Information and Software Technology 54 (2012) 1014–1028

Contents lists available at SciVerse ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

The Pro-PD Process Model for Product Derivation within software product lines

Pádraig O'Leary [a,*], Eduardo Santana de Almeida [a], Ita Richardson [b]

[a] RiSE – Reuse in Software Engineering and Computer Science Department, Federal University of Bahia, Salvador, BA, Brazil
[b] Lero – The Irish Software Engineering Research Centre, University of Limerick, Ireland

ARTICLE INFO

Article history:
Received 12 May 2011
Received in revised form 13 March 2012
Accepted 27 March 2012
Available online 1 April 2012

Keywords:
Software product lines
Product derivation
Process

ABSTRACT

Background: The derivation of products from a software product line is a time consuming and expensive activity. Despite recognition that an effective process could alleviate many of the difficulties associated with product derivation, existing approaches have different scope, emphasise different aspects of the derivation process and are frequently too specialised to serve as a general solution.
Objective: To define a systematic process that will provide a structured approach to the derivation of products from a software product line, based on a set of tasks, roles and artefacts.
Method: Through a series of research stages using sources in industry and academia, this research has developed a Process Model for Product Derivation (Pro-PD). We document the evidence for the construction of Pro-PD and the design decisions taken. We evaluate Pro-PD through comparison with prominent existing approaches and standards.
Results: This research presents a Process Model for Product Derivation (Pro-PD). Pro-PD describes the tasks, roles and work artefacts used to derive products from a software product line.
Conclusion: In response to a need for methodological support, we developed Pro-PD (Process Model for Product Derivation). Pro-PD was iteratively developed and evaluated through four research stages. Our research is a first step toward an evidence-based methodology for product derivation and a starting point for the definition of a product derivation approach.

1. Introduction and motivation

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 a number of products is designed and implemented, and application engineering, where a product is derived based on the platform components [2]. The separation into domain engineering and application engineering allows the development of software artefacts which are shared among the products within that domain. It is during application engineering that the individual products within a product line are constructed. The process of creating these individual products using the platform artefacts is known as product derivation. Product derivation is concerned with the construction process of products and does not consider application engineering tasks such as system delivery, maintenance or support.

A number of publications discuss of the difficulties associated with the product derivation process. Hotz et al. [2] describe it 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, the derivation of individual products from shared software assets is still a time-consuming and expensive activity in many organisations [4,5]. As Deelstra et al. [5] state there "is a lack of methodological support for application engineering and, consequently, organisations fail to exploit the full benefits of software product families." As a result, current approaches fail to provide a holistic view of product derivation. This leaves organisations with no centralised starting point for defining an approach to product derivation and results in ad hoc solutions. Consequently, there is a strong need for a structured approach to product derivation which defines activities, tasks, roles, inputs and outputs of each step in a structured and systematic way.

Despite this, there has been little work dedicated to the overall product derivation process. Rabiser et al. [6] claim that "guidance and support are needed to increase efficiency and to deal with the complexity of product derivation". This paper aims to fill this identified gap and the objective of the research presented is stated

* Corresponding author.
E-mail addresses: padraig.oleary@rise.com.br (P. O'Leary), esa@rise.com.br (E.S. de Almeida), ita.richardson@lero.ie (I. Richardson).

0950-5849/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.infsof.2012.03.008

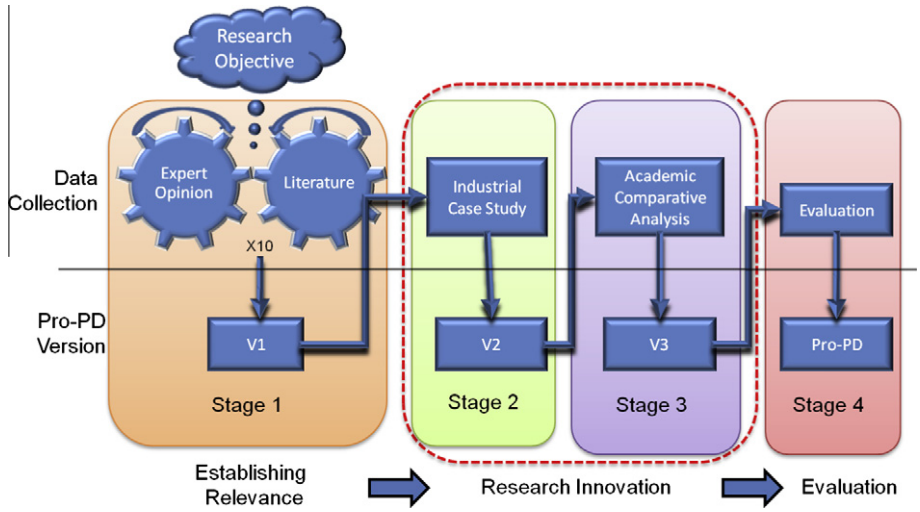

Fig. 1. Overview of research design.

as: *To define a systematic process which will provide a structured approach to the derivation of products from a software product line based on a set of tasks, roles and artefacts.*

In order to achieve this objective, we have developed and evaluated the Pro-PD process for product derivation. Pro-PD is the conclusion of research which has been iteratively developed through four research stages (see Fig. 1). In-progress results were previously published, for example, in [4], and along with the final model, the results of each of these research stages are presented in this paper.

The remainder of this paper is structured as follows: Section 2 discusses existing approaches to product derivation. Section 3 presents the research design. In section 4, we present an overview of the final Pro-PD process reference model. In Section 5, we describe the design decisions taken during the development of Pro-PD. Section 6 presents the evaluation of Pro-PD. Section 7 concludes the paper.

2. Related research

A number of models have been developed to support software product line development within organisations. These include PuLSE, FAST, Dream, COVAMOF, DOPLER^{UCon} (**D**ecision-**O**riented **P**roduct **L**ine **E**ngineering for effective **R**euse: **U**ser-centered **Con**figuration), FIDJI, FORM and KobrA and the SEI Product Line Practice Framework (PLPF). In Section 2.1 we will briefly describe these approaches focusing on the process of application engineering or product derivation. According to Deelstra, product derivation is the process of constructing a product from a software product line's (SPL) core assets [5]. Therefore this research does not consider specific development techniques or tool approaches but focusses on the process involved in the derivation of products. Our observations are described in Section 2.2.

2.1. Main approaches

PuLSE (**P**rod**u**ct **L**ine **S**oftware **E**ngineering) [7] is a method engineering framework consisting of three major elements: Deployment Phases, Support Components and Technical Components. PuLSE-I [8] activities include planning product derivation. However, the approach defines roles and tasks on a very high-level. According to Atkinson et al. [9] while PuLSE proved to be helpful for introducing sound documentation and development techniques into existing development practices, where a formalised process did not exist, the introduction of PuLSE in industry turned out to be problematic.

The FAST application engineering process [10] covers requirements elicitation, requirements analysis, product configuration, and additional development and testing. Product derivation is greatly simplified by describing the products in the application modelling language and individual products can be developed quickly. However, to enable automatic product derivation, system specifications must be precisely defined and specified. Additionally, since requirements are specified using custom domain specification languages, other organisations may have difficulties in adopting them.

Kim et al. [11] propose an overview of a complete method, Dream, which integrates both SPL engineering and model-driven architecture. However, there is little support for the derivation process other than a high level description of the activities required. No guidance is provided as to its use within an industrial setting.

A product derivation framework presented by Deelstra et al. [5] was developed based on two industrial case studies. A first product configuration is derived from the product line artefacts. This initial configuration is modified in a number of subsequent iterations until the product sufficiently implements the imposed requirements. This work by Deelstra et al. provides a framework of terminology and concepts for product derivation (part of COVAMOF approach). The approach is tool-focused and centred on the configuration of products. However, there is no support for the early phases of product derivation or product specific development and testing. The framework assumes that 'engineers' perform all the work and does not specify responsibilities, nor does it define role and task structures. The framework focuses on product configuration and is only a high-level attempt at providing the methodological support that Deelstra et al. [5] and others [8,12,13] agree is required for product derivation.

DOPLER^{UCon} [14] is a tool-supported approach for product configuration with capabilities for adapting and augmenting variability models to guide sales people and application engineers through product derivation. In a research-industry collaboration with Siemens VAI Metals Technologies, the researchers developed a model to support modelling variability in product derivation. DOPLER^{UCon} is focused on providing user-centred tool support for product derivation, rather than supporting the product derivation process within the approach. DOPLER^{UCon} is discussed in more detail in Section 5.3.

The SEI Product Line Practice Framework (PLPF) [1] defines 29 practice areas. A practice area is a body of work or a collection of

activities that an organisation must master to successfully carry out the essential work of a product line [1]. The PLPF will be discussed in more detail in Section 6.

In addition to the PLPF, the SEI also supports the automation of product derivation. McGregor [13] 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* are used to prepare the derivation process. Such plans are documents describing inputs, necessary activities, and desired outputs of product derivation. Chastek and McGregor [15] propose detailed guidelines for creating, using, and evaluating such production plans.

The framework defined by the SEI is a robust description of best practice involving important technical and non-technical aspects grouped in software product line practical areas. However, the framework is generic and does not define process support. There is a strong focus on planning product derivation with the ultimate goal to automate the derivation process.

FORM [16] was directly derived and developed from FODA [17] and extends it to the software design phases. It applies software engineering principles but gives only a few suggestions regarding implementation. KobrA [9] has been derived as an instantiation of PuLSE with the intention of rapid use. It makes use of UML diagrams and templates for the defined development items and is component-oriented. However, it does not provide the required process detail.

The approaches and methods evaluated have very different scope and emphasise different aspects of the derivation process. Some of them, like FODA [17], capture only a small part of the process while others, like PuLSE-I [8] are much broader. All of them (with the exception of the SEI PLPF) come with different amounts of prescription and tool support. Some such as [5,8,11,16,18], describe a generic process at a high level while others such as [9,10,19] are very close to practice and are prescriptive in the definition of their process steps and documentation.

2.2. Limitations of current approaches

From our analysis of current approaches, we have identified key areas which have not been resolved by previous models. Furthermore, evidence collected during the course of this research indicated that from an industry perspective, these areas need to be addressed. Therefore, the development of a model for product derivation should include:

- Definition of the flow of artefacts.
- Definition of roles and responsibilities.
- Provision of process support.

2.2.1. Definition of flow of artefacts

Product development within a SPL requires a high degree of coordination and communication. Frequently, both customer-specific and platform development occur in parallel. There is a need for awareness of the artefacts and the stakeholders involved in product derivation.

Many of the existing approaches do not explicitly define the flow of artefacts within product derivation. In Dream [11] little information is provided on the usage of model artefacts during the different phases of the approach. DOPLER^{UCon} does provide some description of artefact flow however the specified level of granularity is too abstract to give useful guidance for practitioners. PuLSE-I [8] names the development items in a descriptive manner. However, like the others, it does not provide a detailed description of artefact usage within the process.

2.2.2. Definition of roles and responsibilities

Diverse people with diverse tasks, roles, and responsibilities are involved in product derivation. Current approaches do not provide sufficient support for the managing of roles and assignment of roles to tasks and artefacts within the product derivation process. DOPLER^{UCon} acknowledges the need for defining roles and responsibilities during a derivation project but no guidance is provided on how this should occur. FAST [10] assigns activities to one of the three defined derivation roles but this is done at a very high level and is unusable in any practical setting.

2.2.3. Provision of process support

The arguments for defining a process should be a familiar one. A well-defined process can be managed, measured and observed, and therefore improved. An emphasis on processes helps software development to become more like engineering, with predictable time and effort constraints, and less like art [20]. Böckle et al. [21] finds that transforming an organisation to create products as members of a product line requires installing corresponding processes, and organisational structure and methods.

Clements and Northrop state that: "*organisations that do not have a strong process culture will find deploying a successful product line a perilous proposition*" [1].

Furthermore, key authors in the area have called for process support within product derivation [5,6,13,22]. Despite this, there is relatively modest support for the derivation process in existing approaches. KobrA [9] provides a detailed analysis model and some methodological guidelines. However a detailed description is not provided. Deelstra et al. [5] provide a framework of terminology and concepts, however it is presented at a high level. A good starting point could be PuLSE-I [8], as it names and briefly describes the development items, which roles are responsible for which tasks and which artefacts are consumed or produced by a certain activity. Furthermore, many dependencies exist to other parts of the PuLSE process which means that the approach is not very applicable outside the scope of a PuLSE applied product line.

2.2.4. Pro-PD provisions

The development of a useable process reference model needs to explicitly consider these limitations. For those models that we have reviewed, there are a number of defined processes but those described are too abstract or not explicit in their description of the process, lacking defined flow of artefacts, roles and responsibilities. Furthermore, they are frequently too specialised to a specific development technique to serve as a general solution. The research did not consider specific development techniques or approaches to product derivation. In Pro-PD, we have aimed to be cognisant of each of these so that the model is usable in practice.

3. Research methodology

The goal of our research is to provide an evidence based process approach for product derivation. With this in mind, our research design was influenced by Ahlemann and Gastl [23] which focused on empirically grounded and valid process model construction. In an analogy with systems engineering, the overall construction process is based on a cyclic structure to allow for model corrections on preceding construction stages via feedback-loops. Although the stages are dealt with sequentially, they contain cyclic sub-processes. The research design is compatible with common suggestions for qualitative research designs in process models [24]. Stages 1 and 2 are the primary construction steps. Stage 3 is both a development and an evaluation step. Stage 4 is the evaluation step. An overview of the research design is presented in Fig. 1.

Stage 1, core construction, entailed a literature review from which a preliminary version of the model was developed. The literature review aimed to identify the fundamental practices of product derivation, through studying existing identified product derivation practices. Concurrent to the literature review, a series of iterative expert opinion workshops was held. Participation of expert users in the core construction stage is emphasised by Rosemann and Schütte [25] and Schlagheck [26], as the users are the subject-matter experts of the problem domain. Furthermore, as the research is designed for use in both industry and academia, the selection of experts should reflect this. With this in mind, the participants were two academic SPL experts with 20 years of experience, an industrial SPL expert with 10 years of experience and a software process improvement expert.

Participants met twice per month for 6 months. At each workshop, the model was presented to the experts and was evaluated using formal questions on model structure. The model was discussed amongst the group until a consensus was formed and the model was revised. After each workshop we returned to the literature and based upon the expert revisions, Pro-PD V1, was iteratively developed.

Stage 2 was an industrial case study within Robert Bosch GmbH. This was carried out as an inductive, empirical validation [23]. We chose a case study as they are often considered to be the optimal approach for researching practice based problems, where the aim is to represent the case authentically "in its own terms" [27]. Pro-PD V1 was mapped and compared to product derivation within the company. Robert Bosch GmbH was chosen for the case study because previous SPL efforts have been judged a success by their peers [28]. The case study was carried out in conjunction with the corporate research division. This case study was dual-purpose. In the first instance, we modelled the Bosch product derivation process for their internal use and then we updated Pro-PD based on our observations. This resulted in Pro-PD V2.

In conducting the case study, we analysed internal company documentation, which illustrated the existing process through completed projects. We then organised an onsite visit including a 2-day workshop with the corporate research division of Robert Bosch GmbH. Attendees included selected product architects and developers from product line business units within the company. The primary researcher (O'Leary) was accompanied by two other researchers, one of whom had published extensively on case study research. After the workshop, a technical report on the company's product derivation process was created and validated through feedback with Bosch SPL experts. Both the documentation analysis and the workshop outputs were used to identify what components should be included in Pro-PD V2.

Stage 3 of the research, an academic comparative analysis, was carried out during a research collaboration with Johannes Kepler University Linz, Austria (JKU). The results of this comparative analysis are available here [4]. JKU had previously developed the DOPLERP^{UCon} approach. Based on initial discussions and existing documentation of our two approaches, a high-level mapping was created. This was done in a distributed manner using spreadsheets to visualise commonalities and differences between the two approaches. Using this mapping, the authors of this paper met to analyse the first results, discuss open issues, and detail the comparison. We then conducted several telephone conferences with JKU researchers to work on the details of the comparison. Pro-PD V2 was compared to the activities identified by DOPLER for Siemens VAI. Based on this comparison Pro-PD V3 was developed.

Pro-PD V3 was evaluated during stage 4 of the research by studying how key activities were supported by prominent existing approaches and by demonstrating how Pro-PD was compliant with existing standards. Compliance with standards and norms is proven by a direct mapping of reference information model elements. In this stage, we were not looking to identify new elements but to validate the existing elements of Pro-PD or identify gaps within Pro-PD. This stage was performed in two steps.

Firstly, an inter-model evaluation was conducted with the SEI PLPF. During the evaluation, Pro-PD V3 was compared to the SEI PLPF. According to Ahlemann and Gastl [23], process models that are compatible with such standards and norms can be regarded as high quality.

Secondly, we systematically evaluated Pro-PD V3 by analysing support for its activities in three independently developed, published and highly-cited approaches: COVAMOF [22], FAST [10], and PuLSE-I [8]. The approaches have been developed with different goals, for different purposes, and in different domains. Furthermore, in our literature review we identified that these three approaches were influential through their frequent citations.

Although a framework for evaluating product derivation approaches does not exist, we adapted a framework[1] developed for the purpose of evaluating software product line architecture design methods [29]. The original framework by Matinlassi [29] considers software product line architecture design methods by focusing on method context, user, structure and validation. Its purpose is to study and compare existing approaches for their design of software product line architectures. We used this framework as a basis for our validation for two reasons. Firstly, it provided a simple tabular evaluation structure. Secondly, it had previously been published at ICSE, which ensures that it has been peer-reviewed.

We adapted the questions regarding the category context proposed by Matinlassi [29] from "product line architecture design method" to "product derivation approach". We adopted only one element for the category user (target group) as our focus is on evaluating the contents (support for key activities) and not the user support. For the category contents, we adopted the first two elements activities and artefacts. Through following the research stages outlined, we developed and evaluated Pro-PD: Process Model for Product Derivation.

4. Pro-PD – a Process Model for Product Derivation

When these four stages of the research were completed (Core Construction, Industrial Case Study, Academic Comparative Analysis and Evaluation), Pro-PD: Process Model for Product Derivation had been developed. Pro-PD focuses on the roles, artefacts, tasks and activities used to derive products from a software product line. Fig. 2 illustrates the interaction of these various process elements in a matrix. Table 1 presents the matrix legend. These various elements represent the process building blocks of Pro-PD. Roles represent a set of related skills and responsibilities. Artefacts are produced, modified or used by tasks. Tasks are assignable units of work that usually consume or produce one or more artefacts. Activities are grouping of related tasks that share a specific development goal.

4.1. Units of work: tasks and activities

Pro-PD contains the following activities:

- *Initiate Project* – the preparatory tasks required to establish a product derivation project.
- *Identify and Refine Requirements* – the preparatory tasks required to commence a new iteration of the product derivation project.
- *Derive the Product* – creates an integrated product configuration that makes maximum use of the platform and minimises the

[1] This work was a result of a collaboration with Dr. Rick Rabiser.

Activities/Tasks	Roles						Work Products																			
	Customer	Product Analyst	Product Architect	Product Developer	Product Manager	Product Tester	Base Product Configuration	Customer Requirements	Customer Specific Product Requirements	Developed or Adapted Components	Existing Platform Configurations	Glossary	Integrated Product Configuration	Negotiated Customer Requirements	New Platform Release	Platform Architecture	Platform Components	Platform Feedback	Platform Requirements	Platform Test Artefacts	Product Release	Product Specific Platform Requirements	Product Requirements	Product Test Cases	Required Product Development	Selected Platform Components	Translated Customer Requirements
Initiate Project																										
Translate Customer Requirements		P						<				<														>
Coverage Analysis			P						>									<								</>
Customer Negotiation	P				P				<					>												
Create the Product Requirements			P											<				<				>				<
Verify the Product Requirements	a		P		a																	</>				
Define Role and Task Structures					P																	</>				
Identify and Refine Requirements																										
Find and Outline Requirements		P			a																>	</>				
Create the Product Test Cases						P													<			<	>			
Allocate Requirements					P																	</>				
Create Guidance for Decision Makers	a		P																			</>				
Derive the Product																										
Select Closest Matching Configuration			P				>				<											<				
Derive New Configuration			P				>									<						<				
Evaluate Product Architecture			P			a	</>											>				<				
Select Platform Components			a	P			<										<					<			>	
Platform-Product Integration			P				<						>												<	
Platform-Product Integration Testing						P	<						</>				<			<						
Identify Required Product Development			P	a									<									<		>		
Develop the Product																										
Component Development				P						>					<		<							<		
Component Testing				P						</>										<				<		
Component Integration			P	a						<			<								>					
Component Integration Testing			a	a		P														<	>					
Test the Product																										
Run Acceptance Tests						P															</>			<		
Management and Assessment																										
Provide Feedback to Platform Team		a	a	a	P	a			<									</>					<			
Manage Project					P																		</>			

Fig. 2. Pro-PD matrix.

Table 1
Pro-PD matrix legend.

Symbol	Description
P	Primary Performer
A	Secondary Performer
<	An input to a task
>	An output from a task
</>	Both an input and an output from a task

amount of product specific development required.

- *Develop the Product* – facilitates requirements that could not be satisfied by a configuration of the existing assets through component development or adaptation.
- *Test the Product* – validates the current product release.
- *Management and Assessment* – provides feedback to the platform team and monitors progress of derivation project.

Fig. 3 gives on overview of these Pro-PD activities and the iterative nature of the Pro-PD process. Development iterations allow the phased implementation of the product requirements. This permits partial validation of the product with the customer before the derivation is finished, assisting in identifying whether additional development iterations are required.

4.1.1. Initiate Project

Derivation does not start "from scratch", i.e., by just selecting features or taking decisions described in a variability model. The *Initiate Project* activity contains the preparatory tasks required to establish a product derivation project. Requirements elicitation is distributed across three of these tasks: *Translate Customer Requirements*, *Coverage Analysis* and *Create the Product Requirements*. Table 2 describes the *Initiate Project* tasks and their purpose.

4.1.2. Identify and Refine Requirements

The *Identify and Refine Requirements* activity contains the preparatory tasks required to start a new product derivation iteration. Table 3 describes the *Identify and Refine Requirements* tasks and their purpose.

4.1.3. Derive the Product

The *Derive the Product* activity contains the tasks required to create a integrated product configuration that makes maximum use of

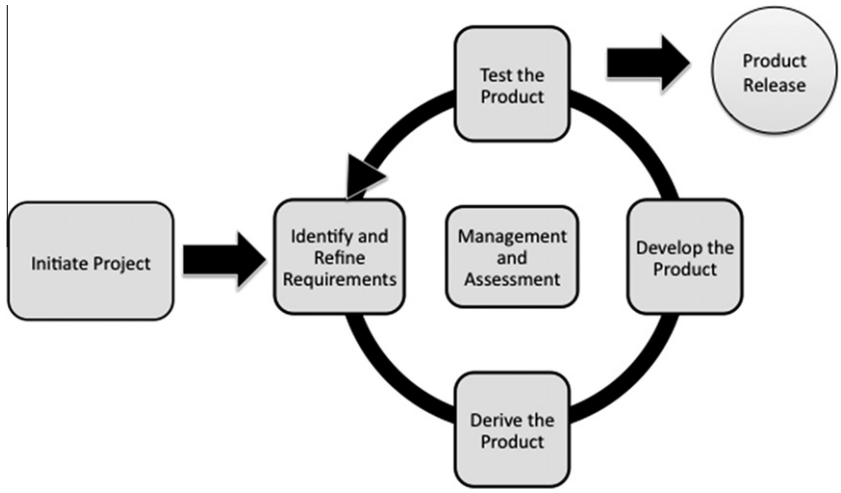

Fig. 3. Overview of Pro-PD activities.

Table 2
Initiate Project Tasks.

Task	Purpose
Translate Customer Requirements	To translate the *Customer Requirements* into the internal organisational language. The *Product Analyst* used a customer terminology *Glossary* to assist in translating specific customer terms to platform specific terminology. Output is the *Translated Customer Requirements*
Coverage Analysis	To perform a comparison between the *Translated Customer Requirements* and the *Platform Requirements*. The *Translated Customer Requirements*, which are within the scope of the platform, are identified. Requirements outside the scope of the platform are contained in the *Customer Specific Product Requirements*
Customer Negotiation	Negotiate customer requirements, which fall outside scope of the product line i.e. *Specific Product Requirements*. Practical arguments such as time to market and costs of implementation must be considered before deciding on requirements for implementation. Requirements are allocated to specific development iterations based on customer priority. The output is the *Negotiated Customer Requirements*
Create the Product Requirements	To form the *Product Requirements* using the *Negotiated Customer Requirements* and the *Translated Customer Requirements*, which were within the scope of the platform. The *Platform Requirements* can be used as a baseline
Verify the Product Requirements	The *Product Requirements* are reviewed and verified by the key stakeholders in the project, the *Customer*, *Product Architect and Product Manager*
Define Role and Task Structures	The goal is to define who is responsible for resolving what remaining variability in product derivation to fulfil the Product Requirements. The output is an annotated *Product Requirements* with individual requirements assigned to specific teams and personal

Table 3
Identify and Refine Requirements Tasks.

Task	Purpose
Find and Outline Requirements	The functional and non-functional requirements for the system are specified and scoped by the *Product Analyst*. With every requirement, it must be decided whether to integrate it into the platform or into an individual product [30]
Create the Product Test Cases	Design the *Product Test Cases* for requirements in the *Product Requirements*. Typically, the *Product Tester* uses the *Platform Test Artefacts* as a basis for the creation
Allocate Requirements	The *Product Requirements* are allocated to relevant disciplines, e.g., hardware discipline, algorithms
Create Guidance for Decision Makers	Guidance can be linked into the *Product Requirements*, often to external sources to provide information on the background to a particular decision. Guidance is essential, especially for domain experts like customers and sales people, who are confronted with many, often technical, decisions

the platform and minimise the amount of product specific development required. Table 4 describes the tasks and their purpose.

4.1.4. Develop the Product

The *Develop the Product* activity contains the tasks that implement requirements that could not be implemented through configuration of existing assets. Table 5 describes the tasks and their purpose.

4.1.5. Test the Product

The *Test the Product* activity contains the tasks required to test the product. Table 6 describes the tasks and their purpose.

4.1.6. Management and Assessment

The *Management and Assessment* activity contains the tasks required to oversee product derivation and to provide essential feedback to the platform team. This is an ongoing activity that occurs in parallel to the phased execution of other activities. Table 7 describes the tasks and their purpose.

4.2. Roles

In Pro-PD we identified roles (see Table 8) that represent the different responsibilities, which occur during product derivation: Customer, Product Analyst, Product Architect, Product Developer, Product Manager and Product Tester. These roles are assigned to specific tasks, which create and modify the different artefacts.

Table 4
Derive the Product Tasks.

Task	Purpose
Select Closest Matching Configuration	A previous product configuration is found in the platform that respects the majority of the P*roduct Requirements*. Configuration selection can also help speed up the development process by choosing a previously tested solution especially in instances when two or more configurations can be used
Derive New Configuration	If no *Existing Platform Configuration* can be found, a new one is derived from the *Platform Architecture*. Here the *Product Architect* makes a copy of the latest *Platform Architecture*. The *Product Architect* tailors the *Platform Architecture* to form the product architecture, which is contained in the *Base Product Configuration* and resolves variation points according to the *Product Requirements*
Evaluate Product Architecture	In the *Evaluate Product Architecture* task the *Base Product Configuration*, which contains the instantiated product architecture, is evaluated to see if it meets the specific behavioural and quality requirements of the product at hand. If the product architecture does not satisfy these requirements then the product team derive the architecture again by performing *Select Closest Matching Configuration* or *Derive New Configuration*. The results of the evaluation are collected in the *Platform Feedback* artefact
Select Platform Components	Components are selected from the collection of *Platform Components* for addition to or replacement of components in the *Base Product Configuration*
Platform-Product Integration	The *Base Product Configuration* and the *Selected Platform Components* are integrated. The output is the *Integrated Product Configuration*
Platform-Product Integration Testing	Validates the platform assets for this particular configuration. The integration tests should reuse *Platform Test Artefacts*
Identify Required Product Development	Theoretically at this stage the *Integrated Product Configuration* could satisfy customer requirements 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 is captured in the *Required Product Development* artefact

Table 5
Develop the Product Tasks.

Task	Purpose
Component Development	The source code to implement new functionality or to adapt an existing platform component at product level is developed by the *Product Developer*. These changes are implemented based on the *Required Product Development* artefact. The output is *Developed or Adapted Components* artefact
Component Testing	When a component is built or adapted, initial or tailored versions of a component will need to be tested rigorously through *Component Testing*. For adapted platform components *Platform Test Artefacts* can be used as a basis for the component unit tests
Component Integration	In *Component Integration*, the *Developed or Adapted Components* are integrated into the *Integrated Product Configuration* to create the *Product Release*
Integration Testing	*Integration Testing* then validates the *Product Release*. The integration tests should reuse *Platform Test Artefacts*. This also ensures that no new errors appear due to the integration of *Developed or Adapted Components* into the *Integrated Product Configuration*

Table 6
Test the Product Tasks.

Task	Purpose
Run Acceptance Tests	The *Product Tester* checks the *Product Release* for compliance with the *Product Requirements*. The *Product Tester* does this through using the *Product Test Cases* creating in the *Create the Product Test Cases* task. The majority of the *Product Requirements* will be a subset of the product line functionality. Therefore the *Product Test Cases* contain many reused tests from the *Platform Test Artefacts*

Table 7
Management and Assessment Tasks.

Task	Purpose
Provide Feedback to Platform Team	In the *Provide Feedback to Platform Team* task, feedback is provided to the Platform Manager on core asset usage during the project, how user friendly the platform assets were and areas for improvement within the platform specifically product requirements which should be adopted by the platform. In addition, the product team identifies product specific components that the platform could potentially benefit from through adoption. The *Platform Feedback* artefact is an input to product line evolution
Manage Project	*Product Manager* assesses project status and identifies any blocking issues. Assessment of project status includes overseeing the phased implementation of the product requirements. The phased implementation allows the partial validation of the product with the customer before the derivation is finished. This assists in identifying whether additional development iterations are required. Identify and manage exceptions, problems and risks. Communicate project status and manage stakeholder's expectations

Table 8
Roles and Responsibilities.

Role	Primary Responsibilities
Customer	Customer Negotiation
Product Analyst	Translate Customer Requirements, Find and Outline Requirements
Product Architect	Coverage Analysis, Create the Product Requirements, Verify the Product Requirements, Create Guidance for Decision Makers, Select Closest Matching Configuration, Derive New Configuration, Evaluate Product Architecture, Platform-Product Integration, Component Integration, Identify Required Product Development
Product Developer	Select Platform Components, Component Development, Component Testing
Product Manager	Customer Negotiation, Define Role and Task Structures, Allocate Requirements, Assess Results, Provide Feedback to Platform Team, Manage Project
Product Tester	Create the Product Test Cases, Platform-Product Integration Testing, Component Integration Testing, Run Acceptance Tests

Table 9
Artefacts.

Artefact Type	Artefacts
Software Artefact	Base Product Configuration, Developed or Adapted Components, Existing Platform Configurations, Integrated Product Configuration, New Platform Release, Platform Architecture, Platform Components, Platform Test Artefacts, Product Release, Product Test Cases, Selected Platform Components
Documentation Artefact	Customer Requirements, Customer Specific Product Requirements, Glossary, Negotiated Customer Requirements, Platform Feedback, Platform Requirements, Product Specific Platform Requirements, Product Requirements, Required Product Development, Translated Customer Requirements

4.3. Artefacts

In Pro-PD, an artefact is produced, modified or used by a task within the derivation process. The list of Pro-PD artefacts is listed in Table 9 and classified as software or documentation artefacts.

4.4. Pro-PD as a reference model

Pro-PD is defined at a high level and should not to be used 'as is' but through specialisation. In order to create a working company specific model this process needs to be specialised and a lower level of model abstraction needs to be constructed. Different instantiations of Pro-PD are created using the roles, tasks, activities and artefacts defined. For example, we have demonstrated the adaptability of Pro-PD as a reference model by proposing an Agile instantiation (A-Pro-PD) [31], which is an adapted version of Pro-PD that satisfies the principles of the Agile manifesto [32].

4.5. Pro-PD summary

Pro-PD was developed to meet the needs for process support in product derivation and is structured around five essential activities. Each of these activities contains roles, tasks and artefacts.

5. Establishing Relevance and Research Innovation

As described in Section 3, when developing Pro-PD, we followed four stages of research. Three of these focused on Establishing Relevance and Research Innovation. We now present the high-level design decisions taken during each of these three stages which resulted in the development of Pro-PD as has been presented in Section 4.

5.1. Stage one – literature review and expert opinion workshops

During stage 1, core construction, we initially created an appropriate model skeleton, showing activities for derivation, to organise the overall structure of the model. For the development of the model skeleton the researcher focused on prominent existing approaches within the literature [8,15,16,22,33]. These approaches were selected as they proposed a process overview for product derivation that considers all or almost all of the derivation process. Furthermore, the considered approaches all had a derivation skeleton themselves, i.e. a high level activities. Based on these sources, we identified the following high level activities for Pro-PD V1:

1. Impact Analysis.
2. Reusability Analysis.
3. Component Development and Adaptation.
4. Product Integration and Validation.

Impact Analysis aims to gather *Product Requirements* based on *Customer Requirements* and negotiation with the *Platform Team*. The *Customer Requirements* are external requirements provided by the customer. They represent the starting point of product derivation. The *Product Requirements* are an internal SPL specific artefact created through negotiation with the customer. The *Product Requirements* defines the system that will be built.

Reusability Analysis purports to create a *Partial Product Configuration* based on the *Product Specific Requirements* and by using the available core assets. During *Component Development and Adaptation*, new components are developed (if required) and existing components are adapted to satisfy requirements, which could not be satisfied by configuring existing core assets. Finally, *Product Integration and Validation* aims to integrate the core asset configuration and newly developed components and to validate the integration by performing appropriate testing procedures.

The model was then discussed during the expert opinion workshop. It was noted that in time it would be interesting to consider a more incremental development process (such as Boehm's spiral model [34]). In particular, this would be worthwhile for the customer-specific part of the application; an incremental process would allow us to validate several pieces of the prototypes with the customer before the product is finished. However, while the discussion on an incremental development process was noted, the workshop experts felt that, until further evidence suggested otherwise, a product derivation process model should be represented as linear and not iterative.

After identifying the main model activities during skeleton construction, the specific tasks to fill these activities were defined. For example, looking at the skeleton filling for the *Impact Analysis* activity, we focused the literature review on tasks that contributed to the primary goal of this activity, "to form a set of product specific requirements which satisfy the customer's needs and ensures long term viability of the product line." When two approaches differed in what accounted for an *Impact Analysis* task, the variation was documented. The end result was a documented overview of the variation and commonalities for that particular activity which could be presented to experts.

Specific task attributes were elicited from the literature. The attributes chosen helped to define a systematic process with task, artefacts and roles. The identification of task attributes was of particular benefit when eliciting expert opinion during the workshops. The researcher formulated these task attributes both as short statements and as questions. For example, the attribute 'Purpose' was reformulated into a question 'What is the purpose of the task?' In the case of the *Customer Negotiation* task the discussion would run until a consensus among workshop experts as to the purpose of *Customer Negotiation* was formed. These short statements and questions could be put to the expert panel where they could be
answered with one of the attributes values. The identification of the task attributes was a pre-cursor to deciding the final artefact flow for a particular activity. The task attributes helped structure the process flow for an activity. The output of this stage of the research was Pro-PD V1 [35].

5.2. Stage two – industrial case study: Robert Bosch GmbH

During stage 2, the product derivation process within Robert Bosch GmbH was researched.

Table 10
Summary of stage 2 changes.

Change	Motivation
Modify the waterfall layout of Pro-PD to a iterative structure. Re-organise *Impact Analysis* activity into two activities, *Initiate Project* and *Identify and Refine Requirements*	Workshop participants were critical of the linear representation of Pro-PD. In Bosch product derivation was more iterative. They initiate a product derivation project once, however the requirements for the project were continually identified and refined. The expert panel also provided evidence for an iterative form of product derivation in stage 1 of the research
Added new task *Translating the Customer Requirements* to *Initiate Project* activity	Translating the Customer Requirements into internal platform language is an important Robert Bosch GmbH task for product derivation preparation. This prevented terminology confusion and customer specific description of assets
New artefact *Glossary* input to *Translating the Customer Requirements* task	Robert Bosch GmbH used a customer terminology glossary to assist in translating specific customer terms to platform specific terminology within requirements specifications
Added artefact *Translated Customer Requirements* as output of *Translating Customer Requirements* task	This artefact is an output of the *Translating Customer Requirements* task
Changed *Map Customer Requirements to Platform Features* task name to *Coverage Analysis*	Robert Bosch GmbH described the task of mapping *Customer Requirements* to platform features as *Coverage Analysis*
Replaced artefact *Platform Artefacts* with *Platform Requirements* as input into *Coverage Analysis* task	In Robert Bosch GmbH coverage analysis was a comparison between the *Customer System Requirements Specification* and the *Platform System Requirements Specification*
Added task *Find and Outline Requirements* to *Identity and Refine Requirements* activity	This task was observed within Robert Bosch GmbH. It involves analysis of the product requirements. The functional and non-functional requirements of the system are specified
Added task *Allocate Requirements* to *Identify and Refine Requirements* activity	This task was observed within Robert Bosch GmbH to involve the allocating of requirements from the product system requirements specification to different organisational disciplines within the product team
Renamed *Reusability Analysis* activity to *Derive the Product* activity	The original name was deemed confusing to Robert Bosch GmbH workshop participants. There is no analysis aspect to this activity. The primary purpose of this activity is to configure platform components
Changed the *Product Integration and Validation* activity to two activities: *Product Development* and *Product Testing*.	In Robert Bosch GmbH the product development team performed product integration. Product testing was a separate activity performed by testing specialist
Removed activity *Component Development and Adaptation*	The tasks of this activity are now contained within the other activities. In the *Find and Outline Requirements* task the development scoping decisions are made. The tasks relating to product development are moved to the *Product Development* activity
Added artefact *New Platform Release*. Input to *Component Development* task	Results of observations from platform product synchronisation process pattern
Added activity *Management and Assessment* which includes task *Manage Project*	Robert Bosch GmbH monitored project progress through metric collection, and milestones for development iterations
Removed decision 'Have the customer requirements change?'	Within Robert Bosch GmbH, the product requirements were contractually agreed at this point. If there was a change to the *Product Requirements*, this was captured during the task *Run Acceptance Tests*

5.2.1. Observations from industrial case study

The main observations obtained from the case study on industrial product derivation practices were that Pro-PD V1 should contain:

- Additional Development Disciplines.
- Additional Roles and Tasks.
- Platform-Product Synchronisation.
- Use of Documentation.

The organisational structure in Robert Bosch GmbH for a particular product derivation business unit is broken into three broad disciplines – software, hardware and mechanics. Within each of these disciplines there are further sub-disciplines. This is a reoccurring set-up within SPL companies. Therefore, the product derivation process had to support multi-development disciplines.

The Robert Bosch GmbH product derivation process contained product derivation roles, which were replicated across the independent product sub-discipline teams and platform teams. Moreover, similar roles exist for hardware and mechanics, requiring appropriate communication and task structures. For instance, the allocation of requirements to responsible teams has to consider the various disciplines, sub-disciplines and modules. Pro-PD V1 was modified to recognise these additional roles and tasks.

Documentation is required to drive the product derivation process. It facilitates communication and synchronised development between the product and platform teams, the different hardware, software and mechanical disciplines and also the sub-disciplines. Document is used as a milestone to plot project progress and as a driver to trigger particular tasks. For Pro-PD V1 to prepare for product derivation, this requires a higher degree of granularity for requirements management tasks than originally envisaged. This was observed when the case study company starts a product-specific project. During the early activities, the customer requirements are translated into a set of internal company documents. These documents are processed and augmented during various tasks where requirements are analysed for reuse potential and then assigned to disciplines. Pro-PD V1 was adapted to consider the need for this additional requirements management tasks.

5.2.2. Impact on Pro-PD

Robert Bosch GmbH case study participants reported that while Pro-PD V1 was a reasonably accurate representation of the product derivation process, there was room for improvement.

For example, they found some activities and task naming caused confusion, for instance when describing *Product Requirements* as *Product Specific Requirements* the participants found the use of the term 'specific' caused confusion. The participants also found that the analysis of *Product Requirements* within Pro-PD V1 lacked detail and missed important tasks required for preparing for product derivation.

In Table 10 a summary of the changes to Pro-PD V1 product derivation preparation as a result of the feedback and observations from the Robert Bosch GmbH case study are presented. Implementation of these changes resulted in Pro-PD V2 [36].

Table 11
Summary of stage three changes to Pro-PD.

Change	Motivation
Remove task *Identify Required Component Development*	Identifying product specific development occurs during the initiation of a derivation project within the DOPLER^{UCon} approach and not during the later *Develop the Product activity*
Modify task Find and Outline Requirements	Scoping decisions on required component development and adaptation are decided during the *Find and Outline Requirements* task. This is in line with both Robert Bosch GmbH and DOPLER^{UCon} approaches
Add task *Create Guidance for Decision Makers to Identify and Refine Requirements* activity	In DOPLER^{UCon} arbitrary guidance (e.g. multimedia) can be created for open decisions and then related to product requirements. Guidance linking remaining variability in the product requirements assists in dealing with the complexity associated with representing product line variability
Modify artefact *Product Requirements*, it now contains a link to variability guidance to be used by the *Product Architect* during the *Derive the Product* activity	In DOPLER^{UCon} the arbitrary guidance (e.g. multimedia) created during the task *Create Guidance for Decision Makers* is linked to the *Product Requirements*
Add task *Platform-Product Integration*. This task integrates the *Base Product Configuration* and the selected *Platform Components*	Previously this integration was performed within the *Select Platform Components* task. The selected base configuration is represented in DOPLER^{UCon} by a derivation model. In this model, the platform components are selected by taking decisions. A separate activity integrates the base configuration and selected platform components
Add task *Integration Testing to Derive the Product* activity	Integration testing is performed by the *Product Architect* and occurs during the *Derive the Product* activity
Add task *Provide Feedback to Platform Team* to *Management and Assessment* activity	The DOPLER^{UCon} task *Product Line Evolution* used feedback passed from the product team to analyse newly developed assets and analyse new requirements
Removed task *Identify Required Component Adaptation*	Decisions on required component development and adaptation are now decided during the *Find and Outline Requirements* task
Add artefact *Platform Feedback*. This is an output of the task *Provide Feedback to Platform Team*	Artefact used to transfer platform feedback to the platform team. It is an output of the *Provide Feedback* to *Platform Team* task
Added task *Define Role and Task Structures*. Responsibility for implementation of specific requirements is allocated to team members who hold the roles of *Product Developer and Product Architect*	In the DOPLER^{UCon} task *Define role and Task Structures*, variability is managed by the product manager through assigning responsibility to different members of the product team
Modify task description of *Select Platform Components*	Based on the responsibility defined during *Allocate Requirements* task, responsibility for binding variability is allocated. This is similar to the DOPLER^{UCon} approach – *Define Roles and Responsibilities and Adapt Variability Model* tasks

Table 12
Evaluation of the PLPF Requirements Engineering practice area.

Practice name	Requirements Engineering (RE) PLPF practice area
Application to product derivation	Product line requirements define the products in the product line together with the features of and the constraints on those products. Product Requirements common across the product line are written with variation points that can be filled in or exercised to create product-specific requirements (RE-1). The product line requirements guide the elicitation of the specific requirements for that product (RE-2). In product development, requirements engineering plays a key role in determining the feasibility of producing a particular product as part of the product line (RE-3). You can use a statement of the requirements specific to that candidate product to help estimate the cost of developing the product (RE-4) The production, testing, and deployment of the particular product. Requirements play a role in these activities just as they do for single-system development (RE-5) Product-specific requirements often "grow up" to become product line requirements if they can be slightly generalised or if they pop up in more than one product. That is the primary mechanism for the evolution of software product lines over time (RE-6)
Key criteria of practice area	RE-1: The platform requirements are used as a baseline to create the product requirements RE-2: The product line requirements guide the elicitation of the specific requirements for that product RE-3: In product development, requirements engineering plays a key role in determining the feasibility of producing a particular product as part of the product line RE-4: A statement of the requirements specific to that candidate product to help estimate the cost of developing the product RE-5: The product requirements are input to production, testing and deployment RE-6: Specific Product Requirements are suggested for adoption by the platform
Does Pro-PD satisfy the practice area criteria?	RE-1: In the Task *Create the Product Requirements*, the organisation uses the *Platform Requirements* as a baseline to create the *Product Requirements*. This satisfies RE-1 RE-2: The *Platform Requirements* are used in the primary requirements elicitation tasks, *Coverage Analysis* and *Create the Product Requirements* tasks. The Platform Requirements are used as a guide for *Coverage Analysis* and taken as a baseline for the *Product Requirements* in *Create the Product Requirements*. This satisfies RE-2 RE-3: The task *Coverage Analysis* is at its simplest a comparison between of the *Customer Specific Product Requirements* and the *Platform Requirements*. *Coverage Analysis* identifies the *Customer Specific Product Requirements* that are satisfied by the platform. If the number of *Customer Specific Product Requirements* is above a certain threshold then the feasibility of the product derivation project is considered. This satisfies RE-3 RE-4: The task Customer Negotiation takes a statement of the requirements specific to the customer product (*Customer Specific Product Requirements*) to help estimate the cost of development. This cost of development is a deciding factor during *Customer Negotiation* on the *Customer Requirements* that are selected for implementation. RE-5: Pro-PD uses the *Product Requirements* in production (*Develop the Product*) and *Product Testing* RE-6: In the *Provide Feedback to Platform Team* task, feedback is provided on areas for improvement within the platform, specifically product requirements which should be adopted by the platform. This satisfies RE-5

5.3. Stage three – academic comparative analysis: DOPLER^{UCon}

We undertook an academic comparative analysis, comparing Pro-PD V2 with DOPLERP^{UCon}. This stage of the research furthered the development of Pro-PD. While both Pro-PD V2 and DOPLERP^{UCon} have been developed with different goals, for different purposes, and in different domains, many interesting parallels were still found.

DOPLER^{UCon} was chosen for comparative analysis for a number of reasons. Firstly, the approach was driven by industry needs with the goal to define a user-centred, tool supported product derivation approach. The approach was mainly influenced by a research-industry collaboration with Siemens VAI, therefore, there was a strong practical industry focus in the approach. Secondly, the approach was focused on adaptable tool support usable in practical settings. The approach therefore was suitable in terms of developing an adaptable approach. Thirdly, the approach had academic credibility, as it had been judged an appropriate tool approach to product derivation by peers through publications in leading journals and software conferences e.g. [4,14,19]. Finally, it was designed to be generic, without focusing on a particular organisation or domain. The approach had a strong industry focus and through choosing this case for the comparative analysis, the researcher was performing a type of indirect industrial study.

In DOPLER^{UCon} the *Initiate Project* activity is called configuration preparation and DOPLER^{UCon} supports much of this activity as part of its application requirements engineering activity. DOPLER^{UCon} does not support *Translate Customer Requirements*. The missing support for *Translate Customer Requirements* in DOPLERP^{UCon} can be explained with the differences in customer management. In a collaborative environment, as assumed by DOPLERP^{UCon}, customer requirements are typically delivered in a product line compatible format. However, Pro-PD has identified that the negotiation with the customer is often required.

The activity '*Identify and Refine Requirements*' is partly supported by DOPLER^{UCon}, as captured requirements can be assigned arbitrary types. This can also be used to define whether they are platform or product specific. *Create the Product Specific Test Cases* is assumed to happen but not defined. DOPLER^{UCon} does not consider the '*Allocate Requirements*' task also.

'*Derive the Product*' is focused on the derivation of a product configuration from the product line, i.e., selecting, customising, and integrating reusable assets. In DOPLER^{UCon} this activity is called 'Product Configuration'. The three '*Derive the Product*' tasks are fully or partly supported by DOPLERP^{UCon}. DOPLER^{UCon} performs the tasks *Derive New Configuration* and *Select Platform Components* in parallel. The explicit linkage of components with decisions allows selection of platform components by taking decisions in the base configuration derivation model. The task *Select Closest Matching Configuration* is supported by options within DOPLER^{UCon} to allow derivation begin with an existing configuration.

In DOPLER^{UCon} the activity '*Product Development*' occurs in Application Engineering. The tool supports all four tasks of this activity. The testing tasks of '*Component Unit Testing*', '*Integration Testing*' and '*System Testing*' is assumed to happen but not defined.

The result of stage three of the research was Pro-PD V3 [37]. In Table 11 a summary of the changes made from V2 to V3 is presented along with the primary motivation for these changes.

6. Evaluation

Following the staged development of Pro-PD, we performed an Evaluation, Stage 4 of the research, by comparing it to prominent existing approaches, namely an inter-model evaluation with the SEI Product Line Practice Framework (PLPF) and a systematic analysis with COVAMOF, FAST and PuLSE-I.

6.1. Inter-model evaluation

To evaluate Pro-PD V3, we examined how it satisfies product derivation according to the SEI Product Line Practice Framework (PLPF). The choice of PLPF practices for the evaluation was based on the documented PLPF practice patterns relevant to product derivation, namely the Product Builder and the Essentials Coverage patterns. The relevant PLPF practice areas were: Requirements Engineering; Architecture Definition; Component Development; Software System Integration; Customer Interface Management; Structuring the Organisation; Testing; Operations; Architecture Evaluation.

Using the practice descriptions, a set of criteria was identified and labelled for each practice e.g. the first criteria for the testing practice was labelled 'T-1'. Table 12 shows the evaluation for the Requirements Engineering practice area. The other evaluations are available in [38].

The practice summary in Table 13 presents the results of the evaluation for each of the nine practices.

6.1.1. Results of the inter-model evaluation

From the above table (see Table 13), it can be seen that Pro-PD V3 completely satisfies six of the nine relevant practices.

Pro-PD V3 only partially satisfies the Operations practice as it does not consider software maintenance, which is outside of the scope of the research. Pro-PD V3 only partially satisfies the Testing practice. According to the Testing practice there should be verification of artefacts between the different activities of Pro-PD V3. Therefore in the *Initiate Project* and *Identify and Refine Requirements* activities, a new task *Verify the Product Requirements* is required during which the new product requirements and the complete product requirements are reviewed by the *Customer*, *Product Architect* and *Product Manager*.

Pro-PD V3 does not satisfy the Architecture Evaluation practice as there is no separate testing of the *Product Architecture* in the Derive the Product activity. Therefore, an *Evaluate Product Architecture* task is included. The task evaluates the instantiated *Product Architecture* to see if it meets the specific behavioural and quality requirements of the product at hand.

Table 13
Practice summary results.

Practice	Pro-PD V3 satisfied the criteria?
Requirements Engineering (RE-1, RE-2, RE-3, RE-4, RE-5, RE-6)	All criteria are satisfied
Architecture Definition (AD-1, AD-2, AD-3)	All criteria are satisfied
Architecture Evaluation (AE-1, AE-2)	No criteria are satisfied
Component Development (CD-1, CD-2, CD-3, CD-4, CD-5, CD-6)	All criteria are satisfied
Testing (T-1, T-2, T-3, T-4, T-5)	T-1 is not satisfied. All other criteria are satisfied
Software System Integration (SSI-1, SSI-2, SSI-3, SSI-4)	All criteria are satisfied
Operations (O-1, O-2, O-3, O-4)	O-2 is not satisfied. All other criteria are satisfied
Customer Interface Management (CIM-1, CIM-2, CIM-3)	All criteria satisfied
Structuring the organisation (SO-1, SO-2, SO-3, SO-4, SO-5)	All criteria satisfied

Table 14
Analysis of COVAMOF, PuLSE-I and FAST for Pro-PD support.

Context	Questions
	What aspects of product derivation does the approach cover?
COVAMOF	Main focus is on product configuration; only partly covers preparing for derivation and additional development and testing
PuLSE-I	Covers preparing for derivation, product configuration, as well as additional development and testing
FAST	Covers requirements elicitation and analysis, product configuration, and additional development and testing
	What is the starting point for the approach?
COVAMOF	Creating a ''product entity'' based on customer requirements.
PuLSE-I	Customer or management has a product request that falls under the scope of the product line
FAST	Final product requirements are established by contract or informal discussion of customer requirements. An application engineer then tries to understand and validate customer requirements and their relation to product line models
User	
	Which stakeholders are addressed by the approach and how?
COVAMOF	Engineers are the target group of the approach. They are (tool) supported to enable iterative derivation of a product based on *customer* requirements
PuLSE-I	Customers and management are explicitly considered as providing input in form of product requests. Project management is also addressed with a project plan artefact. Derivation activities are performed by dedicated application engineers
FAST	Customers are involved in defining the requirements and in validating the derived product. Application engineers and so-called ''producers'' define models from which the application is then generated
Contents	
	What activities/steps/sub-processes does the approach define to accomplish product derivation?
COVAMOF	Product definition: Defining customer and product name Product configuration: Binding of variation points based on customer requirements. Product realisation: Tool-based translation of the configuration of the variability model to a configuration of an executable product, e.g., by setting compiler flags or creating make files Product testing: Determining whether the product meets the customer requirements and deciding whether an additional iteration (product configuration/realisation/testing) is required
PuLSE-I	Plan for the product line instance (the product): Determine whether all characteristics of the required product are covered by the product line Create project plan: Define what is product-specific and what can be fulfilled by the product line Instantiate and validate product line model: Incrementally resolve decisions defined in the product line model (representing variation points) Instantiate and validate reference architecture: Instantiate variability to derive an ''intermediate architecture'' from the product line, validate, and then modify if necessary Product construction: Lower level design, implementation, and testing based on reusable assets
FAST	Determine requirements: The customer identifies or refines the requirements Create application model: The application engineer represents the product requirements as an ''application model'' Analyse model: The application model is analysed to determine whether it satisfies the product requirements Generate application: Generation tool(s) are created and used to generate code and documentation based on the application model Develop product: Engineers develop any custom parts that cannot be generated manually and integrate them with the application Verify integrated application: The customer either accepts the application or the process returns to start
	What artefacts are created and managed by the approach?
COVAMOF	Product entity: Created in product definition with selected variants
PuLSE-I	Detailed project plan: Output of ''plan for product line instance'' activity Requirements specification: Output of ''instantiate and validate product line model'' Product architecture: Output of ''instantiate and validate reference architecture'' Product ready for delivery: Output of ''product construction''; comprising specification, architecture, and code Product configuration: Output of all aforementioned activities; comprising domain decision model instance, architecture decision model instance, and low level configuration
FAST	Application model: Created by application engineers based on product requirements Product: Deliverable code for the application which is typically generated from the application model using generation tools Customer documentation: Might be generated from the application model
	How are the preparing for derivation tasks Initiate Project and Identify and Refine Requirements supported by the approach?
COVAMOF	Partly supported: Customer requirements are assumed to be available and phrased so that engineers can perform product configuration and testing based on these requirements (no explicit specification and translation of customer requirements). Mapping of customer requirements to the base configuration is not part of preparing for derivation but rather assumed to be done manually by engineers during product configuration. COVAMOF provides partial support for creating the product-specific requirements: a list of characteristics that the final system will have is created or reused if the requested product is fully within the scope of the product line. COVAMOF assumes engineers to do the work supported by COVAMOF-VS. It does not consider defining role and task structures. Creating derivation guidance is not considered part of product derivation but may be done in variability modelling by creating variability views
PuLSE-I	Partly supported: During the ''plan for product line instance'' activity a detailed project plan is created as preparation for derivation. Customer requirements (product request) are assumed to be available and phrased so that they can be used to determine whether the requested product is inside the scope of the product line. Overlaps are evaluated and required system-specific developments are defined The output in PuLSE-I is ''a set of characteristics upon which the customer (or the marketing) and the developers have agreed''. Defining a base configuration is also supported: during ''plan for product line instance'', a ''list of characteristics that the final system will have'' is created or reused if the requested product is fully within the scope of the product line. PuLSE-I as such defines the involved stakeholders and their roles and tasks, however, on a rather high-level. Creating derivation guidance is assumed to be provided by the product line decision model and no explicit creation of additional guidance is part of the approach
FAST	Partly supported: During activity ''determine requirements'' the customer identifies the product requirements. The product requirements are the basis for the created application model. The application model is then analysed to determine whether it satisfies the product requirements. This supports the activities specify (and translate) customer requirements, define base configuration, and map customer requirements. FAST provides no explicit support for activities define role and task structures and create derivation guidance
	How is the Derive the Product activity supported by the approach?
COVAMOF	Fully supported: In the task ''Derive new configuration'', a new product entity is created in the COVAMOF variability model. Engineers select variants by specifying values at variation points. COVAMOF-VS supports this with it's configure mode where additional configuration information about the product at hand is shown in variability views. An inference and a validation engine automate this process. A partial product configuration is iteratively created, by

(*continued on next page*)

Table 14 (*continued*)

Context	Questions
	selecting more and more variants for the product entity. Each selected variant can have "effectuation actions" that can be executed to realise the product (product realisation activity of COVAMOF), e.g., by creating make files or settings files
PuLSE-I	Fully supported: PulSE-I supports selecting a subset of existing components as part of the PuLSE-I activity instantiate and validate product line model where decisions are resolved through the customer. Creating a partial product configuration is part of PuLSE-I activities instantiate and validate reference architecture (instantiate variabilities to create an "intermediate architecture" from the product line) and product construction (low-level configuration based on reusable product line assets)
FAST	Fully supported: In the "generate application" activity, generation tools are used to generate application code and documentation based on the application model. This is defined support for the select assets and create partial product configuration activities
	How are the Develop the Product and Test the Product activities supported by the approach?
COVAMOF	Partly supported: System testing is fully supported through the COVAMOF product testing activity. This determines whether the product meets both the functional and the non-functional requirements. COVAMOF however defines no explicit support for component development, component testing, component integration with partial product configurations, or integration testing but assumes this to happen, just like DOPLER^{UCon} does
PuLSE-I	Fully supported: Part of the PuLSE-I activity product construction is the implementation of non-existing product line assets and of product-specifics. This includes testing (unit testing, integration testing, and acceptance testing). All this is conducted in several iterations under consideration of existing reusable product line assets. This supports component development and component testing, component integration and integration testing, as well as system testing
FAST	Fully supported: FAST provides full support for additional development and testing. In the "develop product" activity, any custom parts of the application that cannot be generated are developed and integrated with the generated product. In the "verify integrated application" activity, the customer either accepts the application or the process returns to start
	What activities/sub-activities does the approach include that are not covered by the defined key activities/sub-activities?
COVAMOF	Our key activities include all activities defined by COVAMOF
PuLSE-I	Apart from activities that are considered as application engineering and not product derivation (i.e., system delivery and maintenance), PuLSE-I also includes several feedback loops to other PuLSE phases (e.g., PuLSE-Eco with its scoping activities) belonging to domain engineering. Such feedback loops are currently not considered by our key activities
FAST	Apart from activities that are considered as application engineering activities and not product derivation activities (i.e., product delivery and support), our key activities include all activities defined by FAST
Validation	
	Has the approach been validated in practical industrial case studies?
COVAMOF	COVAMOF has been validated in three industrial product lines [39]; two of them are reported in more detail in [5].[40] report on an industrial validation of the COVAMOF framework. They focus on showing how the use of COVAMOF (supported by COVAMOF-VS) reduced the number of iterations required to derive products from a product line of their industry partner. They also compare results of the use of their framework and tool by "non-experts" vs. the use by "experts"
PuLSE-I	The PuLSE approach has been applied in case studies, for example [41]. [9] claim the approach to have been used in various contexts
FAST	Several application examples are presented in [10]. The authors claim that FAST has been applied for several real-world systems

6.2. Analysis of existing approaches

For the second part of the evaluation, support for Pro-PD V3 product derivation preparation tasks within existing approaches is analysed and shown in Table 14.

There is no activity we defined which has no support. Of course, how the activities are supported differs from approach to approach and depends on both the focus and the scope of the approach. For example, the COVAMOF approach is tool-supported and concentrates primarily on product configuration. FAST has a larger scope but mainly concentrates on automated derivation, i.e., describing products in an application modelling language and then using generators based on that language to create products. PuLSE-I has the largest scope of the three approaches but does not focus on tool support.

Pro-PD V3 considers all preparation tasks defined by COVAMOF. Apart from PuLSE-I activities that are considered as application engineering and not product derivation (i.e., system delivery and maintenance), PuLSE-I also includes several feedback loops to other PuLSE phases (e.g., PuLSE-Eco with its scoping activities) belonging to domain engineering. Such feedback loops are currently not considered by Pro-PD V3. While there are tasks in FAST that are considered as application engineering activities and not product derivation activities (i.e., product delivery and support), our key activities include all activities defined by FAST.

The preparation for derivation activities (*Initiate Project* and *Identify and Refine Requirements*) is only partly supported by all three approaches. Our research has demonstrated that preparing for derivation is an important activity and has to be at least closely related to product derivation. We have experienced that a lack of support for preparing derivation is one of the main reasons that product derivation often fails in practice [6,37]. A special focus has to be the definition of roles and tasks (the task *Define Role and Task Structures*) for product derivation stakeholders as well as the creation of guidance for domain experts (the task *Create Guidance for Decision Makers*). However, approaches such as COVAMOF, recommend that *Create Guidance for Decision Makers* could be performed separate from the product derivation effort.

The *Derive the Product* activity is fully supported by all three approaches in different ways. The focus is clearly on automating the derivation of products as far as possible to ensure return on investment for adopting a product line approach and to make efficient and effective product derivation possible.

All three approaches perceive derivation as an iterative process. COVAMOF and FAST include explicit activities (product testing, verify integrated application) for deciding whether to deliver or perform additional iterations. The key activities we defined also strongly focus on testing and on the iterative nature of product derivation.

PuLSE-I is not an isolated description of product derivation but has many dependencies to other parts of the overall PuLSE product line methodology. It would also make sense to relate our key activities to domain engineering activities and ensure there is a "fast feedback loop" [42]. For instance PuLSE-I sends requests to adapt the scope of the product line to be able to address new customer requirements. Pro-PD by comparison is far more isolated from the domain engineering activities. It has two primary feedback loops to the platform team, In the *Coordinate with Platform Team* task, feedback is provided to the platform team on core asset usage during the project. In the *Identify and Refine Requirements* activity, the platform team receives the platform software requirements containing the required extensions to the existing platform in order to facilitate the new product specific requirements.

The analysis of the three approaches [4] (COVAMOF, FAST and PuLSE-I) illustrates that the activities defined by other approaches are considered by Pro-PD. However greater feedback to domain engineering activities is required. We therefore claim that Pro-PD should be considered when developing or evaluating a product derivation approach. However, how the tasks and activities are implemented in an approach strongly depends on the domain and context. In some cases it might be best to define a domain-specific derivation approach. Some tasks may simply not make sense in particular contexts. The process we define can be used as a checklist when defining, adapting, or evaluating a product derivation approach for a certain domain, context, or problem.

7. Threats to validity

7.1. Risk of case study bias

All qualitative research suffers from the risk of bias and multiple interpretations of data. Data collected during the various research stages was analysed objectively in order to ensure minimisation of bias. Despite this, results taken from the data will be influenced by the inclusion of the Robert Bosch GmbH case study. However, the focus on published research during the development of the model, along with the subsequent academic comparative analysis and evaluation, has given us the opportunity to minimise existing case study bias.

7.2. Handling refinements

Each stage of the research provided the basis for the revision or refinement of Pro-PD. A major challenge when making iterations was the evaluation of different suggestions with respect to each other. For example, before a correction was integrated it had to be determined whether the proposal could be characterised as being universally valid or whether it was tied to a specific context and therefore not suitable for model refinement. As far as possible, modifications made were supported by previously published literature or expert opinion. Furthermore, improvement suggestions made by different persons were sometimes contradictory. There were two options to resolve these situations. First, one proposal was chosen over another if the source was deemed to be of a better quality, either through its experience or the location of the source. This evaluation was conducted by the researcher, and involved a degree of researcher interpretation as to the quality of the various sources. The alternative approach was to consider both suggestions and integrate them both into the model.

8. Conclusion

In response to a need for methodological support for product derivation, the authors identified the following research objective: *To define a systematic process which will provide a structured approach to the derivation of products from a software product line based on a set of tasks, roles and work artefacts.* To meet this objective, we developed Pro-PD (Process Model for Product Derivation). Pro-PD was iteratively developed and evaluated through four research stages involving academic and industrial sources.

When commencing the research, we identified three limitations to current approaches; firstly, lack of a defined flow of artefacts; secondly, no definition of roles and responsibilities; and thirdly, no provision of process support. Through the development of Pro-PD, we have sought to address each of these.

To overcome the limitation, lack of defined flow of artefacts, Pro-PD describes the usage and flow of specific artefacts through the product derivation process. This was observed in the Robert Bosch GmbH industrial case study where documentation was used to drive the product derivation process. These and other observations on artefact flow were modelled in Pro-PD.

It was clear in the early stages of our research that the variety of roles and responsibilities for product derivation could not be undertaken by a single professional group – the engineers (as in [5]). Pro-PD defines different roles and their responsibilities. For example, requirements responsibility is assigned to specific roles and personnel in the *Allocate Requirements* task (see Table 3).

The third limitation was the lack of process support. Pro-PD is a process model defining tasks, artefacts and roles. It is evidence-based, having being developed through industry input. In addition, it is in line with product derivation practice as defined by the Software Engineering Institute's PLPF. In particular, Pro-PD provides systematic support for product derivation preparation. A lack of support for preparing derivation is one of the main reasons that product derivation often fails in practice [37].

In this paper, we have documented the evidence for the construction of Pro-PD and the design decisions made as a result of this evidence. We have identified and elaborated on the essential tasks for product derivation, developing and evaluating a Process Model for Product Derivation, Pro-PD. For SPL to become a mature engineering discipline it needs to define an evidence-based methodology and this is a step in this direction.

The tasks we present are generic and in some situations domain-specific tasks will be required. Therefore, further research is needed to support the definition of when and how tasks are tailored to specific contexts, domains or organisation. It would also be interesting to consider a more rigorous Information Systems Development approach which considers the interplay between platform and product development.

Acknowledgements

This work is supported by IRCSET under Grant Number RS/06/167 and by Science Foundation Ireland through Lero – the Irish Software Engineering Research Centre under Grant Numbers 03/CE2/I303_1 and 10/CE/I1855. This work was partially supported by the National Institute of Science and Technology for Software Engineering (INES[1]), funded by CNPq and FACEPE, grants 573964/2008-4 and APQ-1037-1.03/08 and CNPq grants 305968/2010-6, 559997/2010-8, 474766/2010-1 and FAPESB.

References

[1] P. Clements, L. Northrop, Software Product Lines: Practices and Patterns, Addison-Wesley Longman Publishing Co., Boston, MA, USA, 2001.
[2] L. Hotz, A. Gunter, T. Krebs, A knowledge-based product derivation process and some ideas how to integrate product development, in Proc. of Soft. Variability Management Workshop. Groningen, The Netherlands, 2003.
[3] M.L. Griss, Implementing product-line features with component reuse, in: ICSR-6: Proc. of the 6th Int. Conf. on Software Reuse, Springer-Verlag, London, UK, 2000, pp.137–152.
[4] R. Rabiser, P. O'Leary, I. Richardson, Key activities for product derivation in software product lines, J. Syst. Softw. 84 (2) (2010) 285–300.
[5] S. Deelstra, M. Sinnema, J. Bosch, Product derivation in software product families: a case study, J. Syst. Softw. 74 (2) (2005) 173–194.
[6] R. Rabiser, P. Grünbacher, D. Dhungana, Supporting product derivation by adapting and augmenting variability models, in: 11th Int. Software Product Line Conf., Kyoto, Japan, 2007.
[7] J. Bayer, O. Flege, P. Knauber. R. Laqua, D. Muthig, K. Schmid, T. Widen, J.-M. DeBaud, PuLSE: a methodology to develop software product lines, in: Proc. of Symposium on Software Reusability, ACM, Los Angeles, California, United States, 1999.
[8] J. Bayer, C. Gacek, D. Muthig, T. Widen, PuLSE-I: Deriving instances from a product line infrastructure, in: 7th IEEE Int. Conf. and Workshop on the Engineering of Computer Based Systems, Edinburgh, UK, 2000.
[9] C. Atkinson, J. Bayer, Muthig, Component-based product line development: the KobrA approach, in: Proc. of the First Conf. on Software Product Lines: Experience and Research Directions, Kluwer Academic Publishers, Denver, Colorado, United States, 2000.

[10] D. M Weiss, C.T.R. Lai, Software Product Line Engineering: A Family-based Software Development Process, first ed., Addison-Wesley Professional, 1999.
[11] S.D. Kim, H.G. Min, J.S. Her, S.H. Chang, DREAM: A practical product line engineering using model driven architecture, in: Proc. of the Third Int. Conf. on Information Technology and Applications (ICITA'05), Washington, DC, USA, 2005, pp. 70–75.
[12] M. Sinnema, S. Deelstra, J. Nijhus, J. Bosch, Modeling dependencies in product families with COVAMOF, in: 13th Annual IEEE Int. Conf. and Workshop on the Engineering of Computer Based Systems (ECBS 2006), Potsdam, Germany, 2006.
[13] J.D. McGregor, Preparing for Automated Derivation of Products in a Software Product Line, Carnegie Mellon Software Engineering Institute, 2005.
[14] R. Rabiser, A User-Centered Approach to Product Configuration in Software Product Line Engineering, in Christian Doppler Laboratory for Automated Software Engineering, PhD Thesis, Institute for Systems Engineering and Automation, Johannes Kepler University, Linz, 2009.
[15] G. Chastek, J.D. McGregor, Guidelines for Developing a Product Line Production Plan, in Product Line Practice Initiative, Carnegie Mellon Software Engineering Institute, Pittsburgh, PA, 2002.
[16] K. Kang, S. Kim, J. Lee, K. Kim, G.J. Kim, E. Shin, FORM: a feature oriented reuse method with domain specific reference architectures, Ann. Softw. Eng. 5 (1) (1998) 143–168.
[17] K. Kang, S. Cohen, J. Hess, W. Nowak, S. Peterson, Feature-Oriented Domain Analysis (FODA) Feasibility Study, Carnegie Mellon Software Engineering Institute, Pittsburgh, PA, USA, 1990.
[18] N. Guelfi, G. Perrouin, A Flexible Requirements Analysis Approach for Software Product Lines, in Requirements Engineering: Foundation for Software Quality, Springer, Berlin/Heidelberg, 2007. 78–92.
[19] R. Rabiser, D. Dhungana, Integrated support for product configuration and requirements engineering in product derivation, in: 33rd EuroMicro Conf. on Software Engineering and Advanced Applications, 2007.
[20] D. Rombach, Fraunhofer: the German model for applied research and technology transfer, in: Proc. of the 22nd Int. Conf. on Software engineering (ICSE2000), ACM, Limerick, Ireland, 2000.
[21] G. Böckle, J.B. Muñoz, P. Knauber, C.W. Krueger, J.C.S. do Prado Leite, F. van der Linden, L.M. Northrop, M. Stark, D.M. Weiss, Adopting and institutionalizing a product line culture, in: Proc. of the Second Int. Conf. on Software Product Lines (SPLC 2002), Springer Verlag, San Diego, CA, USA, 2002.
[22] M. Sinnema, S. Deelstra, P. Hoekstra. The COVAMOF derivation process, in: Proc. of the 9th Int. Conf. on Software Reuse (ICSR 2006), Turin, Italy, Springer, Berlin Heidelberg, 2006.
[23] F. Ahlemann, H. Gastl, Process model for an empirically grounded reference model construction, in: P. Fettke, P. Loos (Eds.), Reference Modeling for Business Systems Analysis, IGI Publishing, 2006.
[24] P. Fettke, P. Loos, Reference Modeling for Business Systems Analysis, IGI Publishing, 2006.
[25] M. Rosemann, R. Schütte, Multi-perspective reference modelling, in: J. Becker, M. Rosemann, R. Schütte (Eds.), Referenzmodellierung. State-of-the-art und entwicklungsperpektiven, Physica-Verlag, Heidelberg, 1999, pp. 22–44.
[26] B. Schlagheck, Object-oriented Reference Models for Process and Project Controlling – Foundation Construction Fields of Application, Deutscher Univ. Verlag, Wiesbaden, 2000.
[27] M. Hammersley, R. Gomm, P. Foster, Case Study Method: Key Issues, Key Texts, Sage Publications, London, 2000.
[28] The SPLC Product Line Hall of Fame, <http://www.splc.net/fame.html> (cited 03.02.11).
[29] M. Matinlassi, Comparison of software product line architecture Design methods: COPA, FAST, FORM, KobrA and QADA, in: Software Engineering, ICSE 2004. Proc. 26th Int. Conf. on. 2004, EICC, Scotland, UK, 2004.
[30] A. Birk, G. Heller, I. John, K. Schmid, T. von der Massen, K. Muller, Product line engineering: the state of the practice, IEEE Softw. 20 (6) (2003) 52–60.
[31] P. O'Leary, F. McCaffery, S. Thiel, I. Richardson, An agile process model for product derivation in software product line engineering, J. Softw. Mainten. Evolut. (2010).
[32] Beck, K., et al. Manifesto for Agile Software Development, 2001 1/3/2006, <http://agilemanifesto.org/> (cited 10.09.10).
[33] K. Pohl, G. Böckle, F.v.d. Linden, Software Product Line Engineering: Foundations Principles, and Techniques, Springer, Heidelberg, 2005.
[34] B.W. Boehm, A spiral model of software development and enhancement, IEEE Comput. 21 (5) (1988) 61–72.
[35] P. O'Leary, M. Ali Babar, S. Thiel, I. Richardson, Product derivation process and agile approaches: exploring the integration potential, in: Proc. of 2nd IFIP Central and East European Conf. on Software Engineering Techniques. Poznań, Poland, Wydawnictwo Nakom, 2007.
[36] P. O'Leary, S. Thiel, G. Botterweck, I. Richardson, Towards a product derivation process framework, in: 3rd IFIP TC2 Central and East European Conf. on Software Engineering Techniques CEE-SET. Brno (Czech Republic), 2008.
[37] P. O'Leary, R. Rabiser, S. Thiel, I. Richardson, Important issues and key activities in product derivation: experiences from two independent research projects, in: Software Product Line Conf. San Francisco, CA, USA, 2009.
[38] P. O'Leary, S. Thiel, I. Richardson, Towards a Product Derivation Process Reference Model for Software Product Line Organisations, in: Department of Computer Science and Information Systems, University of Limerick, Limerick, 2010, p. 277.
[39] M. Sinnema, S. Deelstra, J. Nijhuis, J. Bosch, COVAMOF: a framework for modeling variability in software product families, in: Proc. 3rd Int. Conf. Software Product Lines (SPLC 04), San Diego, CA, 2004.
[40] M. Sinnema, S. Deelstra, Industrial validation of COVAMOF, J. Syst. Softw. 81 (4) (2008) 584–600.
[41] K. Schmid, I. John, R. Kolb, G. Meier, Introducing the PuLSE approach to an embedded system population at Testo AG, in: Proc. of the 27th Int. Conf. on Software engineering, St. Louis, MO, USA, ACM, 2005.
[42] W. Heider, R. Rabiser. Supporting evolution of product lines through rapid feedback from application engineering, in: Proc. 4th Int. Workshop on Variability Modelling of Software-intensive Systems (VaMoS 2010), University of Duisburg Essen, Linz, Austria, ICB-Research Report No. 37, 2010.