Information and Software Technology 52 (2010) 324–346

Contents lists available at ScienceDirect

Information and Software Technology

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

Requirements for product derivation support: Results from a systematic literature review and an expert survey

Rick Rabiser [a,*], Paul Grünbacher [a], Deepak Dhungana [b,1]

[a] Christian Doppler Laboratory for Automated Software Engineering, Johannes Kepler University, Altenberger Str. 69, 4040 Linz, Austria
[b] Lero – The Irish Software Engineering Research Centre, University of Limerick, Limerick, Ireland

A R T I C L E   I N F O

Article history:
Received 16 July 2009
Received in revised form 5 November 2009
Accepted 5 November 2009
Available online 11 November 2009

Keywords:
Product derivation
Software product line
Product line engineering
Systematic literature review

A B S T R A C T

Context: An increasing number of publications in product line engineering address product derivation, i.e., the process of building products from reusable assets. Despite its importance, there is still no consensus regarding the requirements for product derivation support.
Objective: Our aim is to identify and validate requirements for tool-supported product derivation.
Method: We identify the requirements through a systematic literature review and validate them with an expert survey.
Results: We discuss the resulting requirements and provide implementation examples from existing product derivation approaches.
Conclusions: We conclude that key requirements are emerging in the research literature and are also considered relevant by experts in the field.

1. Introduction and motivation

There is a clear trend away from single systems to product lines in software engineering [1,2]. More and more organizations adopt software product lines to leverage extensive reuse in development and to deal with many challenges of today's software development [1,3]. Software product line engineering (SPLE) involves the development, management, and actual use of product lines to build software systems [4]. During *domain engineering* the variability and commonalities of diverse reusable assets such as software components, documentation, or test cases are defined, typically in form of variability models. A significant body of research is available on approaches and notations for variability modeling and management, e.g., [5–7]. During *application engineering* concrete products are built based on the reusable assets. Product derivation is a key activity in application engineering and addresses the selection and customization of assets from the product line [8].

Compared to the vast amount of research results on developing and modeling product lines, only few approaches and tools are available for product derivation. Research has so far focused more on how to scope, define, and develop product lines rather than on how to effectively utilize them. However, the underlying assumption in SPLE is that "*the investments required for building the reusable assets during domain engineering are outweighed by the benefits of rapid derivation of individual products*" [8]. This assumption might not hold if inefficient product derivation diminishes the expected gains. A number of publications discuss the difficulties associated with product derivation. For instance, Hotz et al. [9] describe the process as "*slow and error prone even if no new development is involved*". Griss [10] 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, according to Deelstra et al. [8] the derivation of individual products from shared software assets is still a time-consuming and expensive activity in many organizations.

An increasing number of publications, conference tracks, and workshops in recent years show the growing interest of researchers and practitioners in product derivation. Unfortunately, there is still no clear picture regarding the requirements for product derivation support. For instance, there is no generally accepted reference model for product derivation. Available product derivation approaches and tools have been developed fairly independently to address requirements in different product line contexts or domains [11]. Some approaches apply model-driven development techniques; others are merely a collection of guidelines. Yet other approaches provide a high-level methodology or process framework. Also the tool landscape is still very diverse.

A systematic survey and analysis of existing product derivation approaches is still lacking. A clear definition of requirements for product derivation support would be useful both for building and

* Corresponding author. Tel.: +43 (0) 732 2468 8873; fax: +43 (0) 732 2468 8878.
E-mail addresses: rabiser@ase.jku.at (R. Rabiser), gruenbacher@ase.jku.at (P. Grünbacher), deepak.dhungana@lero.ie (D. Dhungana).
[1] Tel.: +353 61 233704; fax: +353 61 213036.

0950-5849/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.infsof.2009.11.001

enhancing product derivation approaches and for evaluating existing ones. Understanding the requirements for product derivation support also would help companies introducing a product line approach in their own development environment.

The goals of this research are thus to identify and define key requirements for product derivation support and to understand their relative importance. Our research method considers both the state-of-the-art in product derivation and expert knowledge from practitioners and academics. More specifically, we investigate three research questions in this paper:

- *RQ1: What are key requirements for product derivation support in software product line engineering*? We systematically identify and analyze these requirements using a systematic literature review [12].
- *RQ2: What is the relevance and relative importance of the requirements*? We analyze the relative priority of these requirements by involving SPLE experts.
- *RQ3: How can the requirements be implemented*? We investigate how existing SPLE approaches and tools realize the requirements and provide implementation examples.

After a short introduction to product derivation we give an overview of our research method. We then present the process and results of the systematic literature review we conducted to get an overview of existing product derivation approaches. The results of the review reveal that almost half of the selected publications stress the need for product derivation support. A much smaller number of publications (14%) describe concrete support. This seems to indicate that despite increasing interest and importance, product derivation is not yet the main focus of the product line research community.

Based on the results of our systematic literature review, we developed an initial definition of requirements for product derivation support. To increase confidence about their relevancy and relative importance, we conducted a survey among experts at two relevant international events on product lines (a conference and a workshop). The results of the survey were used when defining six requirements for product derivation support. For each of these requirements we present implementation examples demonstrating their realization. We also provide references to existing literature on product derivation identified during the systematic literature review.

It was not our intention to develop an assessment model for product derivation. We believe that the area of product derivation is still not mature enough to develop such a model. However, our findings structure the area of concern and might provide a good starting point for developing such an assessment model in the future. We hope that both researchers in product derivation as well as practitioners adopting SPLE in their own organization can benefit from the results of this research. This paper might further be useful for software engineering researchers considering conducting a systematic literature review.

2. Product derivation in application engineering

The ultimate process in SPLE is "*the activity of turning out products*" [3]. Products bring the return on investment needed to operate a product line economically. The investments for building up a product line have to be outweighed by the benefits of rapid derivation of customized products during application engineering [8]. "*The main goal of application engineering is to derive a software product line application by reusing as many domain artifacts as possible. This is achieved by exploiting the commonality and variability of the product line established in domain engineering*" [4]. "*During application engineering individual, customer-specific software products are ideally being developed by selecting and configuring shared assets resulting from domain engineering*" [13].

Fig. 1 depicts a high-level application engineering process. The upper white vertical arrows represent the product derivation process of selecting and customizing reusable assets during application engineering. The lower white arrows indicate deployment activities necessary to arrive at a final product (e.g., deploying and integrating new components developed to address customer requirements with the existing derived components).

The application engineering process in SPLE involves requirements engineering, design, implementation, and testing. However, in contrast to single-system software engineering, each of these processes needs to consider the existing reusable assets and their variability to effectively utilize the product line. Product derivation is about selecting and customizing shared assets during application engineering. It is *the process of making decisions to select a particular product from a product line and to customize it for a particular purpose*. In product derivation, the variability provided by the product line is communicated to the users of the product line (e.g., customers, domain experts, sales people, project managers, developers). Based on customers' requirements, variants are selected from the product line thus resolving the available variability. While sales people, project managers, or architects make high-level choices, technical product configuration is often performed by engineers. In practice, product derivation is rarely a sequential process and several iterations are necessary for eliciting customer requirements and resolving variability.

Product derivation faces a number of challenges: Software product lines are inherently big and complex. Communicating variability to heterogeneous stakeholders is thus challenging and tedious. The knowledge required for derivation is often distributed in the heads of different people who might be unavailable during derivation. Furthermore, a product line's variability is typically documented in rather complex models defining possible choices among reusable assets and diverse interdependencies. Utilizing such models in actual projects is often not straightforward and intuitive. Also, it is hardly possible to satisfy customers' requirements solely by reusing existing assets from the product line. Instead, customers typically articulate new requirements not yet covered by the product line that require additional development effort. Managing such "deviations from the standard" is thus an important task in product derivation.

3. Research method

An increasing number of publications, conference tracks, and workshops in recent years show the growing interest of both researchers and practitioners in product derivation. Unfortunately, there is still no clear picture regarding the requirements for product derivation support. Our research method aimed at analyzing the available literature on product derivation research and at involving experts in the field. After an initial analysis of existing literature in 2006, we conducted the systematic literature review between November 2007 and February 2008. We developed an initial definition of the requirements based on which we conducted expert surveys to validate and refine the requirements. Finally, we mapped the requirements with existing approaches and tools identified in the systematic review and aggregated our results. Fig. 2 depicts an overview of our research process comprising five main activities:

3.1. Conduct systematic literature review (cf. Section 4)

We performed a *systematic literature review* (SLR) [12,14] to identify research issues in product derivation investigated by the

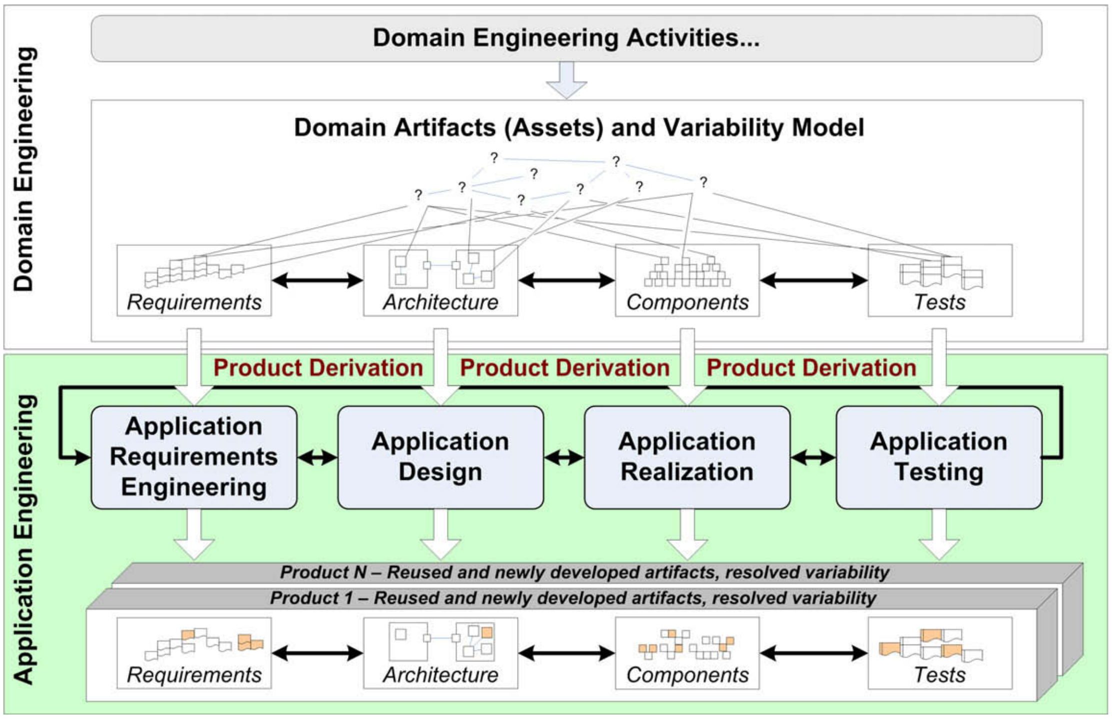
Domain Engineering
Domain Engineering Activities...
Domain Artifacts (Assets) and Variability Model
Requirements
Architecture
Components
Tests
Product Derivation
Product Derivation
Product Derivation
Application Engineering
Application Requirements Engineering
Application Design
Application Realization
Application Testing
Product N – Reused and newly developed artifacts, resolved variability
Product 1 – Reused and newly developed artifacts, resolved variability
Requirements
Architecture
Components
Tests

Fig. 1. SPLE processes (adapted from [4]). The upper white vertical arrows represent the product derivation process of selecting and customizing reusable assets during application engineering.

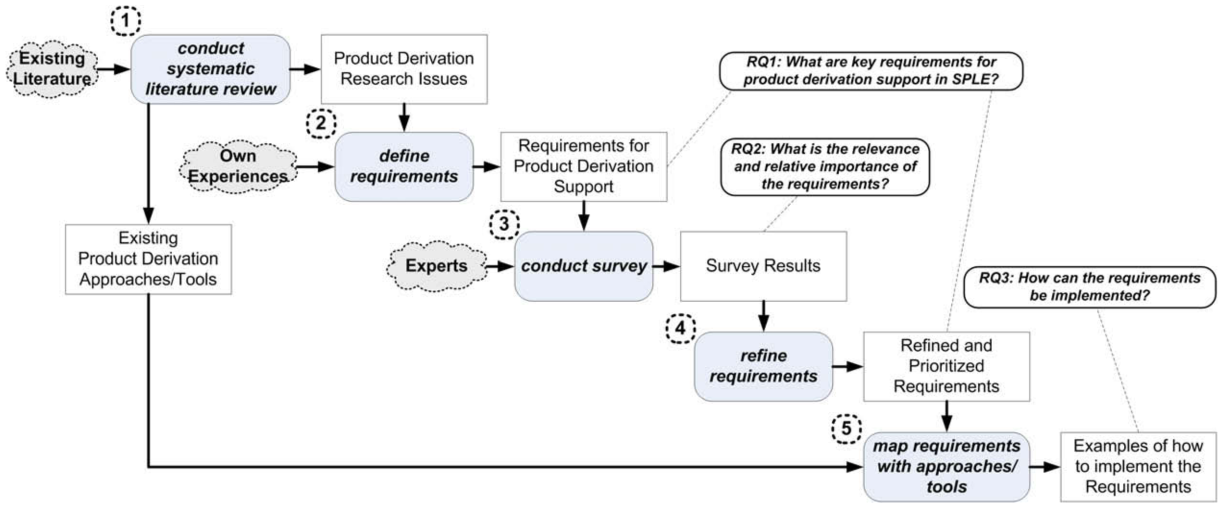
1
Existing Literature
conduct systematic literature review
Product Derivation Research Issues
2
Own Experiences
define requirements
Requirements for Product Derivation Support
RQ1: What are key requirements for product derivation support in SPLE?
RQ2: What is the relevance and relative importance of the requirements?
Existing Product Derivation Approaches/Tools
3
Experts
conduct survey
Survey Results
RQ3: How can the requirements be implemented?
4
refine requirements
Refined and Prioritized Requirements
5
map requirements with approaches/ tools
Examples of how to implement the Requirements

Fig. 2. Research method and research questions.

SPLE research community. Reviewing existing research in a fully objective way is not possible. A systematic review however reduces researchers' bias through pre-defined data forms and criteria that limit the room for interpretation.

Systematic literature reviews – often also termed systematic reviews – have been defined as "*a means of identifying, evaluating and interpreting all available research relevant to a particular research question, or topic area, or phenomenon of interest*" [14]. Such reviews have started to make their way into software engineering research [12,15] but are still not widely used. Performing a systematic review can be grouped into the phases of planning, conducting, and reporting. A key element in systematic reviews is the explicit definition of a review protocol in the planning phase that guides its execution. It aims to reduce researchers' bias and helps in structuring the retrieved results. The protocol defines the research questions for the SLR (focus); the search strategy (sources and timeframe for searching, rationale for choosing particular sources); selection criteria (terms used for searching, general restrictions); quality assessment criteria (inclusion and exclusion criteria for selecting a relevant subset of the found publications from a quality perspective); and the data extraction process (storage procedures for retrieved files, data extraction forms). The review protocol has to be validated, typically by consulting experienced researchers. In our case, the review protocol was developed by one of the authors as part of a PhD thesis [16] and was validated by two senior researchers.

We conducted the SLR and identified the relevant research by following a search strategy and by applying selection criteria. We assessed the quality of the selected publications by applying predefined quality criteria and collected data about the chosen publications in prepared extraction forms. Finally, we integrated and consolidated the collected data for reporting.

3.2. Define requirements (cf. Section 5)

We developed initial requirements for product derivation support based on the research issues identified in the SLR.

3.3. Conduct survey (cf. Section 5)

We used a survey [17] among experts to assess the relevancy, completeness, and relative importance of our initial requirements. Surveys are the preferred research technique for developing generalized suggestions based on collecting information from a certain population [18]. The most common means for data collection are questionnaires and interviews. We created questionnaires based on our initial definition of requirements for product derivation support. To ensure data quality we did not simply send out the questionnaires; instead, we personally handed out the questionnaire to particular participants and answered clarifying questions during the completion of the forms. We involved participants of the 12th International Software Product Line Conference (SPLC 2008, http://www.splc.net) and the 2nd International Workshop on Variability Modelling of Software-Intensive Systems 2008 (VaMoS 2008, http://www.vamos-workshop.net).

3.4. Refine requirements (cf. Section 6)

We prioritized and refined the requirements based on experts' opinion to investigate research questions 1 and 2. We merged some requirements based on the experts' suggestions and rephrased some requirements to address misunderstandings and ambiguities found during the survey. Furthermore, when refining the requirements we also took into account additional requirements suggested by the experts.

3.5. Map requirements with approaches and tools (cf. Section 6)

Finally, to address research question 3, we analyzed how existing approaches and tools (identified in the SLR before) implement the identified requirements and collected examples of the requirements' realization.

4. Product derivation research: a systematic literature review

The review includes publications reporting on existing approaches and tools as well as publications discussing research issues for product derivation. We conducted the SLR in 24 relevant sources and retrieved a total of 275 publications using well-defined search criteria. From these 275 publications, we chose 118 for further analyses based on our set of selection and quality criteria. The complete list of all 275 retrieved publications and details about the conducted searches can be found in Appendix.

4.1. Conducting the systematic review

The first step in conducting the SLR was the development of a review protocol containing the following elements:

The *rationale* for conducting the systematic review was to get an extensive overview of existing product derivation approaches and tools and to understand key issues for product derivation. Our systematic review was guided by the following *research questions*:

Table 1
SLR sources used in the search.

Source(s)	Digital libraries/resources	Publisher(s)
Journals/magazines		
IEEE Software	IEEE DL (www.computer.org)	IEEE
IEEE Transactions on Software Engineering (TSE)	IEEE DL	IEEE
Journal of Systems and Software (JSS)	Sciencedirect (sciencedirect.com)	Elsevier
Informatik Forschung und Entwicklung	Springerlink (http://www.springerlink.com)	Springer
Communications of the ACM	ACM DL (portal.acm.org/dl.cfm)	ACM
IEEE Intelligent Systems	IEEExplore (ieeexplore.ieee.org)	IEEE
IEEE Transactions on Knowledge and Data Engineering (TKDE)	IEEExplore	IEEE
Journal on Empirical Software Engineering	Springerlink	Springer
Journal of Automated Software Engineering (JASE)	Springerlink	Springer
ACM Transactions on Software Engineering and Methodology (TOSEM)	ACM DL	ACM
Conferences		
(International) Software Product Line Conferences (SPLC) and Workshops on Product Family Engineering (PFE)	IEEExplore (SPLC 2007), IEEE DL (SPLC 2006), Springerlink (PFE 98-03, SPLC 00-05), sei.cmu.edu	IEEE, Springer, CMU SEI
International Conferences on Software Reuse (ICSR)	Springerlink (2000–2006), IEEE DL (1996, 1998)	IEEE and Springer
International Conferences on Software Engineering (ICSE)	IEEExplore	IEEE
International Conferences on Automated Software Engineering (ASE)	ACM DL	ACM and IEEE
International Conferences on Requirements Engineering (RE)	IEEExplore	IEEE
European Software Engineering Conferences/ACM SIGSOFT Symposia on the Foundations of Software Engineering (ESEC/FSE)	ACM DL	ACM
Workshops		
International Workshop on Visualisation in Software Product Line Engineering (ViSPLE)	Available from publisher	Kindai Kagaku Sha Co Ltd.
Workshops on Variability Modeling of Software-intensive Systems (VaMoS)	vamos-workshop.net	Lero/Univ. of Duisburg-Essen
Workshop on Software Variability Management for Product Derivation (SVMPD)	Springerlink	Springer
Workshop on Requirements Engineering for Product Lines (REPL)	eprints.fraunhofer.de	Fraunhofer IESE
International Workshops on Product Line Engineering: The Early Steps (PLEES)	eprints.fraunhofer.de	Fraunhofer IESE
Others		
SEI Technical Reports	sei.cmu.edu	CMU SEI
Fraunhofer Technical Reports	eprints.fraunhofer.de	Fraunhofer IESE
Books on SPLE	Available from publishers	Diverse

- *(SLR-RQ1) Which approaches exist in SPLE that support product derivation*?
- *(SLR-RQ2) Which research issues are reported for product derivation in SPLE*?

We defined the following *search strategy*: The sources (presented in Table 1) were selected based on an analysis of product line literature already known by the authors and the reference lists of these publications. Regarding the Software Product Line Conference (SPLC) we searched several sources (cf. www.splc.net) to cover the five European workshops on product family engineering (1996, 1998, 2000, 2001, 2003) and the three US Software Product Line Conferences (2000, 2002, 2004). In 2005 both events were merged into the International Software Product Line Conference. Searching conferences includes searching workshops held jointly with these conferences if published in the same digital library (DL). Relevant sources were explored further. For all primary studies found in these sources we also followed their relevant cited references to find additional contributions outside the above-mentioned subset. All searches have been conducted between November 2007 and February 2008, except one search (in the ACM TOSEM journal) which has been conducted in April 2008 after the suggestion of a colleague. This search did however not lead to any additionally selected publications (cf. Section 4.2). The research dates back to the year 1996 thus covering over 12 years of SPLE research.

Depending on the source, different search terms were used. For the more general conferences (i.e., conferences except SPLC and ICSR) and for journals and magazines we used the search term "`product line`" OR "`product family`" OR "`product-line`" OR "`product-family`" appearing in the full-text of the publications (excluding references). In sources with many product line papers (e.g., SPLC and ICSR) this search term was too broad making it difficult to identify the relevant publications. In such cases we iteratively refined the search term, for example leading to the search term "`product derivation`" OR "`product configuration`" OR "`application engineering`". In some searches many papers were found that did not use the word product line in a SPLE context but rather to describe some "line of products". In these cases we used "*`software`* `product line`" OR "*`software`* `product family`" `or added` OR "`product line engineering`" OR "`product family engineering`".

We defined the following restrictions and quality *criteria for selection* of publications:

- (*Restriction R1*) The study only includes papers available in electronic form. Books were analyzed based on information available online and using the hard copy versions.
- (*Restriction R2*) Only publications written in English were included. For the journal "Informatik Forschung und Entwicklung", which includes German and English papers, only English articles were analyzed.
- (*Restriction R3*) Introductions to special issues, workshops, tutorials, conferences, or conference tracks were excluded.
- (*Quality criterion Q1*) For journal, conference, and workshop publications only peer-reviewed papers were taken into account. We did not exclude books and technical reports as in the SPLE community many books and technical reports of high quality are available, often even containing peer-reviewed papers.
- (*Quality criterion Q2*) Only publications that have been cited in at least five other peer-reviewed publications (evaluated using scholar.google.com and citeseer.ist.psu.edu) were selected. Exceptions from this rule are publications that are newer than January 2006.
- (*Quality criterion Q3*) Each publication was checked for completeness. Publications containing several unsupported claims or frequently referring to existing work without providing citations were excluded.

The retrieved publications were first analyzed regarding the restrictions R1–R3. The remaining publications were carefully assessed regarding quality criteria Q1–Q3. For each retrieved publication the following information was collected in a *data extraction* form:

- Date of search, source, and used search term.
- Authors, title, and publication year.
- Type of publication (conference, workshop, journal, report, or book).
- Classification of publication (research, experience, or position paper).
- Short summary (main claims, presented approach/tool).
- Restrictions R1, R2, R3 (yes or no)?
- Quality criterion Q1, Q2, Q3 fulfilled (yes or no)?
- Addressed SLR research question(s).
- Selected (yes or no)? . . .based on restrictions and quality criteria.
- Comments/rationale regarding selection.

For each selected publication the following additional information was captured in a second form to increase confidence regarding their relevancy for product derivation support:

- *Product derivation focus*. The main focus of the publication is on product derivation (yes) vs. product derivation is only addressed as part of presenting another approach (no)?
- *Need for product derivation tools*. The publication stresses the need for product derivation tool support (yes or no)?
- *Specific product derivation tool features*. The publication describes tool support for product derivation (yes) vs. the publication only mentions tool support or points out that tool support would be required as a proof of concept (no)?
- *Product derivation research issues*. What are the key issues for product derivation (if any) the publication raises (list)?

As our goal was to get an overview of existing product derivation approaches and tools and research issues in product derivation, we did not to perform a formal meta-analysis defined in a *data synthesis strategy*. Our data synthesis is descriptive in nature. We used the results of our SLR as an input for defining the requirements for product derivation support. This also explains lack of publications newer than February 2008: our further research steps were based on the SLR results available at that time.

4.2. Results overview

Thirty-seven searches in 24 sources were carried out using the search terms described above. In total 275 publications were retrieved (cf. Appendix), out of which 12 were not directly found in the 24 sources but by following relevant cited references. Fig. 3 shows the continuous growth of research results on product derivation between 1996 and 2007.

In total we chose 118 publications for further analyses based on the selection and quality criteria. Table 2 presents the number of retrieved and selected publications for each source, the SLR research questions addressed by the selected publications, and issues frequently discussed in the selected publications. We identified these issues by listing them in our data extraction forms. We finally were able to group them into eight key issues. Searching the software product line conferences led to most results (86), out of which 31 (36%) were selected. In total 143 conference papers were

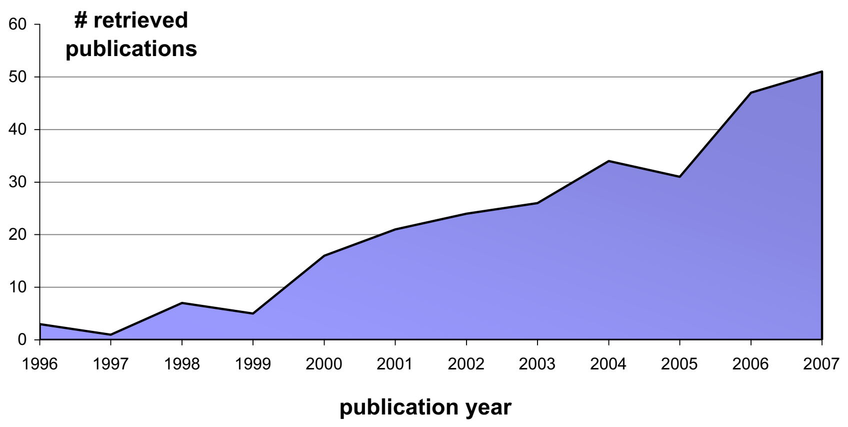

Fig. 3. Number of publications on product derivation (1996–2007).

found in the sources of which 54 (38%) were selected. A minimum of six papers was found in all searched conferences. Searching journals and magazines led to 57 results out of which we selected 27 (47%). Most results (30) were found in IEEE Software and Communications of the ACM. 39 results were retrieved from workshops, out of which we selected 18 (46%), most of them from the VaMoS series of workshops. Regarding other sources, 19 technical reports were retrieved out of which we selected 7. We further selected 12 of 17 identified books on SPLE. The overall selection rate for all sources is 43% (118 selected/275 retrieved publications).

Table 3 shows the focus of the selected publications based on data additionally captured in a second data extraction form (see Section 4.1). Only 22% of all selected publications deal primarily with product derivation. This indicates that despite increasing interest and agreed importance, product derivation is not yet the main focus of the research community. Few of the selected publications (14%) describe tool support for product derivation, whereas a much higher number of publications (47%) explicitly stress the need for product derivation tool support (however, mostly without defining what these tools should support). All selected publications either raise issues for product derivation or at least agree that product derivation support is important.

4.3. Results on SLR research question 1: existing product derivation approaches

We identified the following approaches in the selected publications. We briefly describe how they support product derivation and mention supporting tools (if available). If more than one selected publication describes the same approach and/or is by the same research group, we only reference one current publication (refer to the Appendix for a complete list).

3-Tiered methodology [19]. Krueger presents a pattern for software product line methodologies composed of three tiers, each describing roles, problems, solutions, benefits, and sources from a high-level point of view. The base tier is focused on variation management and automated production. The middle tier motivates core asset development. The top tier emphasizes feature-based portfolio evolution. The basic ideas of this methodology are realized in the tool GEARS [20] by BigLever Software Inc. (http://www.biglever.com). Product derivation is centered on automation by using feature profiles and (code) assets as input for generators.

AHEAD methodology [21]. Based on the well-known concept of step-wise refinement, this methodology defines products as “a sequence of refinements applied to the core artifacts”. Fine-grained pieces are assembled into components which are then assembled to products. The AHEAD tool suite (http://www.cs.utexas.edu/~schwartz/ATS.html) supports specifying the “product build-process” based on build files. The tool suite also provides feature selection and component composition capabilities. The AHEAD methodology and tool suite are mainly focused on code assets.

COVAMOF [9,22]. The *CO*nfiguration in Industrial Product Families (ConIPF) *VA*riability *MO*deling *F*ramework supports modeling variation points and dependencies at different levels, e.g., feature, architecture, and implementation. A derivation process is explicitly defined using the Software Process Engineering Meta-model (SPEM) notation by the Object Management Group. COVAMOF is supported by the COVAMOF-VS tool suite (http://www.covamof.com/) implemented as Microsoft Visual Studio add-ins. It provides variation point and dependency views on variability models and allows defining, configuring, and realizing products following the COVAMOF derivation process.

*D*ecision-Oriented *P*roduct *L*ine *E*ngineering for effective *R*euse (*DOPLER*^{UCon}) [23] supports decision-oriented variability modeling and product derivation (UCon: *U*ser-centered *Con*figuration). The DOPLER tool suite (http://.ase.jku.at/dopler/) supports preparing the product derivation process by defining tasks and roles based on variability models. A wizard then supports interactive product derivation and provides role-specific views on variability models.

KobrA [24,25]. This approach is centered on component-based product line development and integrates existing software engineering technologies like framework and process modeling. KobrA has been described as an “object-oriented customization of the PuLSE method” [26]. The application engineering process is explicitly defined and is based on decision models to present variability to the customer and to instantiate a concrete application based on “framework models”.

Kumbang/Koalish [27] is based on an extension to the architecture description language Koala [28] to enable variability modeling. It is centered on the idea of integrating feature and architecture models in a combined meta-model with formally defined semantics (the “Kumbang ontology”). Based on such models the Kumbang Configurator tool (http://www.soberit.hut.fi/KumbangTools/) supports the configuration of product models by using an AI inference engine called smodels.

Orthogonal variability modeling [4,13]. In this approach product line variability is documented in dedicated models independent of the variability realization in assets. Product derivation is addressed by providing extensions to standard notations like UML use cases in order to visualize the variation points. The application engineering process is organized in application requirements engineering, design, realization, and testing phases based on the

Table 2
Overview of the results of the SLR.

Source	Retrieved	Selected	ad. SLR research question 1	ad. SLR research question 2	Product derivation issues frequently discussed in the selected publications							
					Inadequate knowledge and variability management	Lack of support for domain experts	Weak process integration and tool interoperability	Missing support for project management	Low degree of automation of product derivation	Weak integration with application requirements management	Visualization inadequate for end-users	Missing consideration of product line evolution
Journals/magazines												
IEEE Software	15	10	7	4	1		1	2	2	1		1
IEEE TSE	1	0	0	0								
JSS	4	3	2	1	1	1				1		1
Informatik F&E	4	1	1	1		1				1	1	
Communications of the ACM	15	8	4	5		3			4			
IEEE Intelligent Systems	3	2	2	1					1			
IEEE TKDE	1	0	0	0								
Journal on Empirical Software Engineering	4	1	0	1			1		1			
JASE	2	2	2	2			2			1		
ACM TOSEM	8	0	0	0								
Total for journals/magazines	57	27	18	15	2	5	4	2	8	4	1	2
Conferences												
SPLC/PFE	86	31	25	19	11	4	6	7	12	4	5	8
ICSR	12	5	4	5					5			
ICSE	13	8	6	3	2	1	1				1	2
ASE	6	2	0	2	1				2	1		
RE	10	6	6	4	1	3		2	1	4		
ESEC/FSE	16	2	2	2	×				2			2
Total for conferences	143	54	43	35	15	8	7	9	22	9	6	12
Workshops												
ViSPLE	4	4	4	4	2	3			3		4	
VaMoS	18	10	8	8	6	3			3		2	
SVMPD	9	2	2	2	2		1		2			
REPL	4	2	1	1			1			1		
PLEES	4	0	0	0								
Total for workshops	39	18	15	15	10	6	2	0	8	1	6	0
Others												
SEI Reports	11	3	2	3			1	2	2			
Fraunhofer Reports	8	4	3	3	2		2	2				
Books	17	12	12	8	7	2	5	2	3	2	4	5
Total for others	36	19	17	14	9	2	8	6	5	2	4	5
Overall total	275	118	93	79	36	21	21	17	43	16	17	19

Table 3
Product derivation focus of selected publications.

Selected publications that	Total	%
Are mainly focused on product derivation	26/118	22
Stress the need for product derivation tool support	56/118	47
Describe specific product derivation tool features	17/118	14

domain artifacts. The VARMOD-Prime tool environment (http://www.sse.uni-due.de/wms/en/index.php?go=139) is currently under development. Tool support for product derivation is not yet available.

PLUS [6]. The *P*roduct *L*ine *U*ML-based *S*oftware engineering method provides techniques to extend single-system, UML-based design methods with product line concepts. Application engineering is carried out by tailoring the UML product line models (e.g., use case, application analysis, and architecture models) by selecting application features using a feature dependency model.

PRS [29]. *P*roduct-line *R*equirements *S*pecification is based on the idea of creating a requirements specification for product lines including a description of variability. Variations are described as decisions that allow deriving product requirements specifications based on text-replacement mechanisms.

PuLSE/PuLSE-I [26,30]. The *P*roduct-*L*ine *S*oftware *E*ngineering Method developed by the Fraunhofer IESE includes the application engineering process PuLSE-I (I for instantiation). Several process steps are defined based on other PuLSE artifacts, e.g., the reference architecture, the domain decision model, or the scope definition. PuLSE-I activities cover planning product derivation, instantiating a product architecture from the reference architecture using the decision model, and additional activities for design, implementation, and test. Delivery and maintenance processes are also addressed.

SEI Product Line Practice Initiative [3,31]. The SEI provides a high-level framework of practices for deciding when to automate product derivation, choosing the right technology, and planning and carrying out the derivation process. According to the framework, production plans have to be developed to prepare the derivation process. Such production plans are documents describing inputs, necessary activities, and desired outputs.

Staged configuration [32]. Based on several extensions to FODA feature models [33] Czarnecki et al. propose to perform derivation in stages, where some choices are eliminated in each stage. The output of each stage is a more specialized feature model. A configuration (where all choices have been eliminated) is derived from the most specialized feature model.

VISIT-FC [34]. Based on a self-defined feature meta-model, *Vi*sual and *I*nteractive *T*ool for *F*eature *C*onfiguration (VISIT-FC) represents large feature models for particular stakeholders using techniques known from the software information visualization community and allows selecting features.

4.4. Results on SLR research question 2: research issues in product derivation

Table 2 presented research issues in product derivation frequently discussed in the selected publications. Here, we briefly discuss these issues and point out references to publications discussing them in more detail:

Inadequate knowledge and variability management [8,35,36]. About one third of all selected publications describe this issue. Authors emphasize that product derivation heavily relies on expert involvement as the tacit knowledge regarding variability and reusable assets can hardly be captured completely in models. There is agreement that models can never replace the human expert; however, they should be more usable to support subsequent product derivation. Managing variability is challenged by the size of product lines. From the perspective of product derivation, variability models are difficult to use if not properly structured and modularized.

Lack of support for domain experts [13,22]. Product derivation effort is often centered too much on software engineers and architects who lack knowledge about customer requirements. This can easily lead to products that only inadequately fulfill customers' requirements. Communicating the variability of a product line to domain experts is thus a frequently discussed issue. For example, sales people need to interact with customers when tailoring a product to their very specific needs. Project managers need to monitor and track the status of derivation. The complexity of systems makes is difficult to communicate variability to these domain experts who typically have problems understanding variability at a technical level.

Weak process integration and tool interoperability [3,9]. Authors argue that product derivation is not or only weakly integrated with existing development processes and environments in organizations. SPLE and product derivation often fail because staff continues to follow single-system software engineering processes and practices they are used to. Also, product derivation tools are often not integrated with existing tools in organizations.

Missing support for project management [37,38]. Deriving a product from a product line is a project in itself, which has to be thoroughly planned and managed. This aspect is often not taken into account by existing product derivation approaches. For instance, the roles and responsibilities of the involved stakeholders are often not considered adequately. Also, project managers often lack an overview of the derivation process and therefore cannot decide whether higher-level business goals are fulfilled.

Low degree of automation in product derivation [32,39]. 36% of all selected papers stress the lack of automation and tools available for product derivation. Authors argue that the technical aspects of product derivation have to be automated as far as possible. Stakeholders need guidance for making decisions and resolving conflicts in case of open decisions or problems that cannot be automatically inferred.

Weak integration with application requirements management [40,41]. Integrating requirements engineering and product derivation is not easy or straightforward and needs explicit attention. Only in an unrealistic "blue-sky scenario" all customer requirements can be fulfilled by a product line's core assets. It is common that additional wishes and requirements arise during derivation. Capturing and managing these requirements and tracing them to the existing assets and their variability is seen as critical for product derivation.

Visualization inadequate for end-users [13,34]. Visualizing product line models during product derivation is not trivial. Stakeholders require different forms of visualization supporting them in understanding the relevant domain concepts and dependencies among them. The heterogeneity of the people involved in product derivation makes it very challenging to devise visualization features.

Missing consideration of product line evolution [42,43]. Product lines inevitably evolve due to changing customer requirements and technology. Product derivation results and experiences such as product-specific requirements, resolved conflicts, or the rationale of configuration decisions provide important input for product line evolution that needs to be properly managed.

4.5. Summary

The systematic literature review shows that the interest in product derivation increased considerably over the last decade. However, research is still fragmented and diverse. This strongly motivates our goal of defining and validating requirements for product derivation support (cf. research questions in Section 1).

The research issues addressed in the selected publications show that product derivation support needs to focus on the involved stakeholders to take into account their specific needs and to guide them through the product derivation process. Approaches and tools need to be adaptable and flexible to allow their integration with existing processes and environments. A particular challenge lies in mechanisms to facilitate managing and visualizing product line variability together with the knowledge required for product derivation. The relationship between product line evolution and product derivation needs special attention as continuous changes such as changing customer requirements or evolving technology have to be considered. In this context capabilities for application requirements management are important. Given all these issues it is not surprising that the existing approaches only support some of these requirements.

5. Requirements for product derivation support: an expert survey

Based on the issues for product derivation resulting from our SLR, we developed an initial definition of the requirements for product derivation support. We then performed a survey among experts at the VaMoS 2008 workshop and the SPLC 2008 conference to gain confidence about their relevancy, relative importance, and completeness. Please note that after VaMoS 2008 we extended the SLR by searching relevant papers published at this workshop (fulfilling the defined selection and quality criteria). Analyzing these additional publications confirmed the issues and requirements we identified and defined so far.

5.1. Designing the survey

The main goal of our survey was to validate the relevancy of the initial requirements and to understand their relative importance. Furthermore, we wanted to find additional requirements that might have been overlooked. Our questionnaire (cf. Appendix) contains three questions:

Question 1 (*How many years of experience do you have in the following fields? ...Domain Engineering, Application Engineering, Requirements Engineering, Project Management, Sales/Marketing, Product Management*) was intended to define a threshold for analyzing the respondents' experience in six relevant fields. Our assumption was that at least basic experience in domain engineering is required to understand product derivation requirements. Given the focus of the survey we expected respondents to be even more experienced in application engineering. Our threshold was as follows: Only questionnaires from respondents with at least three years experience in application engineering and at least one year experience in domain engineering were further considered. Information about respondents' experience in the four other fields was used to better understand their primary roles and interest (focus on business or technology).

Question 2 (*How important do you rate the following requirements for product derivation?... list of requirements:*) was intended to assess the initially defined requirements for product derivation support (see list below). To make the questionnaire easily understandable, we phrased some requirements as sentences and also provided examples where adequate (cf. Appendix). We chose to use a 4-point rating scale: −2 (totally irrelevant), −1 (unimportant), 1 (important), 2 (very important) to rate the initial requirements:

- *Support for resolving variability*. Given the complexity of variability models, tool support is needed to support stakeholders in selecting and customizing products by resolving variability. Tools must present variability in an understandable manner that abstracts from technical details and shows only the information necessary for variability resolution. Generally, tools should hide technical details like constraints or dependency conditions and support resolving variability just by enabling users making their choices. This requirement mainly addresses the research issues low degree of automation in product derivation and inadequate knowledge and variability management (cf. Section 4.4).
- *Support for application requirements management*. During product derivation customers usually express special requirements and wishes that are not fulfilled by the existing product line. A product derivation tool must take care of managing product-specific requirements. To understand the effects of such requirements and their potential impact on the evolution of the product line, product derivation tools must also provide support for tracing requirements to the existing variability. This requirement mainly addresses the research issues weak integration with application requirements management and missing consideration of product line evolution (cf. Section 4.4).
- *Guidance for decision-making*. Support for resolving variability may be insufficient when stakeholders have difficulties understanding the choices presented to them. Guidance is therefore needed in order to explain the *why and how* of making choices together with further background information and rationale on product derivation decisions. This requirement mainly addresses the research issue inadequate knowledge and variability management (cf. Section 4.4).
- *Support for domain experts*. A product derivation tool must take into account the specific needs of different domain experts. For example, technical details won't help sales people; they need to know the implications and effects of a particular choice from a higher-level point of view. Usability is particularly important in this context. This requirement mainly addresses the research issue lack of support for domain experts (cf. Section 4.4).
- *Flexibility and adaptability*. A product derivation tool must be adaptable to the specifics of different domains and/or organizations. For example, product derivation tools must be adaptable to domain changes (e.g., if new types of assets or dependencies are defined). It must also be possible to integrate the tools into existing environments and processes. The changing needs of users and the continuous evolution of the product line further motivate flexibility and adaptability of product derivation tools for addressing future needs. This requirement mainly addresses the research issues weak process integration and tool interoperability and missing consideration of product line evolution (cf. Section 4.4).
- *Interactivity and automation*. Variability should be presented to stakeholders in an interactive manner by providing instant feedback on users' choices. Such interactivity relies on automation such as reasoning techniques for evaluating dependencies and constraints on the fly. Users need support for "walking" through the decision-space step-by-step and want immediate feedback on the effects of their choices. Product derivation benefits from automating tasks to make variability resolution as convenient as possible. Evaluation and execution of conditions and constraints must work automatically in the background without requiring further input from the user. This requirement mainly addresses the research issue low degree of automation in product derivation (cf. Section 4.4).
- *Project management support*. Different people are responsible for different aspects of the available variability. Product derivation tools must thus manage users, their roles, and their responsibil-

ities with respect to resolving the available variability. Furthermore, it must be possible (e.g., for project managers) to track the progress of the product derivation process. This requirement addresses the research issue missing support for project management (cf. Section 4.4).

- *Flexible visualizations*. Different users benefit from different notations and visualizations. The kind of representation used should support the resolution of variability and different visualizations should be provided. Product derivation tools should allow integration of arbitrary additional visualizations to meet the specific needs of users in different domains. This requirement mainly addresses the research issue inadequate visualization for end-users (cf. Section 4.4).

Question 3 (*What other requirements do you regard as important?*) was intended to find additional requirements not yet mentioned as part of question 2.

5.2. Conducting the survey

We distributed the questionnaire to respondents in two phases:

5.2.1. Phase 1: VaMoS 2008

We first distributed the questionnaire at the 2nd International Workshop on Variability Modelling of Software-intensive Systems (VaMoS 2008) in Essen, Germany. Twenty-eight people from the SPLE research community participated in this workshop. We distributed 25 questionnaires. Eighteen (72%) were returned. Fourteen met our experience threshold.

5.2.2. Phase 2: SPLC 2008

In the second phase, we distributed the questionnaire at the 12th International Software Product Line Conference (SPLC 2008) in Limerick, Ireland. The main purpose of this second phase was to increase confidence on the results of the first phase. Over 200 people from research and practice participated at SPLC 2008, some had also attended VaMoS 2008. We only distributed questionnaires to selected participants who had not already participated in the survey at VaMoS 2008. We distributed 30 questionnaires. Twenty one (70%) were returned. Fifteen met our threshold requirements.

5.3. Results

We provide an overview covering the results of both phases. Details such as the individual ratings and statistical information are reported in Appendix.

5.3.1. Results for question 1

Most respondents meeting our criterion have significant experience in domain engineering (average experience (AvgExp) = 4.8 years) and application engineering (AvgExp = 9.2 years), many also in requirements engineering (AvgExp = 4.1 years) and project management (AvgExp = 4.6 years). On the other hand respondents' experience in sales/marketing (AvgExp = 0.6 years) and product management (AvgExp = 0.8 years) was rather low. The primary orientation of respondents was therefore clearly on technology rather than business. This is consistent with the purpose of our research and could be expected given the technical focus of the events.

5.3.2. Results for question 2

As shown in Fig. 4 in average all requirements have been rated as relevant and important (for detailed results refer to Table 6 in Appendix). The total average rating was 1.1. No requirement was rated as totally irrelevant (-2). We discuss the results in the order of the aggregated importance of the requirements:

Given the results of the systematic literature review (cf. Section 4) the rating for the first two requirements (*support for resolving variability*, *flexibility and adaptability*) is not surprising. *Support for application requirements management* was ranked third confirming the importance of support for managing product-specific requirements. The results for the requirement *flexible visualizations* – together with the second highest rating for general flexibility and adaptability – indicate that flexibility is essential for product derivation. Tools and visualizations must be adaptable to changing user needs and needs in different domains. Ten respondents out of 31 suggested to the authors verbally or by answering question 3 that *interactivity and automation* should be merged with the first requirement on support for resolving variability. In their opinion support should always be interactive and provide immediate feedback. The requirements *support for domain experts* and *guidance for decision-making* address the understandability of variability (models) and were rated comparably low. One possible reason is that most of the respondents were engineers (cf. the results for question 1) that might thus underestimate the need for guidance in product derivation. However, overall 19 people rated both requirements as important and 5 even as very important. Only 13 rated the item as unimportant. Some respondents commented that both requirements could be merged into a more general requirement "guidance for end-users". *Project management support* received a comparably low rating. However, still people rated the requirement as important and worth to be supported in product derivation. We believe that most of the participants associated this requirement with features of existing project management tools and did not feel the immediate need to integrate such features in product derivation tools.

5.3.3. Results for question 3

Thirty additional requirements were suggested by the respondents. We consolidated their comments and arrived at 11 suggestions we took into account when refining the requirements (the number of occurrences is shown in brackets):

Support for consistency checking across models (2). Adding, renaming, and deleting model elements can easily lead to inconsistencies. When following a model-driven approach, a product derivation tool must support consistency checking within and across models.

Integration with mainstream tools and development methods (5). Many existing product derivation tools cannot be integrated with mainstream software engineering tools (e.g., development environments like Visual Studio or Eclipse or requirements management tools like Doors or RequisitePro) or are only integrated in one specific environment but cannot be integrated in another. It is important to make product derivation tools more flexible and interoperable.

Support for traceability (5). Product derivation tools should support traceability from the variability description (decisions or features) to the realization of variability (assets) and to related requirements to support making choices.

Understandable constraint resolution/guidance in case of problems (3). Users of product derivation tools often have problems understanding the effects of the choices they make. Making constraint resolution understandable and providing guidance in case of problems is therefore seen as essential.

Rich graphic representation (1). Using rich graphics (e.g., 3D visualizations or semantically rich graphical notations) can help users in understanding the available choices.

Ability to add/edit meta-information attached to features/variability (2). Additional information added to decisions or features can help users understanding the why and how of making choices.

Collaborative and concurrent configuration support (1). Supporting the collaboration of the different people in product derivation

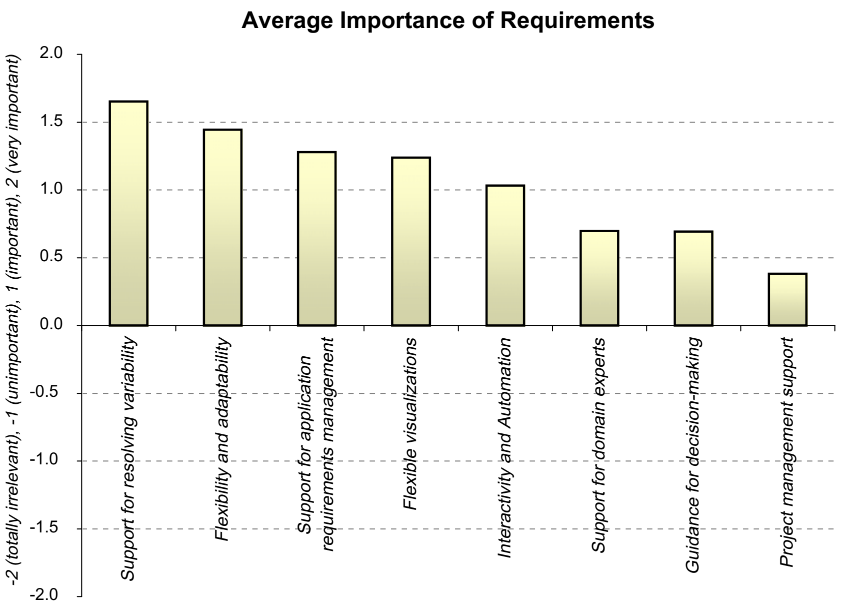

Fig. 4. Average importance of requirements (descending left-to-right).

is important as typically a single person cannot resolve the variability alone. Concurrent configuration would allow people making choices even if they are distributed.

Staged configuration support (3). Staged configuration means to support resolving variability step-by-step [32]. Variability resolved in past steps is removed (or marked as resolved and locked) for decision-makers in the current step.

Support for evolution (4). Product lines inevitably evolve, e.g., due to changes in technology and new customer requirements. Product derivation tools must reflect changes in the product line. Product derivation also drives evolution of the product line due to new customer requirements that emerged during derivation.

Generating documentation (2). The information captured in variability models can also be used to generate documentation such as requirements specifications and user documentation. Product derivation tools should provide such support and allow integrating custom generators.

Estimating the costs of product variant (2). Business-related information like costs can help decision-makers in their choices and also supports the negotiation of customer requirements during product derivation.

6. Requirements for product derivation support

The requirements for tool-supported product derivation are based on the results of both the survey among experts and the systematic literature review. We iteratively refined them based on the initial requirements presented in Section 5.1 resulting in a final set of six requirements. We have merged and rephrased some requirements based on the results of our survey. We were able to integrate all additionally suggested requirements in the six main requirements.

The requirements are intentionally defined at a high level and rather define areas of functionality. We provide implementation examples for the six requirements together with references to realizations in existing approaches and tools to further illustrate them. Most requirements are supported by existing tools at least to some extent; however, no tool supports all of them. We have thus divided the implementation examples in "basic support" and "advanced support". A feature is classified as basic support if it is common in available product derivation tools. It is classified as advanced support if available in few tools only or not supported by any tool. The optimal degree of support depends on the concrete domain or context.

Table 4 shows the six requirements and examples of how they can be realized. The order of the requirements indicates their relative importance given by the survey results (importance decreasing from top to bottom).

6.1. Automated and interactive variability resolution

The most important requirement for tool-supported product derivation is obviously to support resolving variability. Users need tools that present the available variability and let users decide about choices interactively. We have merged the two previously defined requirements *support for resolving variability* and *interactivity and automation* as suggested by experts participating in the survey. An additional requirement suggested by the experts was *support for traceability* as a prerequisite to provide better feedback on diverse choices. We consider the additionally suggested requirements *support for consistency checking across models*, *staged configuration support*, and *collaborative and concurrent configuration support* as part of automated and interactive variability resolution.

6.1.1. Basic support

- *Tool support for making decisions (e.g., selecting features).* Many SPLE tools for variability resolution are model-based, e.g., they visualize feature or decision models and allow users resolving variability by selecting features [27,32,34] or making decisions [23]. Feature selection is usually supported via a hierarchical representation of the feature models with check boxes, where users check the features required for the product to be derived by following the tree structure. Decision models are often presented to the user in tables (comparable to spreadsheets) or questionnaires, where users can select possible values (or answers) for decisions. Decisions/selected features determine the assets for the product to be derived.
- *Engines for resolving constraints and dependencies*. Users make their decisions or select their features and the tools are responsible for making sure that these decisions or features are not

Table 4
Requirements for product derivation with basic and advanced implementation examples.

Requirement	Implementation examples	
	Basic support	Advanced support
Automated and interactive variability resolution	• Tool support for making decisions (e.g., selecting features) • Engines for resolving constraints and dependencies	• Support for the automatic inclusion and customization of assets • Immediate visual feedback when making choices
Adaptability and extensibility	• Meta-modeling support • Extension points for the integration of domain-specific generators	• APIs for interacting with existing tools
Application requirements management support	• Requirements management capabilities in derivation	• Support for relating product-specific requirements with existing variability • Planning support for evolution
Flexible and user-specific visualizations of variability	• Support for filtering, sorting, searching the decision-space	• Role- and user-specific visualizations • Task-specific visualizations • Modifiable visualizations
End-user guidance	• Background information and rationale on choices	• Hints and recommendations • Interface to simulation support
Project management support	• Tracking project-related information	• User/role/task management support

conflicting. Inference engines [27], rule engines [23], or constraint solvers [32] allow resolving the dependencies between variability model elements and/or check (global) constraints during derivation. The effects of resolving dependencies and constraints should be made visible to the users without bothering them with technical details during variability resolution.

6.1.2. Advanced support

- *Support for the automatic inclusion and customization of assets*. Some tools, e.g., [19,22], support the automatic inclusion and customization of reusable assets based on selecting features or making decisions. This is an important feature for automated product derivation. Users' choices can be used not only for generating executable software packages or customizing code assets but also for generating end-user documentation like user manuals or requirements specifications.
- *Immediate visual feedback when making choices*. Interactions with the user during variability resolution benefit from immediate visual feedback. The implications of taking certain decisions should be communicated to the user. For example, when a feature selection leads to an inclusion of another feature, this should be properly visualized, e.g., by highlighting the newly included feature [32,34]. This can also mean to inform the user about the complexity, price, or development effort of current configurations.

6.2. Adaptability and extensibility

Product derivation tools must be adaptable and extensible to support different organizational and technological contexts. For instance, the type and granularity of product line assets varies in different domains and organizations and typically also changes over time. Automated deployment of a derived product also requires different technologies in different domains. While in one domain this could mean generating build files, generating code might be required in another domain. In another environment it might instead be necessary to set parameters, remove certain parts of code from existing source files, or to adapt documentation based on resolved variability. This is reflected by the requirement *generating documentation* suggested by participants of our survey. Furthermore, product derivation tools need to interact with tools already in use in different organizations (cf. the additionally suggested requirement *integration with mainstream tools and development methods*).

6.2.1. Basic support

- *Meta-modeling support*. Meta-modeling [6,27,34] is frequently adopted to support product derivation in different domains and organizations. Product derivation tools should adapt themselves based on the defined meta-model and provide support for reacting on meta-model changes during derivation, e.g., when adding new asset types to the product line from which a product is derived.
- *Integration of domain-specific generators*. Product derivation tools should provide extension points for easy integration of domain-specific generators [19,23]. Based on the results of product derivation, different outputs can be generated, e.g., build or make files, configuration files, requirements specifications, documentation, test cases, models, or even code.

6.2.2. Advanced support

- *APIs for interacting with existing tools*. Product derivation tools should also be able to interact with existing tools like off-the-shelf development environments, process and project management tools, etc. This is typically facilitated by providing a programming interface as, for example, provided by service-oriented architectures.

6.3. Application requirements management support

Product-specific requirements which cannot be fulfilled by the product line are likely to arise during product derivation. Users need to capture these additional requirements without being distracted from resolving variability. This is also important for managing a product line's evolution. We consider the additionally suggested requirement *support for evolution* part of application requirements management in our context.

6.3.1. Basic support

- *Requirements management capabilities in derivation*. Users need to capture product-specific requirements arising during derivation. Product derivation tools can either provide such capabilities or define interfaces to external requirements management tools. Support for text notes, sketches, audio recordings, and videos integrated in product derivation tools can be useful to further illustrate such requirements.

6.3.2. Advanced support

- *Support for relating product-specific requirements with existing variability*. Relationships between product-specific requirements and available product line features [23] are important to support product line evolution. Engineers should be able to analyze the new requirements to decide whether they should become a part of the product line in the future. For instance, product derivation tools can help engineers by providing associative search engines. Also, support for traceability is very important in this context.
- *Planning support for evolution*. The new requirements, customer comments, and the experiences of the users arising during product derivation need to be considered when maintaining and evolving the product line. All these inputs need to be managed and used when negotiating the scope of the evolving product line.

6.4. Flexible and user-specific visualizations of variability

Users find it difficult to understand variability models because of their sheer size: several thousand model elements with thousands of often complex dependencies can hardly be understood. Domain experts like sales and marketing staff need capabilities to communicate "high-level" variability to customers in order to elicit their requirements. Engineers like developers or testers need to resolve more technical variability based on these requirements. Product derivation tools should not be limited to one kind of representation and provide support for role- and user-specific visualizations. The additionally suggested requirement *rich graphic representation* is a part of requirement flexible and user-specific visualizations of variability.

6.4.1. Basic support

- *Support for filtering, sorting, searching the decision-space* is helpful to navigate in visualizations of large variability models in product derivation as for example provided by Sinnema et al. [22,23,34]. Such features currently are rather simple and are typically based on text without information regarding the semantics. This can however be improved by more advanced features based on ontologies, associative text search engines, or machine learning.

6.4.2. Advanced support

- *Role- and user-specific visualizations*. Diverse stakeholders are involved in product derivation and have to understand different aspects of variability. Depending on the current user and his role, different visualizations of variability in product derivation are necessary. While business people are more used to flat and tabular representations (e.g., spreadsheets), technical stakeholders also prefer hierarchical representations (e.g., the package structure of source code). Some approaches, e.g., [19,30], support a role-user concept, however, they do not (or not fully) use it for visualization purposes in product derivation tools.
- *Task-specific visualizations*. Depending on the task of the current user, e.g., negotiation with customers or technical configuration, different visualizations might be needed. For example, product derivation tools can present an interactive wizard to support negotiation or a simple settings dialog to support technical configuration.
- *Modifiable visualizations*. Visualization support must be flexible enough to allow adapting it to changing and future needs. It should be possible to integrate new types of visualizations. For example, interactive graphic elements improve user experience and make product derivation more attractive for users.

6.5. End-user guidance

The heterogeneous users performing product derivation not only need views on the available variability but also guidance when making decisions in product derivation. The requirements *understandable constraint resolution/guidance in case of problems*, *estimating the costs of product variant*, and *ability to add/edit meta-information attached to features/variability* have been additionally suggested in this context.

6.5.1. Basic support

- *Background information and rationale on choices*. Different users have different background knowledge about the product (line) which can make it hard to understand the variability in product derivation. Often, users can not make a choice due to a lack of information. For example, users would benefit from technical or economic details about a certain feature or information about the implications of deciding for a particular functionality. Furthermore, dependencies between choices must be explained properly [13,34].

6.5.2. Advanced support

- *Hints and recommendations*. Guidance can be useful to recommend particular features and also to make decision-makers aware of important dependencies. It is also useful to provide end-users with recommendations about which choice might be the best under certain conditions. This can also open new selling opportunities during product derivation [23]. Recommender systems can make suggestions to end-users based on the observed behavior of other users.
- *Interface to simulation support*. Understanding early in the process how the currently derived product will finally look like can make it easier to make choices. Simulating the end product might not be possible for all kinds of product lines. However, there are other ways letting users "preview" the product before it is completely derived and configure.

6.6. Project management support

Product derivation is typically conducted as a project. Over a possibly long period of time, several people with different responsibilities and interests resolve variability with the common goal to derive a product to fulfill customers' requirements. Supporting project management in product derivation means to manage the involved people, their rights and responsibilities throughout all product derivation activities.

6.6.1. Basic support

- *Tracking project-related information.* When product derivation is performed over a long period of time, project-related information must be available to measure the project's progress, for example, information about when and by whom decisions have been made for certain parts of the system. Such information can be a basis for creating statistics and optimizing the product derivation projects in the future.

6.6.2. Advanced support

- *User/role/task management support*. It must be clearly defined who is allowed to decide on which variation points. Managers need a derivation history with detailed information about who decided what and when. Existing product line approaches like for example the PuLSE method [30] describe important roles and tasks in product line engineering in general. However, existing approaches have not (or not yet fully) implemented such a concept in product derivation tools.

7. Threats to validity

As with any empirical research, there are threats to the validity of our results:

Choice of quality/selection criteria of the SLR, Defining the quality criteria for a SLR is not an easy and straightforward task. For example, our criterion Q2 (only taking into account publications cited in at least five other peer-reviewed publications if older than 2006) may lead to controversial opinions. In our opinion, the number a paper has been cited by others does not have to be a statement for its quality but is an indicator for its acceptance and visibility in the research community (for instance, a similar procedure has been used at ICSE, the International Conference on Software Engineering (http://www.icse-conferences.org), to define the "most influential paper"). Such acceptance and visibility in turn is a reason for selecting the paper in a systematic review. Quality criterion Q3 (publications containing several unsupported claims or frequently referring to existing work without providing a citation were excluded) may also lead to controversial opinions. It depends on subjective judgments by the reviewer which can only be reduced through feedback from peers.

Search engines used in the SLR. All retrieved results rely on the functionality and accuracy of the search engines of the used digital libraries. Unfortunately, many search engines of computer science digital libraries turned out to be unreliable. For example, in rare cases two consecutive searches with the same search term led to different results.

Abstracts of primary studies retrieved in the SLR. It is common in systematic literature reviews to base the selection of primary studies on reading the abstracts only. This is often not possible in the software product line domain because of the differing quality of the abstracts. For a more informed decision, we therefore frequently also read introduction, conclusion, and other parts of the publication. However, the decision to read or not read much more than the abstract (for the purpose of selecting a paper) strongly depends on the subjective feeling of the reviewer.

The definition of requirements was influenced by industrial and academic partners. There is a risk that we were biased by our own experience [16,23,44–46] as we collaborate with different industrial and academic partners. These experiences certainly influenced our views. However, by conducting a survey we tried to reduce this bias.

The orientation of respondents of our survey was mainly technology-oriented and not business-oriented. Only few respondents had experience in sales or marketing or product management. While this matches our research goal of defining requirements for product derivation tools it could mean that our results are biased towards the needs of technicians as opposed to the needs of more business-oriented people.

8. Conclusions

An increasing number of publications in the SPLE community show the growing interest in product derivation. The area of product derivation is however still rather immature. Existing work has not yet clearly defined the requirements for product derivation support. Rather, diverse publications outline different issues of and challenges for product derivation. Existing approaches and tools have been created independently, dealing with challenges in particular contexts or domains [11]. The aim of this paper was to create a systematic overview and to define and evaluate requirements for product derivation support in SPLE.

We presented the proceeding and results of a systematic literature review we conducted to get an extensive overview of existing research on product derivation. The review provides an overview of important existing approaches and tools and reveals the issues most frequently discussed in the identified publications. Based on these issues, we defined initial requirements for tool-supported product derivation. We presented the procedure and results of a survey we conducted among experts at two relevant events to gain confidence about the relevancy, completeness, and relative importance of these requirements.

Based on the results of the survey we iteratively refined the initial requirements. A challenge was to find a balance between too specific vs. too generic requirements. We have deliberately defined the requirements at a rather high-level. Researchers and practitioners can build on our results and refine them based on their own experiences or a concrete problem they are facing.

Our iterative refinements resulted in a final set of six requirements for product derivation support (ordered by their relative importance given through the results of the survey): (1) automated and interactive variability resolution; (2) adaptability and extensibility; (3) application requirements management support; (4) flexible and user-specific visualizations of variability; (5) end-user guidance; (6) project management support. For each requirement, we provided examples on how they can be realized and referenced existing approaches and tools identified in the systematic literature review where appropriate. We hope that our results are useful for researchers or practitioners when developing derivation support or when evaluating existing approaches.

Acknowledgements

This work has been supported by the Christian Doppler Forschungsgesellschaft, Austria and Siemens VAI Metals Technologies. We also want to thank Stefan Biffl for valuable feedback on earlier drafts of this paper.

Appendix A. Systematic literature review: conducted searches and retrieved publications

Table 5 presents the searches conducted and the list of publications retrieved in our systematic literature review. For brevity's sake, we only provide the following information: Resource, search terms, and for each retrieved paper: name of first author, title of

Table 5
Systematic literature review: conducted searches and retrieved publications.

Resource, search terms		
First author	Title	Pub. year
IEEE Software @ IEEE CS DL, "product line" or "product family" or "product-line" or "product-family"		
Clements	Project Management in a Software Product Line Organization	2005
Bockle	Calculating ROI for Software Product Lines	2004
Birk	Product Line Engineering: The State of the Practice	2003
Jaaksi	Developing Mobile Browsers in a Product Line	2002
Thiel	Modeling and Using Product Line Variability in Automotive Systems	2002
Kang	Feature-Oriented Product Line Engineering	2002
Schmid	The Economic Impact of Product Line Adoption and Evolution	2002
Northrop	SEI's Software Product Line Tenets	2002
McGregor	Guest Editors' Introduction: Initiating Software Product Lines	2002
van der Linden	Software Product Families in Europe: The Esaps and Café Projects	2002
Knauber	Applying Product Line Concepts in Small and Medium-Sized Companies	2000
Keepence	Using Patterns to Model Variability in Product Families	1999
Additionally found relevant references		
Clements	Point/Counterpoint	2002
Coplien	Commonality and Variability in Software Engineering	1998
Bruckhaus	The Impact of Tools on Software Productivity	1996
IEEE Transactions on Software Engineering @ IEEE CS DL, "product line" or "product family" or "product-line" or "product-family"		
Moon	An Approach to Developing Domain Requirements as a Core Asset Based on Commonality and Variability Analysis in a Product Line	2005
Journal of Systems and Software @ Sciencedirect, "product derivation" or "product configuration" or "application engineering"		
Sinnema	Industrial validation of COVAMOF	2007
Sardinha	A combined specification language and development framework for agent-based application engineering	2006
Deelstra	Product derivation in software product families: a case study	2005
Additionally found relevant references		
Bosch	Product instantiation in software product lines: a case study	2000
Informatik Forschung und Entwicklung @ springerlink.com, "product line" or "product family" or "product-line" or "product-family"		
Kiebusch	Prozess–Familien–Punkte	2006
Reuys	Szenario-basierter Systemtest von Software-Produktfamilien	2005
Härder	Editorial	2004
Halmans	Communicating the variability of a software-product family to customers	2004
Communications of the ACM @ ACM Digital Library, "software product line" or "software product family" or "product line engineering" or "product family engineering"		
Kishi	Formal verification and software product lines	2006
Czarnecki	Multi-level customization in application engineering	2006
Peña	Multi-agent system product lines: challenges and benefits	2006
Eriksson	Software product line modeling made practical	2006
Helferich	Product management for software product lines: an unsolved problem?	2006
Batory	Automated analysis of feature models: challenges ahead	2006
Bosch	The challenges of broadening the scope of software product families	2006
Krueger	New methods in software product line practice	2006
Mohan	Change management patterns in software product lines	2006
In	A quality-based cost estimation model for the product line life cycle	2006
Lee	Feature-oriented variability management in product line engineering	2006
Pohl	Software product line testing	2006
Clements	Getting there from here: a roadmap for software product line adoption	2006
Sugumaran	Introduction	2006
VaMoS'07 @ www.vamos-workshop.net, "product derivation" or "application engineering" or "product configuration"		
Liaskos	Exploring the Dimensions of Variability: a Requirements Engineering Perspective	2007
Brown	Requirements Modelling and Design Notations for Software Product Lines	2007
Myllärniemi	KumbangSec: An Approach for Modelling Functional and Security Variability	2007
Nestor	Visualisation of Variability in Software Product Line Engineering	2007
Lopez-Herrejon	Language and UML Support for Features: Two Research Challenges	2007
Classen	On the Structure of Problem Variability: From Feature Diagrams to Problem Frames	2007
Dhungana	DecisionKing: A Flexible and Extensible Tool for Integrated Variability Modeling	2007
Benavides	FAMA: Tooling a Framework for the Automated Analysis of Feature Models	2007
Krzanik	Specifying Variability of an Evolving Mobile System for Feasible Stakeholder Communication and Optimized Delivered Product Architecture	2007
IEEE/ACM International Conferences on Automated Software Engineering (ASE) @ ACM Digital Library, "software product line" or "software product family" or "product line engineering" or "product family engineering"		
Kavimandan	A parameterized model transformations approach for automating middleware QoS configurations in distributed real-time and embedded systems	2007

(*continued on next page*)

Table 5 (*continued*)

Resource, search terms		
First author	Title	Pub. year
Dhungana	Integrated Variability Modeling of Features and Architecture in Software Product Line Engineering	2006
Eriksson	The PLUSS toolkit: extending telelogic DOORS and IBM-rational rose to support product line use case modeling	2005
Zisman	Workshops: 2nd Workshop on the state of the art in automated software engineering	2005
Blundell	Parameterized Interfaces for Open System Verification of Product Lines	2004
Morisio	Extending UML to Support Domain Analysis	2000
European Software Engineering Conference/ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE) @ ACM Digital Library, "software product line" or "software product family" or "software product-line" or "software product-family"		
Baerisch	Model-driven test-case construction	2007
Fantechi	A behavioural model for product families	2007
Uzuncaova	A specification-based approach to testing software product lines	2007
Grammel	BugzillaMetrics: an adaptable tool for evaluating metric specifications on change requests	2007
Estublier	Reuse and variability in large software applications	2005
Pettersson	Industrial experience with building a web portal product line using a lightweight, reactive approach	2005
Browne	Verified systems by composition from verified components	2003
Batory	Refinements and multi-dimensional separation of concerns	2003
Bertolino	Use case-based testing of product lines	2003
Jahnke	Engineering component-based net-centric systems for embedded applications	2001
Morisawa	An architectural style of product lines for distributed processing systems, and practical selection method	2001
Coady	Using aspectC to improve the modularity of path-specific customization in operating system code	2001
Sullivan	The structure and value of modularity in software design	2001
Fisler	Modular verification of collaboration-based software designs	2001
Bayer	Transitioning legacy assets to a product line architecture	1999
Jarzabek	Synergy between component-based and generative approaches	1999
IEEE International Conferences on Software Engineering (ICSE) @ IEEEXplore, "product line engineering" or "product family engineering" or "software product line" or "software product family"		
Liu	Handling Safety-Related Feature Interaction in Safety–Critical Product Lines	2007
Dehlinger	DECIMAL and PLFaultCAT: From Product-Line Requirements to Product-Line Member Software Fault Trees	2007
Gomaa	Model-Based Software Design and Adaptation	2007
Ubayashi	A Reflective Aspect-Oriented Model Editor Based on Meta-model Extension	2007
Metzger	Variability Management in Software Product Line Engineering	2007
Verlage	Five years of product line engineering in a small company	2005
Schmid	Introducing the PuLSE approach to an embedded system population at Testo AG	2005
Salzmann	ICSE workshop: software engineering for automotive systems	2004
Matinlassi	Comparison of software product line architecture design methods: COPA, FAST, FORM, KobrA and QADA	2004
Schmid	A comprehensive product line scoping approach and its validation	2002
Bratthall	Integrating hundred's of products through one architecture – the industrial IT architecture	2002
Additionally found relevant references (by following references in the above-mentioned publications)		
Bayer	PuLSE-I: Deriving Instances from a Product Line Infrastructure	2000
Bayer	PULSE: A Methodology to Develop Software Product Lines	1999
Journal on Automated Software Engineering (ASE) @ Springerlink, "software product line" or "software product family" or "product line engineering" or "product family engineering"		
Dehlinger	PLFaultCAT: A Product-Line Software Fault Tree Analysis Tool	2006
Padmanabhan	Tool-Supported Verification of Product-Line Requirements	2005
IEEE International Conference on Requirements Engineering (RE) @ IEEEXplore, "software product line" or "software product family" or "product line engineering" or "product family engineering"		
Djebbi	Industry Survey of Product Lines Management Tools: Requirements, Qualities and Open Issues	2007
Metzger	Disambiguating the Documentation of Variability in Software Product Lines: A Separation of Concerns, Formalization and Automated Analysis	2007
Rabiser	Involving Non-Technicians in Product Derivation and Requirements Engineering: A Tool Suite for Product Line Engineering	2007
Salifu	Specifying Monitoring and Switching Problems in Context	2007
Reiser	Managing Highly Complex Product Families with Multi-Level Feature Trees	2006
Schobbens	Feature Diagrams: A Survey and a Formal Semantics	2006
van de Weerd	Towards a Reference Framework for Software Product Management	2006
Smith	Systematic Development of Requirements Documentation for General Purpose Scientific Computing Software	2006
Liaskos	On Goal-based Variability Acquisition and Analysis	2006

Table 5 (*continued*)

Resource, search terms First author	Title	Pub. year
Additionally found relevant references		
Faulk	Product-Line Requirements Specification (PRS): an Approach and Case Study	2001
Workshop on Requirements Engineering for PLs (REPL) @ http://publica.fhg.de/documents/N-14658.html, "product derivation" or "application engineering" or "product configuration"		
MacGregor	Requirements Engineering in Industrial Product Lines	2002
Bertolino	Use Case Description of Requirements for Product Lines	2002
John	Tailoring Use Cases for Product Line Modeling	2002
Padmanabhan	DECIMAL: A Requirements Engineering Tool for Product Families	2002
International Workshops on PLE: The Early Steps (PLEES) @ http://www.plees.info/, "product derivation" or "product configuration" or "application engineering"		
van Zyl	An Approach to Assemble Software Products using a Product Line Approach	2003
Pohjonen	Automated Production of Family Members: Lessons Learned	2002
Halmans	Considering Product Family Assets when Defining Customer Requirements	2001
Maccari	Industrial Keynote – Feature Modeling in an Industrial Context	2001
Books on Product Line Engineering @ Amazon.de "product line engineering" or "product family engineering"		
Olumofin	A Holistic Method for Assessing Software Product Line Architectures. Rationale, Concepts, Stages, and Case Studies	2007
Pankratius	Product lines for digital information products	2007
van der Linden	Software Product Lines in Action: The Best Industrial Practice in Product Line Engineering	2007
Käkölä	Software Product Lines. Research Issues in Engineering and Management	2006
Gorton	Essential Software Architecture	2006
Pohl	Software Product Line Engineering. Foundations, Principles, and Techniques	2005
Schmid	Planning Software Reuse – A Disciplined Scoping Approach for Software Product Lines	2003
Muthig	A Light-weight Approach Facilitating an Evolutionary Transition Towards Software Product Lines	2002
Books on Product Line Engineering @ http://www.softwareproductlines.com/resources/books.html "product line engineering" or "product family engineering"		
Gomaa	Designing Software Product Lines with UML	2005
Greenfield	Software Factories	2004
Böckle	Software-Produktlinien: Methoden, Einführung und Praxis	2004
Atkinson	Component-Based Product Line Engineering with UML	2002
Clements	Software Product Lines: Practices and Patterns	2001
Bosch	Design and Use of Software Architectures: Adopting and evolving a product-line approach	2000
Czarnecki	Generative Programming: Methods, Tools, and Applications	2000
Weiss	Software Product-Line Engineering: A Family-Based Software Development Process	1999
Additionally found relevant references		
Hotz	Configuration in Industrial Product Families – The ConIPF Methodology	2006
IEEE International Software Product Line Conference 2007 @ IEEEXplore, "product derivation" or "application engineering" or "product configuration"		
Gruler	Development and Configuration of Service-based Product Lines	2007
Janota	Reasoning about Feature Models in Higher-Order Logic	2007
Sellier	Introducing Software Product Line Engineering for Metal Processing Lines in a Small to Medium Enterprise	2007
Krueger	The 3-Tiered Methodology: Pragmatic Insights from New Generation Software Product Lines	2007
Ganesan	Comparing Costs and Benefits of Different Test Strategies for a Software Product Line: A Study from Testo AG	2007
Ishida	Software Product Lines Approach in Enterprise System Development	2007
Chang	A Variability Modeling Method for Adaptable Services in Service-Oriented Computing	2007
Rabiser	Supporting Product Derivation by Adapting and Augmenting Variability Models	2007
Park	A Component Model supporting Decomposition and Composition of Consumer Electronics Software Product Lines	2007
Voelter	Product Line Implementation using Aspect-Oriented and Model-Driven Software Development	2007
Schirmeier	Tailoring Infrastructure Software Product Lines by Static Application Analysis	2007
Habli	Challenges of Establishing a Software Product Line for an Aerospace Engine Monitoring System	2007
Loesch	Optimization of Variability in Software Product Lines	2007
Journal on Empirical Software Engineering @ springerlink, "software product line" or "software product family" or "product line engineering" or "product family engineering"		
Kitchenham	Evaluating guidelines for reporting empirical software engineering studies	2007
Mohagheghi	Quality, productivity and economic benefits of software reuse: a review of industrial studies	2007
Natt och Dag	An experiment on linguistic tool support for consolidation of requirements from multiple sources in market-driven product development	2006

(*continued on next page*)

Table 5 (*continued*)

First author	Title	Pub. year
Svahnberg	An Investigation of a Method for Identifying a Software Architecture Candidate with Respect to Quality Attributes	2005
1st Workshop on Product-Family Engineering ("Workshop on the Design and Evolution of Software Architecture of Product Families") @ http://www.sei.cmu.edu/productlines/splc/las_navas/Proceedings.htm, "product derivation" or "product configuration" or "application engineering"		
Maymir-Ducharme	A Product Line Business Model	1996
Karhinen	Configurable Designs	1996
1st Software Product Line Conference @ citeseer (only some papers available (9 of 27), "product derivation" or "product configuration" or "application engineering"		
Ardis	Domain Engineered Configuration Control	2000
Atkinson	"Component-Based Product Line Development: The KobrA Approach"	2000
Faulk	"Value-Based Software Engineering (VBSE): A Value-Driven Approach to Product-Line Engineering	2000
10th International Software Product Line Conference @ IEEE DL, "product derivation" or "product configuration" or "application engineering"		
Asikainen	A Unified Conceptual Foundation for Feature Modelling	2006
Ganesan	Predicting Return-on-Investment for Product Line Generations	2006
Helferich	Reconciling Marketed and Engineered Software Product Lines	2006
Hunt	Organizing the Asset Base for Product Derivation	2006
John	A Practical Guide to Product Line Scoping	2006
Kolb	Experiences with Product Line Development of Embedded Systems at Testo AG	2006
Lee	A Feature-Oriented Approach to Developing Dynamically Reconfigurable Products in Product Line Engineering	2006
Scheidemann	Optimizing the Selection of Representative Configurations in Verification of Evolving Product Lines of Distributed Embedded Systems	2006
Trask	Using Model-Driven Engineering to Complement Software Product Line Engineering in Developing Software Defined Radio Components and Applications	2006
Workshop on Software Variability Management for Product Derivation: Towards Tool Support @ http://www.soberit.hut.fi/SPLC-WS/, "product derivation" or "product configuration" or "application engineering"		
Asikainen	Using a Configurator for Modelling and Configuring Software Product Lines based on Feature Models	2004
Beuche	Demonstration: Variant and Variability Management with pure::variants	2004
Gomaa	Tool Support for Software Variability Management and Product Derivation in Software Product Lines	2004
Hotz	Combining Software Product Lines and Structure-based Configuration – Methods and Experiences	2004
Myllärniemi	Tool for Configuring Product Individuals from Configurable Software Product Families	2004
Pohjonen	Product Derivation through Domain-Specific Modeling: Collected Experiences	2004
Sinnema	Tool Support for COVAMOF	2004
Stephenson	Automated Component Configuration in Safety–Critical Domains	2004
von der Maßen	Deficiencies in Feature Models	2004
IEEE Transactions on Knowledge and Data Engineering @ IEEEXplore, "product line" or "product family" or "product-line" or "product-family"		
Zhou	On the Customization of Components: A Rule-Based Approach	2007
2nd Software Product Line Conference 2002 @ springerlink, "product derivation" or "product configuration" or "application engineering"		
Ferber	Feature Interaction and Dependencies: Modeling Features for Reengineering a Legacy Product Line	2002
Geyer	On the Influence of Variabilities on the Application-Engineering Process of a Product Family	2002
Krueger	Variation Management for Software Production Lines	2002
Additionally found relevant references		
van Deursen	Feature-Based Product Line Instantiation Using Source-Level Packages	2002
3rd Software Product Line Conference 2004 @ springerlink, "product derivation" or "product configuration" or "application engineering"		
Bosch	On the Development of Software Product-Family Components	2004
Czarnecki	Staged Configuration Using Feature Models	2004
Deelstra	Experiences in Software Product Families: Problems and Issues During Product Derivation	2004
Lago	Observations from the Recovery of a Software Product Family	2004
Männistö	Workshop on Software Variability Management for Product Derivation – Towards Tool Support	2004
Niemelä	Practical Evaluation of Software Product Family Architectures	2004
Pavel	Dynamic Configuration of Software Product Lines in ArchJava	2004
Sinnema	COVAMOF: A Framework for Modeling Variability in Software Product Families	2004
Steger	Introducing PLA at Bosch Gasoline Systems: Experiences and Practices	2004
Bosch	Software Product Family Evaluation	2004
9th International Software Product Line Conference 2005 @ springerlink, "product derivation" or "product configuration" or "application engineering"		
Batory	Feature Models, Grammars, and Propositional Formulas	2005
Ben Lamine	Cost Estimation for Product Line Engineering Using COTS Components	2005
Böckle	Innovation Management for Product Line Engineering Organizations	2005

Table 5 (*continued*)

Resource, search terms		
First author	Title	Pub. year
Etxeberria	Product-Line Architecture: New Issues for Evaluation	2005
Kishi	Design Verification for Product Line Development	2005
Niemelä	Strategies of Product Family Architecture Development	2005
Tessier	Using Variation Propagation for Model-Driven Management of a System Family	2005
Tolvanen	Defining Domain-Specific Modeling Languages to Automate Product Derivation: Collected Experiences	2005
von der Maßen	Determining the Variation Degree of Feature Models	2005
Zhang	Reuse without Compromising Performance: Industrial Experience from RPG Software Product Line for Mobile Devices	2005
Additionally found relevant references		
Díaz	"Supporting Production Strategies as Refinements of the Production Process"	2005
2nd Workshop on Product-Family Engineering ("ESPRIT ARES Workshop: Development and Evolution of Software Architectures for Product Families") 1998 @ springerlink, "product derivation" or "product configuration" or "application engineering"		
DeBaud	The Relation between the Product Line Development Entry Points and Reengineering	1998
3rd Workshop on Product-Family Engineering ("International Workshop IW-SAPF-3: Software Architectures for Product Families") 2000 @ springerlink, "product derivation" or "product configuration" or "application engineering"		
Stuart	Dependency Navigation in Product Lines Using XML	2000
Bosch	Organizing for Software Product Lines	2000
Vehkomäki	A Comparison of Software Product Family Process Frameworks	2000
Cerón	A First Assessment of Development Processes with Respect to Product Lines and Component Based Development	2000
van der Linden	ESAPS – Engineering Software Architectures, Processes and Platforms for System Families	2000
4th Workshop on Product-Family Engineering (PFE) 2001 @ springerlink, "product derivation" or "product configuration" or "application engineering"		
Bachmann	Report on Discussion Sessions "Diversity Solutions" and "Light-Weight Processes"	2001
Bayer	Introducing Traceability to Product Lines	2001
Becker	Comprehensive Variability Modelling to Facilitate Efficient Variability Treatment	2001
Bosch	Variability Issues in Software Product Lines	2001
John	Integrating Legacy Documentation Assets into a Product Line	2001
Krueger	Easing the Transition to Software Mass Customization	2001
Pohl	Considering Variabilities during Component Selection in Product Family Development	2001
Salicki	Expression and Usage of the Variability in the Software Product Lines	2001
5th Workshop on Product-Family Engineering (PFE) 2003 @ springerlink, "product derivation" or "product configuration" or "application engineering"		
Asikainen	A Koala-Based Approach for Modelling and Deploying Configurable Software Product Families	2003
Bachmann	A Meta-model for Representing Variability in Product Family Development	2003
Bayer	Design for Quality	2003
Bertolino	PLUTO: A Test Methodology for Product Families	2003
Bosch	Research Topics and Future Trends	2003
Brown	A Relational Architecture Description Language for Software Families	2003
Buhrdorf	Salion's Experience with a Reactive Software Product Line Approach	2003
Deelstra	A Product Derivation Framework for Software Product Families	2003
Fantechi	Elicitation of Use Cases for Product Lines	2003
Gomaa	Dynamic Software Reconfiguration in Software Product Families	2003
Hallsteinsen	Patterns in Product Family Architecture Design	2003
Jaring	Variability Dependencies in Product Family Engineering	2003
Kamsties	Testing Variabilities in Use Case Models	2003
Krueger	Towards a Taxonomy for Software Product Lines	2003
McRitchie	Managing Component Variability within Embedded Software Product Lines via Transformational Code Generation	2003
Mohagheghi	Different Aspects of Product Family Adoption	2003
van der Linden	Software Product Family Evaluation	2003
von der Maßen	RequiLine: A Requirements Engineering Tool for Software Product Lines	2003
Ziadi	Towards a UML Profile for Software Product Lines	2003
International Conferences on Software Reuse @ IEEE and springerlink, "product derivation" or "product configuration" or "application engineering"		
Blois	Variability Modeling in a Component-Based Domain Engineering Process	2006
Gomaa	Designing Software Product Lines with UML 2.0: From Use Cases to Pattern-Based Software Architectures	2006
Gomaa	Feature Driven Dynamic Customization of Software Product Lines	2006
Jansen	GENMADEM: A Methodology for Generative Multi-agent Domain Engineering	2006
Lee	An Approach to Managing Feature Dependencies for Product Releasing in Software Product Lines	2006
Peng	Ontology-Based Feature Modeling and Application-Oriented Tailoring	2006

(*continued on next page*)

Table 5 (*continued*)

Resource, search terms First author	Title	Pub. year
Sinnema	The COVAMOF Derivation Process	2006
Spagnoli	Adaptation and Composition Within Component Architecture Specification	2006
Anastasopoulos	An Evaluation of Aspect-Oriented Programming as a Product Line Implementation Technology	2004
van der Storm	Variability and Component Composition	2004
Forsell	Use and Identification of Components in Component-Based Software Development Methods	2000
Griss	Integrating Feature Modeling with the RSEB	1998
ViSPLE 2007 @ www.lero.ie/visple2007 *and in printed form, "product derivation" or "product configuration" or "application engineering"*		
Hashimoto	An EMF-based Product Derivation Tool for Large Product Lines	2007
Sellier	Visualizing Product Line Requirement Selection Decisions	2007
Rabiser	Tool Support for Product Derivation in Large-Scale Product Lines: A Wizard-based Approach	2007
Botterweck	Towards Supporting Feature Configuration by Interactive Visualisation	2007
11th International Software Product Line Conference – Tool Demonstrations @ printed proceedings, "product derivation" or "product configuration" or "application engineering"		
Myllärniemi	Kumbang Tools	2007
Dhungana	DOPLER An Adaptable Tool Suite for Product Line Engineering	2007
VaMoS 2008 @ www.vamos-workshop.net, *"product derivation" or "product configuration" or "application engineering"*		
Alférez	Tracing between Features and Use Cases: A Model-Driven Approach	2008
Bencomo	Reflective Component-based Technologies to Support Dynamic Variability	2008
Cawley	Interactive Visualisation to Support Product Configuration in Software Product Lines	2008
Forster	Understanding Decision Models – Visualization and Complexity reduction of Software Variability	2008
Grimm	Increasing the Reliability of Model-Driven Software Family Engineering and Product Configuration for Automotive Controller Software	2008
Raatikainen	Svamp – An Integrated Approach to Modeling Functional and Quality Variability	2008
Rabiser	Value-Based Elicitation of Product Line Variability: An Experience Report	2008
Schmid	Model-Based Implementation of Meta-Variability Constructs: A Case Study using Aspects	2008
Siegmund	Integrated Product Line Model for Semi-Automated Product Derivation Using Non-Functional Properties	2008
IEEE Intelligent Systems @ IEEEXplore, "product line" or "product family" or "product-line" or "product-family"		
Sinz	Configuration	2007
Sabin	Product Configuration Frameworks – A Survey	1998
McGuinness	An Industrial-Strength Description Logic-Based Configurator Platform	1998
fraunhofer technical reports @ publica.fraunhofer.de, "application engineering" or "product derivation" or "product configuration"		
John	Proceedings of the Software Product Lines Doctoral Symposium	2006
Lehner	ASG Application and Service Engineering Approach	2006
Lehner	ASG Development Processes – Application and Service Engineering	2006
Bayer	Product Line Engineering and Software Project Management	2004
Muthig	GoPhone – A Software Product Line in the Mobile Phone Domain	2004
Bunse	UML-based Development of Embedded Software Systems	2004
Knauber	Proceedings of Software Product Lines: Economics, Architectures, and Implications	2001
Additionally found relevant references		
Schmid	People Management in Institutionalizing Product Lines	2003
CMU SEI technical reports @ www.sei.cmu.edu, "product derivation" or "product configuration" or "application engineering"		
McGregor	Preparing for Automated Derivation of Products in a Software Product Line	2005
Clements	The U.S. Army's Common Avionics Architecture System (CAAS) Product Line: A Case Study	2005
Elm	Designing for Reuse of Configurable Logic	2005
Bachmann	Variability in Software Product Lines	2005
Bergey	Fourth DoD Product Line Practice Workshop Report	2001
Clements	Control Channel Toolkit: A Software Product Line Case Study	2001
Bass	Fourth Product Line Practice Workshop Report	2000
Bergey	DoD Product Line Practice Workshop Report	1998
Bass	Second Product Line Practice Workshop Report	1998
Bass	Product Line Practice Workshop Report	1997
Additionally found relevant references		
Chastek	A Study of Product Production in Software Product Lines	2004
ACM Transactions on Software Engineering and Methodology (TOSEM) @ ACM DL, "product line" or "product family" or "product-line" or "product-family"		
Krishnamurthi	Foundations of Incremental Aspect Model-Checking	2007
Estublier	Impact of Software Engineering Research on the Practice of Software Configuration Management	2005
Dashofy	A Comprehensive Approach for the Development of Modular Software Architecture Description Languages	2005

Table 5 (*continued*)

Resource, search terms		
First author	Title	Pub. year
Roshandel	Mae—A System Model and Environment for Managing Architectural Evolution	2004
Smaragdakis	Mixin Layers: An Object-Oriented Implementation Technique for Refinements and Collaboration-Based Designs	2002
Batory	Achieving Extensibility Through Product-Lines and Domain-Specific Languages: A Case Study	2002
Medvidovic	Modeling Software Architectures in the Unified Modeling Language	2002
Perry	Parallel Changes in Large-Scale Software Development: An Observational Case Study	2001

Table 6
Results of the requirements survey.

Q	Question 1 (respondents' experience)						Question 2 (respondents' rating for requirements)							
	DE	AE	RE	PM	S/M	PDM	TS	ARM	GDM	DE	FAT	IA	PMS	FV
VaMoS 2008														
1	6	6	0	0	0	0	2	2	2	1	2	2	2	1
2	6	15	14	7	0	0	2	1	1	1	1	2	2	2
3	0	0	0	6	0	3	2	1	1	1	2	2	1	2
4	1	1	3	1	1	1	2	2	1	1	2	1	1	1
5	5	5	0	3	0	0	1	2	1	2	1	2	1	2
6	3	0	0	0	0	0	2	1	2	1	2	2	–	2
7	10	10	15	8	8	8	1	1	−1	−1	1	2	1	−1
8	2	6	14	10	0	0	2	2	2	1	2	1	1	1
9	2	7	2	2	1	3	2	1	1	−1	2	−1	1	1
10	1	3	0	0	0	0	2	2	2	−1	1	1	1	2
11	5	25	0	15	0	0	1	1	−1	1	2	2	−1	1
12	7	13	4	2	0	0	2	−1	−1	1	2	1	−1	2
13	4	4	0	1	0	0	1	2	−1	−1	1	−1	−1	1
14	3	15	5	4	0	0	1	2	2	1	1	1	−1	1
15	10	10	0	10	0	0	2	1	1	1	2	1	1	1
16	0	0	0	0	0	0	1	1	−1	2	2	1	1	1
17	3	3	3	1	0	0	1	1	2	1	2	2	1	1
18	3	7	1	1	0	0	2	2	1	–	−1	−1	−1	1
x	4.8	9.2	4.1	4.6	0.6	0.8	1.6	1.4	0.8	0.5	1.4	1	0.4	1.1
V	–	–	–	–	–	–	0.3	0.7	1.6	1.1	0.7	1.4	1.3	0.6
σ	–	–	–	–	–	–	0.5	0.8	1.3	1.1	0.8	1.2	1.2	0.8
SPLC 2008														
19	2	2	5	5	0	0	2	−1	1	−1	2	2	−1	2
20	5	5	1	0	0	0	2	1	1	1	2	2	1	2
21	15	15	15	15	15	15	2	1	1	2	1	2	1	1
22	3	3	2	0	0	2	1	1	1	1	−1	1	−1	1
23	0	0	0	8	0	0	2	1	1	2	2	1	2	2
24	3	6	0	0	0	0	2	1	−1	1	2	1	1	1
25	2	8	2	0	0	0	2	−1	1	1	2	−1	2	1
26	5	10	2	1	0	1	2	1	1	1	2	−1	−1	2
27	6	6	6	0	0	0	2	2	1	1	2	1	1	2
28	4	6	3	3	0	0	2	1	1	2	2	1	1	2
29	4	12	12	6	0	0	2	2	−1	−1	2	1	−1	1
30	3	3	0	0	0	0	2	2	1	1	1	2	−1	1
31	5	3	5	5	1	0	1	2	1	2	2	2	2	1
32	1	1	0	0	0	0	2	2	−1	−1	1	2	2	1
33	4	4	4	4	0	0	1	2	1	1	1	1	1	1
34	0	3	0	0	0	1	2	2	2	−1	1	2	1	2
35	2	5	2	0	0	0	2	2	1	1	2	1	1	1
36	5	5	5	0	0	0	1	−1	−1	−1	1	1	−1	1
37	1	1	0	0	0	0	2	2	1	2	2	1	1	1
38	3	3	3	0	0	0	2	2	1	1	2	2	−1	2
39	2	2	0	2	0	0	2	1	1	1	2	2	1	2
x	4.6	6.3	4.1	2.3	1.1	1.2	1.7	1.2	0.6	0.9	1.5	1.1	0.3	1.3
V	–	–	–	–	–	–	0.2	1.0	0.7	0.8	0.7	0.9	1.4	0.2
σ	–	–	–	–	–	–	0.5	1.0	0.8	0.9	0.8	1.0	1.2	0.5

paper, and publication year. Searches are highlighted in dark gray, selected publications (based on selection and quality criteria as defined in the review protocol, cf. Section 4.1) are highlighted in light gray.

Appendix B. Requirements for product derivation survey

We used the following questionnaire in our requirements survey (cf. Section 5):

(**Question 1**)

How many years of experience do you have in the following fields?

Domain engineering: _____	Application engineering: _____
Requirements engineering: _____	Project management: _____
Sales/marketing: _____	Product management: _____

(**Question 2**)

How important do you rate the following requirements for product derivation in large-scale product lines comprising thousands of assets and variation points?

(−2... totally irrelevant|−1... unimportant|1... important|2... very important)

Support for resolving variability (*e.g., selecting/customizing assets by taking decisions/selecting features*)
Support for capturing and managing product-specific requirements not yet covered by the product line
Guidance and support for decision-making (*e.g., multimedia hints with recommendations and background information that explain decisions/features and provide rational for decision-making*)
Explicit support for domain experts (*e.g., sales and marketing people*)
Adaptability and flexibility (*e.g., adaptability to diverse domains, scalability to large models, openness of the tool environment*)
Interactivity and automation (*e.g., instantly presenting results after taking decisions/selecting features*)
Project management support (*e.g., the concept of tasks and roles for product derivation*)
Flexible visualization capabilities (*e.g., different variability visualizations for different users*)

(**Question 3**)

What other requirements do you regard as important?

Table 6 shows the results of our survey. Gray colored cells mark those questionnaires that do not fulfill the experience threshold of 1 year domain engineering and 3 years application engineering experience and that therefore have not been used for analyses. Additional requirements suggested by participants are not shown here (cf. Section 5.3).

Key: *Q*: questionnaire; *x*: mean/average; *V*: variance; σ: standard deviation

Question 1. DE: domain engineering, AE: application engineering, RE: requirements engineering, PM: project management, S/M: sales/marketing, PDM: product management.

Question 2. TS: tool support for resolving variability; ARE: support for application requirements management; GDM: guidance for decision-making; DE: support for domain experts; FAT: flexible and adaptable tools; IA: interactivity and automation; PM: project management support; FV: flexible visualizations.

References

[1] F. van der Linden, K. Schmid, E. Rommes, Software Product Lines in Action – The Best Industrial Practice in Product Line Engineering, Springer, Berlin/Heidelberg, 2007.
[2] A. Birk, G. Heller, I. John, K. Schmid, T. von der Maßen, K. Müller, Product line engineering: the state of practice, IEEE Software 20 (6) (2003) 52–60.
[3] P. Clements, L. Northrop, Software Product Lines: Practices and Patterns: SEI Series in Software Engineering, Addison-Wesley, 2001.
[4] K. Pohl, G. Böckle, F. van der Linden, Software Product Line Engineering: Foundations, Principles, and Techniques, Springer, 2005.
[5] K. Czarnecki, C.H.P. Kim, Cardinality-based feature modeling and constraints: a progress report, in: Proc. of the International Workshop on Software Factories at OOPSLA'05, ACM Press, San Diego, USA, 2005, pp. 1–9.
[6] H. Gomaa, Designing Software Product Lines with UML, Addison-Wesley, 2005.
[7] K. Schmid, I. John, A customizable approach to full-life cycle variability management, Journal of the Science of Computer Programming, Variability Management 53 (3) (2004) 259–284 (Special Issue).
[8] S. Deelstra, M. Sinnema, J. Bosch, Product derivation in software product families: a case study, Journal of Systems and Software 74 (2) (2005) 173–194.
[9] L. Hotz, K. Wolter, T. Krebs, S. Deelstra, M. Sinnema, J. Nijhuis, J. MacGregor, Configuration in Industrial Product Families – The ConIPF Methodology, IOS Press, 2006.
[10] M.L. Griss, Implementing Product-Line Features with Component Reuse, Springer, London, UK, 2000.
[11] P. O'Leary, R. Rabiser, I. Richardson, S. Thiel, Important issues and key activities in product derivation: experiences from two independent research projects, in: Proc. of the 13th International Software Product Line Conference (SPLC 2009), Software Engineering Institute, Carnegie Mellon, San Francisco, CA, USA, 2009, pp. 121–130.
[12] B. Kitchenham, P. Brereton, D. Budgen, M. Turner, J. Bailey, S. Linkman, Systematic literature reviews in software engineering – a systematic literature review, Information and Software Technology 51 (1) (2009) 7–15.
[13] G. Halmans, K. Pohl, Communicating the variability of a software-product family to customers, Informatik – Forschung und Entwicklung 18 (3–4) (2004) 113–131.
[14] B.A. Kitchenham, Guidelines for Performing Systematic Literature Reviews in Software Engineering, Version 2.3, EBSE Technical Report, EBSE-2007-01, Software Engineering Group, School of Computer Science and Mathematics, Keele University, Keele, Staffs, ST5 5BG, UK and Department of Computer Science, University of Durham, Durham, UK, 9 July, 2007.
[15] P. Brereton, B.A. Kitchenham, D. Budgen, M. Turner, M. Khalil, Lessons from applying the systematic literature review process within the software engineering domain, Journal of Systems and Software 80 (4) (2007) 571–583.
[16] R. Rabiser, A User-Centered Approach to Product Configuration in Software Product Line Engineering, PhD thesis, Institute for Systems Engineering and Automation, Christian Doppler Laboratory for Automated Software Engineering, Johannes Kepler University, Linz, Austria, 2009.
[17] E. Babbie, Survey Research Methods, Wadsworth, 1990.
[18] C. Wohlin, P. Runeson, M. Höst, Experimentation in Software Engineering: An Introduction, Kluwer International Series in Software Engineering, 2000.
[19] C.W. Krueger, The 3-tiered methodology: pragmatic insights from new generation software product lines, in: Proc. of the 11th International Software Product Line Conference (SPLC 2007), IEEE Computer Society, Kyoto, Japan, 2007, pp. 97–106.
[20] C. Krueger, BigLever software gears and the 3-tiered SPL methodology, in: Proc. of the Conference on Object Oriented Programming Systems Languages and Applications (OOPSLA'07), ACM, Montreal, Quebec, Canada, 2007, pp. 844–845.
[21] O. Díaz, S. Trujillo, F.I. Anfurrutia, Supporting production strategies as refinements of the production process, in: Proc. of the 9th International Software Product Line Conference (SPLC 2005), Rennes, France, Springer, Heidelberg/Berlin, 2005, pp. 210–221.
[22] M. Sinnema, S. Deelstra, Industrial validation of COVAMOF, Journal of Systems and Software 81 (4) (2008) 584–600.
[23] R. Rabiser, P. Grünbacher, D. Dhungana, Supporting product derivation by adapting and augmenting variability models, in: Proc. of the 11th International Software Product Line Conference (SPLC 2007), IEEE Computer Society, Kyoto, Japan, 2007, pp. 141–150.
[24] C. Atkinson, J. Bayer, D. Muthig, Component-based product line development: the KobrA approach, in: Proc. of the First Software Product Line Conference (SPLC-1), Kluwer Academic Publishers, Denver, CO, USA, 2000, p. 19.
[25] C. Atkinson, J. Bayer, C. Bunse, E. Kamsties, O. Laitenberger, R. Laqua, D. Muthig, B. Paech, J. Wüst, J. Zettel, Component-Based Product Line Engineering with UML, Addison-Wesley, 2002.
[26] 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

the 1999 Symposium on Software Reusability, Collocated with the 1999 International Conference on Software Engineering (ICSE'99), ACM Press, Los Angeles, CA, USA, 1999, pp. 122–131.

[27] T. Asikainen, T. Soininen, T. Männistö, A Koala-based approach for modelling and deploying configurable software product families, in: Proc. of the 5th International Workshop on Product-Family Engineering (PFE 2003), Siena, Italy, Springer, Berlin/Heidelberg, 2003, pp. 225–249.

[28] R. van Ommering, F. van der Linden, J. Kramer, J. Magee, The Koala component model for consumer electronics software, IEEE Computer 33 (3) (2000) 78–85.

[29] S.R. Faulk, Product-line requirements specification (PRS): an approach and case study, in: Proc. of the International Symposium on Requirements Engineering, IEEE CS, Toronto, Canada, 2001, pp. 48–55.

[30] J. Bayer, C. Gacek, D. Muthig, T. Widen, PuLSE-I: deriving instances from a product line infrastructure, In: Proc. of the 7th IEEE International Conference and Workshop on the Engineering of Computer Based Systems (ECBS), IEEE Computer Society, Edinburgh, Scotland, UK, 2000, pp. 237–245.

[31] J.D. McGregor, Preparing for Automated Derivation of Products in a Software Product Line, Technical Report CMU/SEI-2005-TR-017, 2005.

[32] K. Czarnecki, S. Helson, U.W. Eisenecker, Staged configuration using feature models, in: Proc. of the 3rd International Software Product Line Conference (SPLC 2004), Springer, Berlin/Heidelberg, Boston, MA, USA, 2004, pp. 266–283.

[33] K.C. Kang, S. Cohen, J. Hess, W. Nowak, S. Peterson, Feature-Oriented Domain Analysis (FODA) Feasibility Study, Technical Report CMU/SEI-90TR-21, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, USA, 1990.

[34] C. Cawley, D. Nestor, A. Preußner, G. Botterweck, S. Thiel, Interactive visualisation to support product configuration in software product lines, in: Proc. of the Second International Workshop on Variability Modeling of Software-Intensive Systems, Essen, Germany, ICB-Research Report, 2008, pp. 7–16.

[35] J.M. Hunt, Organizing the asset base for product derivation, in: Proc. of the Proceedings of the 10th International Software Product Line Conference (SPLC 2006), IEEE CS, Baltimore, MD, USA, 2006, pp. 65–74.

[36] M.-O. Reiser, M. Weber, Managing highly complex product families with multi-level feature trees, in: Proc. of the 14th IEEE International Requirements Engineering Conference (RE'06), IEEE CS, Minneapolis, MN, USA, 2006, pp. 149–158.

[37] J. Bayer, T. Lehner, D. Muthig, Product Line Engineering and Software Project Management, Fraunhofer IESE-Report No. 124.06/E, 2004.

[38] P.C. Clements, L.G. Jones, L.M. Northrop, J.D. McGregor, Project management in a software product line organization, IEEE Software 22 (5) (2005) 54–62.

[39] G. Chastek, P. Donohoe, J.D. McGregor, A Study of Product Production in Software Product Lines, CMU/SEI-2004-TN-012, 2004.

[40] L. Geyer, M. Becker, On the influence of variabilities on the application-engineering process of a product family, in: Proc. of the 2nd International Software Product Line Conference (SPLC-2), Springer, Berlin/Heidelberg, San Diego, CA, USA, 2002, pp. 1–14.

[41] T. Käkölä, J.C. Duenas, Software Product Lines – Research Issues in Engineering and Management, Springer, 2006.

[42] J. Bosch, G. Florijn, D. Greefhorst, J. Kuusela, H. Obbink, K. Pohl, Variability issues in software product lines, in: Proc. of the 4th International Workshop on Software Product-Family Engineering (PFE 2001), Springer, Berlin/Heidelberg, Bilbao, Spain, 2001, pp. 13–21.

[43] F. Loesch, E. Ploedereder, Optimization of variability in software product lines, in: Proc. of the 11th International Software Product Line Conference (SPLC 2007), IEEE CS, Kyoto, Japan, 2007, pp. 151–160.

[44] R. Rabiser, R. Wolfinger, P. Grünbacher, Three-level customization of software products using a product line approach, in: Proc. of the 42nd Annual Hawaii International Conference on System Sciences, IEEE CS, Waikoloa, Big Island, HI, USA, 2009, p. 10.

[45] R. Rabiser, D. Dhungana, Integrated support for product configuration and requirements engineering in product derivation, in: Proc. of the 33rd EUROMICRO Conference on Software Engineering and Advanced Applications (EUROMICRO-SEAA'07), IEEE Computer Society, Lübeck, Germany, 2007, pp. 219–228.

[46] R. Rabiser, Flexible and user-centered visualization support for product derivation, in: Proc. of the 12th International Software Product Line Conference (SPLC 2008), Second Volume, 2nd International Workshop on Visualisation in Software Product Line Engineering (ViSPLE 2008), Limerick, Ireland, Lero, 2008, pp. 323–328.