of product derivation such as requirements management and project initiation. Clements and Northrop [11] describe the role of requirements engineering when deriving a product. Halmans and Pohl [12, 13] describe a use-case-driven method to communicate the variability to the customers and to capture requirements.

These different approaches have been developed with different goals, for different purposes, and in different domains. Some are intended to provide a (process) framework for product derivation [3, 14, 15], and others focus on tool-support [8]. Our research into pre-derivation has been influenced by these existing approaches. The key activities and important issues we derive in Section 4 and 5 therefore also partly reflect this previous work.

3. Research Approach

The preparatory stage of this research involved reviewing existing SPL whitepapers, product derivation papers and software process improvement (SPI) practices. The research aimed to identify the fundamental practices of pre-derivation, including available empirical evidence on the topic – scientific as well as anecdotal. The initial results were further developed and assessed through a series of iterative workshops over a four month period. Evidence and feedback from SPL practitioners and researchers was collected from these organised workshops.

For the case study, we collected data on the product derivation practices of a major supplier of automotive systems. The systems produced consist of both hardware (such as processors, sensors, connectors, and housing) and software. Prior to an on-site visit of the case study company, we had access to internal company documentation. These documents included information on product derivation practices within a particular business unit, organisational structure of the company's teams and information on various derivation techniques applied within the company.

For the onsite visit to the company, we organised a two day workshop. During the workshop we presented our preliminary findings on the company's derivation practices and used these initial findings to drive the workshop discussion. In total three researchers facilitated the running of the workshop.

Our research was further developed through a six month visit to LASSY lab[1]; where our model of product derivation activities and FIDJI [16] were mapped. FIDJI is a flexible product derivation process which forms part of a model-driven SPL development methodology. Mapping our research to FIDJI provided academic validation.

We conducted a collaboration project with Doppler Laboratory[2] where we investigated the application of their DOPLER^{UCon} [6] approach to product derivation which was developed in conjunction with Siemens VAI. We investigated the issues and activities observed within Siemens VAI and our research to date. This paper builds on the results from that collaboration [17].

4. Pre-Derivation: Key Activities

From our research, we have identified that the following preparatory steps need to be conducted in a product derivation project:

- Requirements Management;
- Identify Starting Point for Derivation;
- Map Customer Requirements to Platform Features;
- Customer Negotiation;
- Create Product Specific Requirements;
- Identify Role and Task Structures;
- Plan the Project;
- Prepare Guidance for Decision Makers.

4.1 Requirements Management. We identified the need for a more sophisticated requirements management process when dealing with large distributed SPL teams, particularly within the case study company. *Customer requirements* are *translated* into the internal organizational language. This prevents terminology confusion and customer-specific description of assets. This has to be done in close collaboration with the customer. These requirements are processed and augmented through various tasks where requirements are analysed for reuse potential and then assigned to responsible disciplines.

4.2 Identify Starting Point for Derivation. A "base configuration" may be chosen as a *starting point for derivation*, i.e., from a set of previous product configurations. Similar customers often have comparable requirements and experiences from past

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

[2] Christian Doppler Lab. for Automated SW Eng., Johannes Kepler University Linz, Austria

projects are captured in these product configurations. Reusing previous product configurations can speed up the derivation process. If an existing product configuration can not be used for the "base configuration", a new one is derived from a subset of the overall platform architecture.

4.3 Customer Negotiation. *Customer requirements* are mapped to the base configuration. Requirements which cannot be satisfied by existing assets have to be negotiated with the customer. Effort estimation issues can make customer negotiation difficult. The trade-off here is to meet as many of the customer's needs as possible while retaining the profitability of the platform assets for the whole product line.

In the case study we observed how, through coverage analysis, the project manager identifies which requirements are covered by the platform. If specific requirements cannot be completely satisfied, they are broken into smaller requirements and then mapped to specific components.

4.4 Create the Product Specific Requirements. The satisfied customer requirements and the negotiated customer requirements are merged to form the product specific requirements. This could also include the restructuring of the customer requirements specification into the internal organisation format.

We observed how forming the Product Specific Requirements can also include allocating requirements to relevant disciplines. The requirements allocation is often held in separate requirements documents, such as the platform software requirements specification and the customer hardware requirements specification.

4.5 Identify Role and Task Structures. The role and task structures for the product derivation project have to be defined. Through allocating role and task structures, responsibility for resolving any remaining variability in product derivation to fulfill the product requirements is defined. This is very important as it provides different views on variability for different people involved in product derivation and helps to lower the complexity of large decision spaces (c.f. Section 5.2).

4.6 Plan the Project. We observed two types of project planning. Manual non-tool supported product derivation projects tended to have *'big bang'* releases after substantial development periods. Automated approaches appeared to be more iterative in nature, as each new version of the product required less effort then the manual approaches.

4.7 Prepare Guidance for Decision Makers. Preparing for derivation also means to create *guidance for decision-makers*. Remaining variability must be explained to deal with complexity issues in representing product line variability. Guidance is essential, especially for sales people, who are confronted with many – often technical – decisions [18].

5 Pre-Derivation Issues

Pre-derivation issues were also identified during the course of this this research:

- Customer Relationship;
- Mapping customer requirements to platform features is often (too) complex;
- Use of Documentation;
- Introduction of Iterative Development.

5.1 Customer Relationship. Customer involvement in product derivation is typically portrayed as a combative relationship involving negotiation between separate parties with contrasting motivations. This is in contrast to customer relationship approaches we have observed.

The customer can play a very active and positive role in the derivation process. It can be a collaborative role, where the customer makes design decisions alongside the derivation team. Good communication where the limitations and opportunities provided by the product line feature set are clearly explained, can nurture a collaborative relationship with the customer.

5.2 Mapping customer requirements to platform features is often (too) complex. Poor requirements elicitation practices can lead to poorly specified requirements. The specification of incompatible customer requirements and undocumented dependencies can be costly at a later stage in the product derivation process. The size and complexity of variability models for large-scale product lines exasperates the issue, as difficulties in communicating the variability provided by the product line may lead to unrealistic customer requirements.

In industrial contexts, where there are hundreds or even thousands of requirements, the cognitive complexity makes mapping customer requirements to platform features difficult. As a result, situations can develop where the product team cannot distinguish between requirements which are mapped or not. To compensate, product teams perform extensive verification which is expensive and time-consuming.

5.3 Use of Documentation. Different organizations have different attitudes towards documentation. Organizations with a documentation culture tend to use it in response to other problems. For instance, in communicating information across large distributed teams, such organizations tend to be overly-reliant on documentation. An organization's documentation often becomes bloated as teams attempt to capture too much. Such overly detailed documentation decreases traceability of relevant information and results in failure to correctly identify artefacts for reuse especially in team sizes where the transfer of tacit knowledge is prohibitive.

Alternatively, organizations may rely on tacit knowledge and do not have practices of knowledge externalization. For instance, during product assembly, product teams often remark that the selected components are incompatible. This is due to the fact that all compatibility aspects between these components are not externalized.

5.4 Project Planning: Iterative Development. The identification of product derivation iterations is a key aspect of deriving high quality, customer satisfying products. According to Carbon et al. [19] with a SPL, 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

During the course of this research we observed that for iteration management, product teams could benefit from applying the planning game practice from the XP methodology for gathering and negotiating product specific requirements. In the planning game, a customer priorities the requirements and the developers estimate the effort required to satisfy those requirements. The end dates of iterations are specified and requirements are allocated to specific iterations based on their priority [19].

6. Conclusions and Future Work

In this paper, we have presented the results of our research into the early stages of product derivation. We compared both industrial and academic approaches to the establishment of a product derivation project. For academia, our results provide structure to an important phase of product derivation. Our work points to areas of uncertainty and helps to identify remaining challenges in preparing for derivation. Such a roadmap encourages the insertion of those pieces that may be missing, or the extra detail that may be needed.

For industry, it is envisaged that our results will help the advancement of product derivation practices. It will assist organisations by specifying the activities to be supported when initiating product derivation and highlight key issues to be considered.

In future work, we plan to continue case study research for further elaboration on activities and issues to be considered. Based on these results, we will define a framework of activities for pre-derivation.

7. Acknowledgements

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

References

1. 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)
2. 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)

3. 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
4. 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)
5. Griss, M.L.: Implementing Product-Line Features with Component Reuse. Springer-Verlag, London, UK (2000)
6. Rabiser, R., Grünbacher, P., Dhungana, D.: Supporting Product Derivation by Adapting and Augmenting Variability Models. 11th International Software Product Line Conference, Kyoto, Japan (2007)
7. 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)
8. Sinnema, M., Deelstra, S., Nijhuis, J., Bosch, J.: Modeling Dependencies in Product Families with COVAMOF. 13th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems (ECBS 2006), Potsdam, Germany (2006)
9. Chastek, G., McGregor, J.D.: Guidelines for Developing a Product Line Production Plan. Product Line Practice Initiative. Software Engineering Institute, Pittsburgh, PA (2002)
10. McGregor, J.D.: Preparing for Automated Derivation of Products in a Software Product Line. Software Engineering Institute, (2005)
11. Clements, P., Northrop, L.: Software Product Lines: Practices and Patterns. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (2001)
12. Halmans, G., Pohl, K.: Communicating the Variability of a Software-Product Family to Customers. Informatik - Forschung und Entwicklung **18** (2003) 113-131
13. Pohl, K., Böckle, G., v. d. Linden, F.: Software Product Line Engineering: Foundations, Principles, and Techniques. Springer, Heidelberg (2005)
14. 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
15. Kang , K.C., Cohen, S.G., Hess, J.A., Novak, W.E., Peterson, A.S.: Feature-Oriented Domain Analysis (FODA) Feasibility Study. Carnegie-Mellon University Software Engineering Institute, Pittsburgh, PA, USA (1990)
16. 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
17. O'Leary, P., Rabiser, R., Richardson, I., Thiel, S.: Important Issues and Key Activities in Product Derivation: Experiences from Two Independent Research Projects (awaiting notification of acceptance). Software Product Line Conference, San Francisco, CA, USA (2009)
18. Rabiser, R., Dhungana, D., Grünbacher, P., Lehner, K., Federspiel, C.: Product configuration support for nontechnicians: Customer-centered software product-line engineering. IEEE Intelligent Systems **22** (2007) 85-87
19. 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)