Important Issues and Key Activities in Product Derivation: Experiences from Two Independent Research Projects

Pádraig O’Leary¹ Rick Rabiser² Ita Richardson¹ Steffen Thiel³
¹ Lero, the Irish Software Engineering Research Centre, University of Limerick, Ireland
² Christian Doppler Lab. for Automated SW Eng., Johannes Kepler University Linz, Austria
³ Department of Computer Science, Furtwangen University of Applied Sciences, Germany

padraig.oleary@lero.ie rabiser@ase.jku.at ita.richardson@lero.ie steffen.thiel@hs-furtwangen.de

Abstract

When compared to the vast amount of research on domain engineering and building product lines, relatively little work has been dedicated to the derivation of individual products from product lines. Existing approaches to product derivation have been developed in isolation with different aims and purposes. The definition of a generic product derivation approach applicable to every domain may not be possible. However, comparing existing approaches allows the identification of both important issues to be addressed and key activities to be supported. In this experience paper we report on how we compared two product derivation approaches developed in two different, independent research projects. Based on the comparison and our experiences, we identify key activities that any approach to product derivation should consider. Additionally, we point out areas of uncertainty and identify remaining challenges within product derivation.

1. Introduction and Motivation

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” [1]. 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. [2] describe the process as “slow and error prone even if no new development is involved”. Griss [3] 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. [1] 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” [4].

Two product derivation approaches with different aims and purposes have been developed at both Lero (the Irish Software Engineering Research Centre) and JKU (Johannes Kepler University Linz, Austria) in the course of two different and independent research projects: (i) Pro-PD (Process framework for Product Derivation) was developed at Lero with the goal of defining a general process framework for product derivation [5]. (ii) DOPLER_UCon (Decision-Oriented Product Line Engineering for effective Reuse: User-centered Configuration) was developed at JKU driven by industry needs with the goal to define a user-centred, tool-supported product derivation approach [4].

Both, Pro-PD and DOPLER_UCon were designed to be generic, without focusing on a particular organisation or domain. In research collaboration between Lero and JKU we have compared our approaches in detail. While they have been developed in independent projects with different goals and for different purposes, we still found many interesting parallels.

The remainder of this paper is organised as follows: In Section 2 we present the background of this paper, i.e., existing work that influenced our research. In Section 3 we discuss the research methodology: we describe how Pro-PD and DOPLER_UCon were developed and how we performed the comparison. In Section 4 we briefly outline the activities of both approaches and then present important results and findings of the comparison. In Section 5, based on our experiences, we define key activities which should be supported by any product derivation approach and specify important issues to be addressed. In Section 6 we present important lessons learnt for product derivation. We conclude the paper with a summary and an outlook on future work in Section 7.
2. Background

The work presented in this paper was strongly influenced by Deelstra et al. [1] who present a product derivation approach and two industrial case studies. COVAMOF (COfiguration in Industrial Product Families VAriability MOdeling Framework) consists of two phases: an initial and an iteration phase. During the initial phase, a first product configuration is derived from a product line’s assets. This initial configuration is modified in a number of subsequent iterations during the iteration phase until the product sufficiently implements the requirements imposed. Requirements that cannot be accommodated by existing assets are handled by product-specific adaptation or reactive evolution. Parts of COVAMOF have been implemented in the research tool COVAMOF-VS [6]. The work by Deelstra et al. provides 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. [1] and others [6-11] agree is required for product derivation.

McGregor [7] describes a high-level framework of practices for deciding when to automate product derivation, how to choose the right technology, and how to plan and carry out the derivation process. According to the framework, production plans have to be developed to prepare the derivation process. Such plans are documents describing inputs, necessary activities, and desired outputs of product derivation. Chastek and McGregor [12] propose detailed guidelines for creating, using, and evaluating such production plans.

Bayer et al. [11] describe the “derivation part” of the PuLSE (Product Line Software Engineering) method developed at the Fraunhofer IESE called PuLSE-I (1 for instantiation). PuLSE-I activities cover planning product derivation, instantiating a product architecture from the product line reference architecture using decision models, and additional designing, implementation, and testing activities. Delivery and maintenance processes are also addressed. Several process steps are defined based on other PuLSE artefacts, e.g., reference architecture, domain decision model and scope definition.

Halman and Pohl [9] report on experiences of using extensions to standard notations like UML use cases to visualize and communicate variability to stakeholders in the application requirements engineering phase of product derivation.

Based on several extensions to FODA [8] feature models (e.g., cardinalities, groups, attributes) Czarnecki et al. [10] propose to perform derivation in stages, where some choices are eliminated in each stage. The output of each stage is a more specialized feature model. A configuration (where all choices have been eliminated) is derived from the most specialized feature model.

All these different approaches have been developed with different goals, for different purposes, and in different domains. Some are rather focused on the early phases of derivation [9, 10], some are intended to provide a (process) framework for product derivation [6], and others also focus on tool-support [13]. The development of both Pro-PD and DOPLER\textsuperscript{UC} has been influenced by these existing approaches. The important issues and key activities for product derivation (cf. Section 5) we derive from the experiences of comparing our approaches therefore also partly reflect this previous work.

3. Research Methodology

We briefly describe why and how Pro-PD and DOPLER\textsuperscript{UC} were developed and explain how we compared the two approaches.

3.1 Developing Pro-PD

The preparatory stage of developing Pro-PD was an extensive literature review that revealed a lack of methodological support for product derivation which we wanted to address. We constructed a preliminary version of Pro-PD based on the literature review. This preliminary version was iteratively developed and assessed through a series of workshops with two academic software product line (SPL) experts, one academic software product process expert, and one industry SPL expert, as well as through feedback from a leading SPL author. The output of this four month iterative development stage was version one of Pro-PD.

We then conducted case study research with Robert Bosch GmbH \textsuperscript{1} where we investigated product derivation practices within an automotive systems sub-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. We extended our framework by generalizing and discussing our observations. The output of this case study research resulted in version two of Pro-PD.

Pro-PD was further developed through a six month visit to LASSY lab\textsuperscript{2}; where Pro-PD and FIDJI [13] were mapped. FIDJI is a flexible product derivation process which is part of a model-driven SPL development methodology. Mapping Pro-PD to FIDJI provided academic validation for Pro-PD.

We used the Eclipse Process Framework (EPF) [14] to model Pro-PD. EPF supports the development, maintenance, and deployment of process content and assists in

\textsuperscript{1} http://www.bosch.com

\textsuperscript{2} Laboratory of Advanced Software Systems, University of Luxembourg
the development of situational method content. 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 framework is a good starting point for the integration of non-standard techniques such as agile practices, at appropriate times of the development process [15, 16].

3.2 Developing DOPLER\textsuperscript{UCon}

In research-industry collaboration with Siemens VAI Metals Technologies\textsuperscript{3}, the world leader in engineering and plant building for the iron, steel, and aluminium industries, we have developed the DOPLER approach to product line engineering. The goal of the collaboration was to support modelling the variability of Siemens VAI’s CC-L2 (Continuous Casting Level 2) software system for the automation of continuous casting in steel plants and to support the use of this system’s variability in product derivation.

Together with our industry partner we first analysed several existing product line approaches. We discovered a lack of integrated tool support for both variability modelling and product derivation that is adaptable and extensible enough to tailor it to Siemens VAI’s domain. We further found that the concepts of existing decision-oriented approaches, i.e., [17, 18], were preferred by Siemens VAI staff.

We developed our own decision-oriented approach based on this existing work. From the beginning, our goal was to develop an integrated, tool-supported approach for variability modelling as well as product derivation. Approaches and tools should be adaptable and generic to also be of use outside Siemens VAI.

We iteratively developed our approach and tools over a period of three years based on constant feedback and close collaboration with Siemens VAI. We applied our approach in other projects and for other systems to get more feedback. For example, we also applied our approach in another project in the enterprise resource planning domain [19]. Additionally we frequently presented our approach, the tools, and the ideas behind our work at several workshops and conferences to get the feedback from our peers, e.g., [4, 19-21]. In parallel, we conducted a systematic literature review which helped us define the issues to be addressed by our approach and tools that go beyond our industry partner’s requirements.

DOPLER comprises two parts: (i) DOPLER\textsuperscript{SM} [21] supports decision-oriented variability modelling and management. (ii) DOPLER\textsuperscript{UCon} [4] is a tool-supported, user-centred approach for product derivation.

DOPLER\textsuperscript{UCon} aims to support both domain experts like sales staff or managers as well as engineers in product derivation based on DOPLER variability models. Tool support developed for DOPLER\textsuperscript{UCon} provides derivation stakeholders with different views on decisions and assets and allows them to take decisions to derive a customized product. DOPLER\textsuperscript{UCon}’s development, and its validation are also described in [22].

3.3 Performing the Comparison

The idea of comparing Pro-PD with DOPLER\textsuperscript{UCon} emerged during a meeting of JKU and Lero researchers in February 2008. The main motivation at first was to learn from each other and try to improve both approaches. While Pro-PD was influenced by Deelstra et al. [1] and a case study with Robert Bosch GmbH, DOPLER\textsuperscript{UCon} was mainly influenced by the research-industry collaboration with Siemens VAI. While the first approach was developed as a generic methodology, the latter was developed focused on adaptable tool support usable in practical settings. These differences motivated our efforts of comparing the two approaches.

Table 1. Simplified version of a spreadsheet we used for a first high-level mapping of our two approaches.

<table>
<thead>
<tr>
<th>Pro-PD activity</th>
<th>Mapping (none/partial/full)</th>
<th>DOPLER\textsuperscript{UCon} activity</th>
<th>Explanation of mapping</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>…</td>
<td>Y</td>
<td>…</td>
<td>…</td>
</tr>
</tbody>
</table>

Based on initial discussions and existing documentation of our two approaches, we created a first high-level mapping in a distributed manner using spreadsheets to visualize commonalities and differences between the two approaches (cf. Table 1). Using such a high-level mapping, the authors of this paper met at SPLC 2008 to analyse the first results, discuss open issues, and detail the comparison. After this meeting we conducted several telephone conferences to work on the details of the comparison.

4. Comparing Two Product Derivation Approaches – Experiences and Results

We provide overviews of both Pro-PD and DOPLER\textsuperscript{UCon} and present important results and findings regarding the comparison of both approaches.

4.1 Overview of Pro-PD

From a high-level point of view Pro-PD comprises the following activities which need to be conducted in an iterative manner:

The goal of Pre-Derivation is to perform the preparatory steps required before actual derivation can begin. Pre-
Derivation is aimed at forming the product-specific requirements based on customer requirements and negotiation with the platform team. Requirements are prioritized and assigned to development iterations. Description of each of the sub-activities can be seen in Table 2.

In Product Configuration the goal is to build the product by reusing as much as possible the platform artifacts and minimizing the amount of product-specific development effort. Product derivation has to be an iterative process starting with selecting/customizing a set of assets from the product line, determining possible additional development, and testing. Requirements are developed iteratively based on their priority given in the previous step, iterations continue until all customer requirements have been fulfilled.

During Product Development and Testing, product specific development is undertaken. Both the changes and the final product are tested to ensure it satisfies customer expectations.

### 4.2 Overview of DOPLER\textsuperscript{UCon}

From a high-level point of view DOPLER\textsuperscript{UCon} comprises the following activities which need to be conducted in an iterative manner (see [4] for details):

- **Configuration Preparation** project managers prepare DOPLER variability models for a concrete project/customer. They capture customer information and already resolve variability based on high-level requirements known early on. They further define the roles and tasks of the people involved in product derivation. Additionally, domain experts model guidance on decisions to provide additional rationale or recommendations for decision-making. Configuration preparation is supported by the tool ProjectKing [4]. The output is a project-specific version of the original variability model called the derivation model.

- **Product Configuration** starts with presenting decisions to sales people and engineers according to their roles and tasks defined in the derivation model. Sales people communicate with customers to elicit their detailed requirements and take decisions accordingly. Engineers perform more technical configuration based on sales people’s decisions. Product configuration is supported by the tool ConfigurationWizard [20]. The outputs are selected and customized assets.

- **Application Requirements Engineering** aims at capturing, negotiating, and managing requirements that can not be fulfilled by the product line. These will likely arise during product configuration. ConfigurationWizard supports capturing such requirements and relating them to existing assets and decisions [4].

- **Additional Development** product-specific requirements are addressed. Developers have to take into account the already existing assets and their relationships.

New developments need to be tested. Activities like prototyping and unit testing are therefore typically involved.

*Product Integration and Deployment* means integrating derived assets with new developments and preparing them for deployment. The steps involved differ from company to company. ConfigurationWizard can be extended with domain-specific tools, e.g., to enable generating build files or settings files.

In *Product Line Maintenance and Evolution* domain and application engineers collaborate to find out which of the additionally developed and/or changed assets should become part of the product line.

### 4.3 Results and Findings

Due to space constraints it is not possible to describe the results of the comparison of our two approaches in detail. We chose to focus on the early phases of derivation (pre-derivation activity of Pro-PD and configuration preparation activity of DOPLER\textsuperscript{UCon}) to illustrate how we conducted the research. Preparing for derivation is a problematic area of product derivation because all further activities depend on these early steps. This fact is also reflected in Section 5.2, where many of the issues relate to handling requirements, documentation, and customer management.

In Table 2, we summarize which sub-activities of the pre-derivation activity in Pro-PD (cf. Section 4.1) were supported by DOPLER\textsuperscript{UCon} (cf. Section 4.2):

<table>
<thead>
<tr>
<th>Pro-PD Pre-Derivation Activity</th>
<th>Purpose</th>
<th>Supported by DOPLER\textsuperscript{UCon}?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rationalise Customer Requirements</td>
<td>“Translate” customer requirements (rqts) to domain language.</td>
<td>Not Supported (Customer rqts are assumed to be available in domain language).</td>
</tr>
<tr>
<td>Select closest matching configuration</td>
<td>Select a base configuration from existing/previous ones.</td>
<td>Supported (possible to start with an existing configuration)</td>
</tr>
<tr>
<td>Derive new configuration</td>
<td>Alternatively, derive a new base configuration.</td>
<td>Supported (Derivation can also start “from scratch” by creating a new derivation model).</td>
</tr>
<tr>
<td>Map Customer Requirements with Base Configuration</td>
<td>Determine rqts satisfied through base configuration and document mapped/unmapped rqts.</td>
<td>Supported (go through available decisions to find out which customer rqts can be fulfilled (-&gt; mapped rqts) and which not (-&gt; new rqts -&gt; capture).</td>
</tr>
</tbody>
</table>
Table 2 Continued: Overview of mapping the Pro-PD pre-derivation sub-activity to DOPLER\textsuperscript{UConP}.

<table>
<thead>
<tr>
<th>Pro-PD Pre-Derivation Activity</th>
<th>Purpose</th>
<th>Supported by DOPLER\textsuperscript{UConP}?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customer Negotiation</td>
<td>Negotiate unmapped customer rqts and check their feasibility.</td>
<td>Supported (relating new rqts with variability. Based on this information effort and risk level for realization can be defined - Customer rqts are negotiated with the customer).</td>
</tr>
<tr>
<td>Modify Configuration</td>
<td>Alter the base configuration based on results of customer negotiation.</td>
<td>Supported (Customer rqts that can be fulfilled lead to taking decisions that fulfill these rqts – “new” rqts lead to additional development).</td>
</tr>
<tr>
<td>Form Product Specific Require-ments</td>
<td>Involves merging mapped and negotiated rqts.</td>
<td>Supported (Negotiations lead to changes in captured rqts).</td>
</tr>
<tr>
<td>Discipline Mapping</td>
<td>Allocate rqts to relevant disciplines, e.g., hardware discipline, algorithms, ...</td>
<td>Partly Supported (Tasks in derivation models group related decisions. Rqts related with decisions in a task also allocated to a discipline).</td>
</tr>
<tr>
<td>Allocate Requirements to specific iterations based on priority</td>
<td>Prioritise implementation iteration of particular product rqts.</td>
<td>Not Supported (not part of DOPLER\textsuperscript{UCon} because assumed to be too domain-specific).</td>
</tr>
<tr>
<td>Create Product Specific Test Cases</td>
<td>Create test-cases using the product-specific rqts.</td>
<td>Partly supported (assumed to happen in additional development phase but not defined how).</td>
</tr>
</tbody>
</table>

From the ten activities identified within Pro-PD, we can see that eight activities are fully or partly supported by DOPLER\textsuperscript{UCon}. Only two activities, i.e., “rationalise customer requirements” and “allocate requirements to specific iterations based on priority”, are not supported.

The missing support for rationalising customer requirements in DOPLER\textsuperscript{UCon} can be explained with the differences in customer management. In a collaborative environment, as assumed by DOPLER\textsuperscript{UCon}, customer requirements are typically delivered in a product line compatible format.

The use of iterative development cycles is not directly supported in DOPLER\textsuperscript{UCon}. However, additional attributes can be defined for requirements and these can be used to allocate specific requirements to specific iterations. Pro-PD is designed with iterative development cycles in mind. The specification of product-specific requirements goes hand in hand with allocation of these requirements to specific iterations based on prioritisation and customer negotiation.

Even though the two approaches were developed separately and with different aims and purposes in mind, we could discover many interesting parallels (cf. Table 2) and comparably few differences:

Requirements management is one area where Pro-PD and DOPLER\textsuperscript{UCon} have different approaches. In DOPLER\textsuperscript{UCon} customer requirements that can not be satisfied by the product line are captured and documented together with relations to existing variability. Pro-PD takes unsatisfied customer requirements and performs customer negotiation where the feasibility of implementing customer requirements is investigated and discussed with the customer. DOPLER\textsuperscript{UCon} does not clearly define how to handle/negotiate unsatisfied customer requirements, it is “only” possible to capture these requirements and mark them as product-specific implementations.

Pro-PD is applicable to organizations seeking to achieve regulatory compliance such as Auto-SPICE [23] due to specific practices dedicated to the formation of requirements specifications. DOPLER\textsuperscript{UCon} would require additional requirements specification practices to make it applicable in regulated environments.

DOPLER\textsuperscript{UCon} is focused on providing user-centred tool support for product derivation. For example, different views on existing variability are provided for different users to allow them taking decisions. Pro-PD does not define which activities should be supported by tools and how they can be supported.

Product derivation user management is also not directly supported in Pro-PD. While DOPLER\textsuperscript{UCon} requires defining the people involved in product derivation and their roles and responsibilities (who decides what and when), Pro-PD does not explicitly enforce such a user management.

5. Key Activities and Important Issues in Product Derivation

When we began this research collaboration, our first motivation was to learn from each other and improve our approaches according to the results of our comparison. However, we quickly realized that we could also use the results of the comparison and our experiences for other purposes. We were able to define key activities to be supported and important issues to be addressed by product derivation approaches. This is of interest to both research and industry.
5.1 Key Activities to be supported

Based on our general experiences and the experiences we made through mapping our two approaches we define the following key activities to be supported by a product derivation approach (cf. Figure 1):

**Preparing for Derivation.** Derivation does not start “from scratch”, i.e., by just selecting features or taking decisions described in a variability model. From both research projects, we observed that before actual derivation can start, several preparatory steps need to be conducted:

Customer requirements need to be 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.

A “base configuration” may be chosen as a starting point for derivation, i.e., from a set of existing platform configurations. Experiences made in past projects are of great use as similar customers often have comparable requirements. If no matching base configuration can be found, a new one has to be created.

Customer requirements then have to be mapped to the base configuration. Requirements which cannot be satisfied by existing assets have to be negotiated with the customer. Effort estimation issues such as those described in Section 5.2 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.

The role and task structures for the product derivation project have to be defined. For example, a discipline mapping can be performed where product requirements are allocated to relevant disciplines. The goal is to define who is responsible for resolving what remaining variability in product derivation to fulfill the product requirements. This is very helpful to provide different views on variability for different people involved in product derivation and helps to lower the complexity of large decision spaces. Also, as the duration of product derivation projects can be quite long, it is important to know who decided what and when.

Preparing for derivation also means to create guidance for decision-makers. Guidance is essential, especially for customers and sales people, who are confronted with many – often technical – decisions or features. Remaining variability must be explained to deal with complexity issues in representing product line variability (cf. Section 5.2).

![Figure 1. Key activities to be supported by a product derivation approach.](image)
velopment will be required. Required development activities have to be defined and prioritized based on customer requirements.

Additional Development/Testing. It is the responsibility of the product development team to implement the required changes at the product level.

When a component is built or adapted, initial or tailored versions of a component need to be tested rigorously, for example through unit testing. As confirmed by Kauppinen [24], conventional unit test methods must be utilized as no product line specific methods have been developed so far.

Newly developed/adapted assets need to be integrated with the partial product configuration. This can for example require writing sufficient “glue” code to interfaces [25] or implementing architectural changes to facilitate integration of the developed/adapted assets.

Integration testing is essential to find out whether the newly developed/adapted assets interact correctly with the existing architecture: The product has to be checked for consistency and correctness.

In system testing the product has to be checked for compliance with the product-specific requirements [1]. If the customer requirements for this iteration have been satisfied, the product is delivered. Otherwise, further iterations are required (selecting/customizing assets and additional development/testing).

From our experience, all the activities we described should be supported by a product derivation approach. How the activities are conducted strongly depends on the domain. In some cases it might be best to define a domain-specific derivation approach. In other cases a generic approach might be more useful. The activities we defined can be used as a checklist when defining, adapting, or evaluating a product derivation approach for a certain domain, context, or problem.

5.2 Important Issues to be addressed

We have identified important issues in product derivation that should be addressed by further research. From our experience, these issues contribute to making product derivation “a time-consuming and expensive activity” [1].

There are no mechanisms supporting effort estimation in product derivation. The estimation of effort required to satisfy unmapped customer requirements through the adaptation of platform assets is an important product derivation task. Unfortunately there are no industry mechanisms to support this activity. This can lead to poor development scoping decisions and best-guess project delivery. Furthermore, customer negotiation is a difficult process because organizations find it difficult to predict effort and estimate cost of particular customer requirements.

Documentation is not used or used for the wrong purpose. Generally, organizations either have a culture of using documentation or not. Organizations that do use documents tend to use it in response to other problems. For instance, in communicating information across large distributed teams organizations often over-rely on documentation. Organizations’ documentation becomes bloated as teams attempt to capture too much. This over-explicit documentation decreases traceability of relevant information and results in failure to correctly identify the artefacts for reuse, especially in teams where the transfer of tacit knowledge is prohibitive.

Alternatively, organizations 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 not all compatibility aspects between these components are externalized. Generally, this problem is observed in later stages of product derivation and forces product engineers to return to product assembly in order to select other components. Consequently, this leads to wasted time.

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

Poor requirements elicitation practices lead to poorly specified requirements. The specification of incompatible customer requirements and undocumented dependencies can be costly further down the product derivation process. The size and complexity of variability models of large-scale product lines exasperates the issue, as difficulties in communicating the variability provided by the product line lead to unrealistic customer requirements.

Representing product line variability is difficult. The size and complexity of product lines make representing product line variability difficult. Different people need to understand different aspects of the provided variability. While sales people interacting with customers need to understand variability from a rather high level, engineers need more details to perform technical configuration. Depending on the tasks, roles, and responsibilities of the people involved, different representations of the available variability are required; e.g., for the customer to elicit his requirements, for business-oriented stakeholders like sales people and project managers who interact with the customer, and for engineers who are responsible for technical configuration.

Current process models and tools do not integrate well. Process integration is a key criterion. All the different
people involved in product derivation are supported in their tasks by different approaches and tools. Because of the difficulty of integrating these different approaches and tools, product derivation can become an error-prone and tedious task quickly. Organisations need a product derivation approach that can be easily integrated with different existing processes and environments. To support this, research into new approaches must ensure that they are adaptable and that supporting tools are flexible and extensible.

6. Lessons Learnt for Product Derivation

From our collaboration with different industrial partners and from comparing our two approaches we learnt some lessons that are valuable for others working on product derivation. The applicability of these lessons depends on the domain and context and on whether an existing approach is used (last three lessons) or a new one is developed (all five lessons).

Find a balance between being generic and too specific. A product derivation approach useful in many different domains easily becomes too generic. This impedes its use in practice as many adaptations have to be made before it can be applied. Approaches and tools must be specific enough to be useful but not too specific. Finding a balance is very hard to achieve yet absolutely necessary for real-world success.

Focus derivation approaches and tools on their users. Many existing product derivation approaches and tools have not been developed with the needs of concrete (or at least possible) users in mind. Throughout our collaborations with industrial partners we have learned that approaches and tools have to focus on the needs of the different involved people to be useful (and used!) in practice. Derivation approaches should not be developed in the laboratory but with close collaboration and involvement of real users.

Present variability in the language of the product derivation stakeholders. It does not matter which product line approach is used – be it feature-oriented, decision-based, or something else. In any case, the variability of the product line must be presented in the language of the different derivation stakeholders. Different people – from customers, over marketing and sales, to engineering staff – have to understand different aspects of the variability to be able to make their choices. This is only possible if variability is presented in a way that is understandable by the different people. While engineering staff might want rather technically phrased choices, sales staff and customers are more interested in the added value of the choices to be made.

Minimize deviations from the standard. Only in the “blue-sky” scenario of product derivation all customer wishes can be fulfilled by exploiting the variability the product line provides. Typically, customers will have additional wishes and requirements that lead to additional development effort. Negotiation with customers is essential to minimize such – often costly – “deviations from the standard”. Often customers can be convinced that a much cheaper solution based on the product line will also do, even if it only fulfils 80 percent of their requirements [9].

We have learned that customers’ additional requirements are one of the main reasons making product derivation costly and tedious.

Define a synchronisation strategy for product and platform teams. Product development requires a high degree of coordination and communication. The product team designs and implements customer-specific assets based on customer requirements. The platform team receives the platform software requirements containing a description of the required extensions to the existing platform to facilitate the new customer requirements. Both, customer-specific and platform development occur in parallel. The product team needs to interface correctly with the new platform release which is why defining a synchronisation strategy is quite helpful.

7. Conclusions and Future Work

We found the comparison of our two approaches very beneficial. The researchers were exposed to alternative viewpoints. We had to carefully consider tool integration within Pro-PD while DOPLER had to justify the use of tools within a specified process. Researchers gained a better sense of the strengths and weaknesses of their particular approach. The collaboration fostered discussion and debate, the results of which we presented here.

The definition of a generic product derivation approach applicable to every domain may not be possible. However, comparing existing approaches allows defining important issues to be addressed and key activities that should be supported. The experience gained through the analytical comparison of our approaches has helped us to improve our individual research. The observations made are of interest to both researchers and industry practitioners:

For academia, our results provide structure to the area under concern. As a roadmap, our work points to areas of uncertainty and helps to identify remaining challenges. Such a roadmap encourages the insertion of those pieces that may be missing, or the extra detail that may be needed. The roadmap can offer clarity to the field. We do not claim that the issues and lessons learnt we presented are complete however we hope that other researchers can use our work as a starting point for presenting their experiences with product derivation.

For industry, despite the growing adoption of SPL, product derivation remains an expensive and error-prone activity which is hard to automate and support by tools.
Our research can support the adoption of automated product derivation approaches through the structuring and identification of the involved activities into a larger process model. We see the contribution of this roadmap to the automation of product derivation as twofold: (i) it allows us to put automated approaches, which tackle one particular task, into a bigger context and, (ii) it lays the foundation for tools which support the overall process.

In our future work, we aim to investigate the tool and automation needs for industrial product derivation. We also plan to perform detailed comparisons of Pro-PD and DOPLER with other product derivation approaches to further elaborate on areas of uncertainty and remaining challenges. Based on these comparisons and the results of further projects at Lero and JKU, we will improve both approaches. The authors will keep in touch to continuously learn from each other.

8. Acknowledgements
This work is partially funded by IRCSET under grant no. RS/06/167 and by Science Foundation Ireland under grant no. 03/CE2/1303 1. This work has also been supported by the Christian Doppler Forschungsgesellschaft, Austria and Siemens VAI Metals Technologies. We also want to thank Iris Groher for valuable feedback on earlier versions of this paper.

9. References


