Industry Survey of Product Lines Management Tools: Requirements, Qualities and Open Issues

Olfa Djebbi, Camille Salinesi, Gauthier Fanmuy

Industry Survey of Product Lines Management Tools: Requirements, Qualities and Open Issues

Olfa Djebbi, Camille Salinesi, Gauthier Fanmuy

Abstract

PLM approaches are becoming a prominent approach in the Software Engineering and Systems Engineering PL contexts. The idea behind PLM is to focus on artifacts that are shared and that vary from one product to the other, so as to facilitate reuse and adaptation. Gains are expected in terms of time to market, consistency across products, easier identification of requirements for future products, costs reduction, better flexibility, and better management of change requirements. While most of the recent research works are focusing on methods and modeling techniques, little has been done so far with respect to PLM tools and their ability to answer industry needs. A study was thus undertaken in collaboration with a group of industrials to evaluate existing PLM tools. The purpose of the study was twofold: to understand the salient characteristics of PLM tools, and to evaluate the ability of existing tools to satisfy the expectations of industrials. The study was conducted using (a) a state of the art of PLM methods, (b) an analysis grid developed by the industrial partners to analyze the characteristics of RM tools in general, and (c) interviews with our industrial partners.

This paper reports our analysis under the form of a benchmark aimed at being used by industrials to select existing tools, and discusses open issues in the domain of RE for PL.

Résumé

Les approches de PLM (Product Line Management) deviennent de plus en plus courantes dans les contextes de l'ingénierie logicielle et de l'ingénierie des systèmes. L'idée derrière le PLM est de se concentrer sur les artefacts partagés qui varient d'un produit à l'autre, afin de faciliter la réutilisation et l'adaptation. On attend des avantages en termes de réduction du temps de mise sur le marché, de cohérence entre les produits, de facilité d'identification des exigences pour les futurs produits, de réduction des coûts, de meilleure flexibilité et de meilleure gestion des exigences de changement. Alors que la plupart des travaux de recherche récents se concentrent sur les méthodes et les techniques de modélisation, peu a été fait jusqu'à présent en ce qui concerne les outils de PLM et leur capacité à répondre aux besoins de l'industrie. Une étude a donc été entreprise en collaboration avec un groupe d'industriels pour évaluer les outils de PLM existants. L'objectif de l'étude était double : comprendre les caractéristiques saillantes des outils de PLM et évaluer la capacité des outils existants à satisfaire les attentes des industriels. L'étude a été menée en utilisant (a) un état de l'art des méthodes de PLM, (b) une grille d'analyse développée par les partenaires industriels pour analyser les caractéristiques des outils de RM (Requirements Management) en général, et (c) des entretiens avec nos partenaires industriels. Cet article rapporte notre analyse sous forme d'un benchmark destiné à être utilisé par les industriels pour sélectionner des outils existants, et discute des questions ouvertes dans le domaine de l'ingénierie des exigences pour le PL.

Industry Survey of Product Lines Management Tools: Requirements, Qualities and Open Issues

Olfa Djebbi[12], Camille Salinesi[1], Gauthier Fanmuy[3]
[1]*CRI, Université Paris 1 – Sorbonne, 90, rue de Tolbiac, 75013 Paris, France*
[2]*Satgo Instruments, 125 avenue Louis Roche, 92230 Gennevilliers, France*
[3]*PSA Peugeot-Citroën, Vélizy, France*
olfa.djebbi@malix.univ-paris1.fr, camille@univ-paris1.fr, odjebbi@stago.fr, gauthier.fanmuy@mpsa.com

Abstract

PLM approaches are becoming a prominent approach in the Software Engineering and Systems Engineering PL contexts. The idea behind PLM is to focus on artifacts that are shared and that vary from one product to the other, so as to facilitate reuse and adaptation. Gains are expected in terms of time to market, consistency across products, easier identification of requirements for future products, costs reduction, better flexibility, and better management of change requirements. While most of the recent research works are focusing on methods and modeling techniques, little has been done so far with respect to PLM tools and their ability to answer industry needs. A study was thus undertaken in collaboration with a group of industrials to evaluate existing PLM tools. The purpose of the study was twofold: to understand the salient characteristics of PLM tools, and to evaluate the ability of existing tools to satisfy the expectations of industrials. The study was conducted using (a) a state of the art of PLM methods, (b) an analysis grid developed by the industrial partners to analyze the characteristics of RM tools in general, and (c) interviews with our industrial partners.

This paper reports our analysis under the form of a benchmark aimed at being used by industrials to select existing tools, and discusses open issues in the domain of RE for PL.

1. Introduction

Product Line Engineering (PLE) is rapidly emerging as a viable and important systems development approach. Experience already shows that SPLE can allow companies to realize order-of-magnitude improvements in time to market, cost, productivity, quality and flexibility [1].

These new outcomes can be attributed to strategic software reuse. The basic principle in PLE techniques is to explicitly capitalize on commonality and formally manage the variations among products in the product line (PL). As a result, the main effort while designing a product relates to arbitrations that must be made with respect to variations.

Several approaches have addressed this concern. Code generation [2], components composition [3] and model transformation [4] are examples of techniques that address this issue at the implementation level. At the design level, several variability languages and methodological processes [5] [6] [7] have been defined to guide the specification of reusable assets of the PL, but little has been done so far with respect to PLM tools and their ability to answer industry needs.

Based on PLE frameworks [8] and MDD (model driven development) paradigm, the system specification and modeling steps lead to an effective technical reuse, especially for large and complex systems. When a PLE approach is adopted in a company, project managers are expected to establish an adequate working requirements management process. In this context, tools are important for an effective method deployment in the particular context of industrial scale PLs. Tools help automating many operations that can, if done manually, seriously hinder the quality and efficiency analysis of large scale systems. The need to select the "best" product line management (PLM) tool in a given context gets even more important in competitive environments where time to market, budget constraints and quality goals are crucial for achieving the company's business objectives. This is for instance the case in the automotive and medical industries [9] [10].

The purpose of this paper is to facilitate tool selection in the context of SPLE. As the purpose is specific and not general as in [maiden], we decided to proceed by an action research approach. This step belongs to a wider project that is to develop a derivation process for Stago products (medical company) from the PL specifications [11]. These specifications include requirements variability, hardware and software reference architecture variability, as well as the relationship between them. At this moment, the paper deals with a part of our research project. It tries to recommend an effective tool that enables the deployment of the future PLM method.

The process was: (i) to define expected objectives, (ii) to specify requirements to reach them, (iii) and finally, to evaluate tools based on the defined criteria.

The structure of the paper is as follows. Section 2 reviews the set of relevant and reliable requirements expected on variability tools. Section 3 presents our requirements elicitation and tools selection processes. Section 4 maps our findings to an evaluation grid. The paper concludes with a discussion of the results and an outlook to future work.

2. Evaluation approach

The approach is conceived to be as structured as possible. It aims to constitute a solid benchmark helping industrials to understand the salient characteristics of existing PLM tools, and to evaluate their ability to satisfy project managers' expectations. The purpose of the benchmark is to form a relevant reference, in witch both domain analyzers and product managers find their respective scientific and technical needs. In this context, a study was conducted by industrial practitioners and academic researchers. The concerns while designing the study were to (i) determine substantial criteria to be satisfied by PLM tools, (ii) search existing PLM tools, and (iii) evaluate them against defined criteria.

Three categories of tool selection criteria regarding PL tools were identified: we distinguish thus criteria relating to the Product Line Engineering itself, criteria relating to the tools capabilities and finally criteria concerning projects management. The table below (Table 1) presents criteria and provides a precise explication for each of them.

3. Analysis protocol

The evaluation was designed to be achieved in 4 different steps. Step (i) and (ii) are generic and thus can be achieved once for all, whereas steps (iii) and (iv) must be undertaken for every new tool selection.

The steps are as follows:

(i) State of the art: a comparative survey on PL methods and languages was elaborated [12]. The review showed that features are more widely used by industrials. That may be due to the simplicity of this notation and to the large amount of research on variability that was reported using it [5] [7]. The study helped to test feature notations capability to model industrial PLs.

(ii) Case studies: in order to get a better insight in the notations, case studies adapted from our companies contexts were developed using feature notations. The purpose was to illustrate features capabilities and to initiate discovery of industrial requirements regarding PLM tools.

(iii) Tools identification: a number of RE PL tools are available on the market. Tools were searched out from different sources such as journals and conferences papers [13] [14] [15] [16] [17], seminars [18] as well as web search engines [19] [20] [21].

The initial list comprises twelve tools that are: *Captain Feature* (successor of *AmiEddi*) [21], *Pure::Variants* [22], *Feature Plugin* [23], *SSEP toolset* [13], *DecisionKing* [15], *DOORS T-REK* [18], *XFeature* [24], *FMP* [17], *FORM/ASADAL* [25], *Gears* [20], *VarMod* [19] and *RequiLine* [26].

Some tools were abandoned based on their non-availability. Based on presentation papers and reports describing their functionalities, we also abandoned tools that we judged too insufficient from an industrial point of view. This first pre-selection was not based on the selection criteria defined in Table 1. However, it helped to get an initial knowledge on all tools and allowed reducing the number of tools to evaluate, to those that we believe are really likely to be used in industry. Remaining tools were then experimented with a small case study. More tools were abandoned ofter using them with the case study when we found they were too impractical.

(iv) Evaluation: only four tools were finally retained: RequiLine, Pure::Variants, XFeature and DOORS TREK. They were evaluated against the criteria grid.

The grid was adapted from a previous analysis grid developed by the industrial partners from AFIS[1] to

[1] Association Française d'Ingénierie Système (French association on System Engineering)
http://www.afis.fr/

Table 1. A grid of industry criteria for PL tools

N°	Criterion	Definition
Product Line Engineering criteria		
1	Attributes management	• The tool should help to manage requirements attributes (identifier, description, justification, cost…): multi-criteria sorting, sorting using logic operators …
2	Feature meta-modeling	• The tool should help to model FODA-like concepts such feature decomposition, feature type (mandatory, optional …), cardinalities, dependency links … • The tool should help modeling related PL concepts: feature modeling on several abstraction levels (external functions, internal functions…), relationships intra and inter -models, links with nomenclature (parts list)
3	Feature Metamodel maturity	• The tool should allow to define a PL metamodel • The tool should be unambiguous, should be well documented • The tool should have an updatable PL metamodel implementation • The tool should enable validation checking mechanisms : PL model/ PL metamodel consistency, application model/ PL metamodel consistency
4	Application modeling	• The tool should differentiate between PL and Product requirements • The tool should support product model / PL model consistency check • The tool should help determining the optimal number of variants for the following equation: (external functions variants * internal functions variants directly executable by components * nomenclature) • The tool should support requirements management in multi-projects (intra-project requirements vs. inter-project requirements)
Management criteria		
5	Traceability management	• The tool should support requirements traceability with external documents (Marketing Requirements Specification, nomenclature lists …) • The tool should support traceability management of inter-requirements links • The tool should support metamodel traceability: updating reports, decisions historisation …
6	Impact analysis	• The tool should implement impact analysis when changing a requirement, when changing the metamodel, when changing links inter-requirements • The tool should implement impact analysis following a requirements selection • The tool should support upstream and downstream impact analysis
7	Reporting	• The tool should be able to generate indicators, dash-board
Technical criteria		
8	Access mode	• The tool should allow multi-user access • The tool should allow access with profiles • The tool should allow user-rights generation by LDAP connection
9	Technical environment	• The tool should support synchronization • The tool should support disconnected mode, import/export • The tool should support data exchange via DB, APIs, neutral format files, specific developments
10	Usability	• It should be easy to take a grip on the tool, it should have an efficient support • The tool should offer a high accessibility of functions, zoom, views, queries • The tool should have a light charge of installation, maintenance and migration cost
11	Requirements and relationships presentation	• The tool should present requirements under the form of: Word processing, tables, forms, diagrams • The tool should allow report generation
12	Automatic filters	• The tool should implement automatic filters on requirements presentation • The tool should automatic filters on report generation
13	Alerts management	• The tool should allow manual and/or automatically alerts management • The tool should report alerts on logbooks, should edit alerts by colors … • The tool should enable parameterizeable management of alerts

analyze the characteristics of RM tools in general. Filters were applied on this initial grid to remove, in the context of our study, insignificant and irrelevant criteria. Based on the issues of two first steps and on expert interviews, new criteria relating to PL engineering were added.
Evaluation against the obtained grid was applied essentially by running tools, but also through tools documentation and demonstrations as well as discussion with vendors and users interviews.
Counter valuation of editors and a global review made by business experts helped to validate our results.

4. Evaluation Results

The previous section shows how and why some of the available PL tools were selected. This section makes a brief introduction on these tools and then evaluates them to determine to what extent they address the criteria presented in Table 1.

- XFeature [24] [14] was initially developed by P&P Software GmbH and the Automatic Control Laboratory of ETH-Zürich. It is currently extended by and used at ETH-Zürich in the context of the ASSERT project. The tool is provided as a plug-in for the Eclipse platform. The first version of the tool prototype was released in the summer 2005. The tool is available as free and open source software downloadable from the project web site. XFeature is a feature modeling tool that supports the modeling of product families features and of the applications instantiated from them. XFeature allows users to define their own feature meta-model.
- Pure::Variant [22] is a commercial PL tool also developed - by the company 'pure-systems GmbH' in Magdeburg, Germany - as an eclipse plug-in. The tool was first released and distributed in 2005. It allows modeling and visualizing PL in three forms: trees, diagrams and matrix. Besides, it uses Prolog to create constrains on features and provide consistency checking. Furthermore, Pure::Variant enables exchange with additional available plug-ins such TWiki for model elements, Bugzilla integration. It also grants synchronization with Borland CaliberRM and Telelogic DOORS, and access to version control systems such as CVS or Subversion.
- RequiLine [26] [16] (still under development) was created in 2005 by a research Group of the Computer Science Department at the RWTH Aachen University and is headed by Prof. Dr. H. Lichter. It is based on the .Net and Oracle/MySQL technologies. RequiLine enables users to model the product line using features and requirements and to derive product configurations from the specified model. It is based on the FORM metamodel. Additionnaly, it contains a consistency checker, a query interface, a user management with different views, and an XML interface for importing and exporting data.
- T-REK (Thalès/Telelogic Requirement Engineering Kit) [18] is a new product developped in collaboration by Télélogic and Thalès Research & Technologies as an add-on DOORS. It allows to manage the traceability of projects requirements, but also of PL requirements. It implements advanced functionalities to deal with requirement variability.

Table 2 shows the result of this evaluation. Four marks are used to express how a tool satisfies a criterion: A (++) denotes full support, (--) denotes no support, (+) denotes partial support and (-) denotes support with important restrictions.

Table 2. Tools comparative table

	Pure	**XF**	**RL**	**TREK**
PL engineering criteria				
1	++	- -	++	++
2	++	+	+	-
3	+	++	+	-
4	+	+	+	+
Management criteria				
5	- -	-	+	++
6	+	+	+	- -
7	++	- -	- -	- -
Technical criteria				
8	++	- -	++	++
9	++	-	++	++
10	+	-	+	+
11	+	+	+	+
12	+	- -	++	++
13	++	-	++	++

PL enginnering criteria

All tools rather exploit PL engineering concepts, except TREK that uses more general concepts. While all the other tools are built upon features, TREK uses notions such as Sideline criteria, Program, Component, Object, Family, Option, or Variant set that users should adapt to their own activity.

One can notice that despite the common background, tools do not implement the same feature metamodel. For example, TREK and RequiLine do not allow expressing feature set cardinalities.

On the other side, only XFeature implements classical feature concepts. Pure::Variants and RequiLine propose their own domain modeling concepts:
Pure::variants is based on two domain modeling concepts: Feature Models and Family Models. The

variability and the commonality of a product line from the customer or marketing perspective is captured in Feature Models that are then derived on Variant Description Models. Asset modeling is supported by Family Models describing the software from engineering perspective in terms of architectural elements. Family Models are then derived on a Variant Result Model by using the transformation engine that executes association rules.
In RequiLine, a domain model can be designed during requirement and features analysis. Features describe the core characteristics of the product that 'can be understood easily by client and developer and are used in order to draft the domain model' [16]. Each feature is then supplemented by a set of requirements. 'Requirements are concrete demands to a product that are very specific and detailed' [16]. Both features and requirements are modeled following FODA-like concepts.

Unfortunately, no explicit support is provided to model domain specific assets such as hardware resources.

Also, while all tools deal with intra-PL requirements, none supports really adequately inter-PL requirements management.

As for the feature metamodel maturity, XFeature is set apart. Indeed, in XFeature the meta-model is fully user-defined witch makes it an open and highly configurable tool.

At the application level, all tools provide validation features to check the consistency of the product model againt the PL model, and of the product and PL models against the PL metamodel. This is supported either through 'the correctness by construction', on demand checks or by automatic resolution of conflicting choices made at configuring products.

However, none of the four tools offers an advanced decision assistance for user, unless some consistency alerts arised using checking mecanisms. Operations like automatic derivation of the optimal feature combination or the identification of possible configurations starting from a feature pre-selection, are missing.

Management criteria

As shows the evaluation table, all tools are rather equivalent in addressing management criteria.

All tools except Pure::Variants offer the possibility to attach external documents to PL requirements.

Only RequiLine and TREK tools manage inter-requirements links traceability. Nevertheless, no one supports PL metamodel traceability since they do not allow metamodel updating, and neither does XFeature.

In all studied tools, impact analysis is strictly enabled through consistency checking. No advanced functionalities are available although such functionalities could be extremely useful for managers and analyzers.

In so far as reporting is concerned, only Pure::Variant implements it and offers a rich dashboard. Metrics are as various as: elements counting following complex criteria, separate counts for each kind of selection, implicit selections, average children per element, maximum tree depth, etc.

Technical criteria

Although RequiLine is an academic tool, it shows the same level quality than of industrial tools. Only XFeature shows some weaknesses; for example, XFeature is the only to be mono-user software.

Both Pure::Variants and TREK are built on a sound technical environment. The first enables disconnected mode, versioning, data exchange via a central repository (Oracle, SQL, PostgreSQL), synchronization with multiple tools (configuration management tools such as CVS, Subversion, requirements management tools that are DOORS, CaliberRM, Simulink, Database access and build manager tools) as well as importing/exporting data in XML, XMI, MSR-SW, MSR-FR and other user-definable formats.
The latter is also based on Oracle DB, enables disconnected mode and supports rich import/export interface such Excel, Word, XML, and other programmable interfaces supporting JavaScript and C#.

Over all, taking a grip on the tools was easy. Indeed, they presented a high accessibility to most of the useful functions and a good documentation support (in particular for Pure::Variants and RequiLine). Unfortunately, RequiLine has a long and heavy installation procedure and its user interface is not very nice.

Incidentally, RequiLine presents requirements in forms and diagrams, XFeature visualizes them in trees and diagrams; Pure::Variants also shows them under the form of matrixes. In TREK requirements are edited as free text.

Paradoxically, although Pure::Variants is technically very competitive, it does not support filters and report generation. TREK and especially Requiline offer advanced functionalities in filters, report generation and errors management. XFeature does not manage alarms for 'live validation' due to its fully user-defined metamodel.

5. Conclusion

PLM approaches are becoming prominent development approaches in Software Engineering and Systems Engineering PL contexts. Regarding this situation, tools get more important, particularly in competitive environments where deploying RM methods cannot be effective without adequate tools. Several PLM tools are available on the market to deal with this issue. This paper proposes a relevant benchmark to guide the selection of a suitable tool that satisfies stakeholders' requirements. The study was conducted near industrials and academics and refers to state of the art PL modeling methods and languages. First, selection criteria were defined based on technical and scientific industrial expectations. Then, existing PLM tools were evaluated against those criteria.

Tool selection depends on the context priorities. For example, XFeature was the recommended for Stago, a small company that seeks to capitalize its products commonalities and to manage their variabilities. In this case, technical capabilities of tools were neglected. Indeed, only one analyzer can deal with PL requirements. However, due to its complex systems, a high PL metamodel maturity was required. A user-defined PL metamodel would be an important facility to handle with Stago's PL specific concepts, namely multi-level requirements modeling and their covering by hard architectural components.
Discussions engaged with PSA Peugeot-Citroën requirements analyzers showed that Xfeature is not adapted to their activity. It is a large company where several teams must collaborate. So an effective requirements method can not be deployed without an effective tool support enabling data exchange and synchronisation between different working versions and between several management tools. Given that they already use DOORS, TREK was selected.

Our benchmark does not reveal a 'best tool'. On a general level, Requiline and Pure::Variants seem to be dominating the benchmark as they have comparable results, albeit the latter is more adapted to industrial use. However, as our two previous examples show it, another tool might be felt more adapted because it satisfies specific expectations of the enterprise. Once priorities are fixed, the benchmark presents a relevant mean helping managers selecting suitable tools.

The benchmark also helped underline some open issues. Detailed processes for requirements modelling and guidance for requirements derivation aren't enough studied in research community and aren't handled by PLM tools [11]. Besides, most tools implement former feature metamodels despite their ambiguities [12].

6. References

[1] http://www.sei.cmu.edu/productlines/plp_hof.html
[2] K. Czarnecki, U. Eisenecker, "Generative Programming: Methods, Tools, and Applications", Addison Wesley, 2000.
[3] T. Asikainen, T. Mnnist, T. Soininen, "Using a configurator for modelling and configuring software product lines based on feature models", SPLC, 2004.
[4] J. Garcia, M. Laguna, Y. Carvajal, B. Baixauli, "Requirements variability support through MDD and graph transformation", in Graph and Model Transformation, 2006.
[5] K. Kang et al., "Feature-Oriented Domain Analysis (FODA) Feasibility Study", Technical Report, SEI, 1990.
[6] M. Griss, J. Favaro, M. Allesandro, "Integrating Feature Modeling with RSEB", ICSR, Canada, 1998.
[7] M. Riebisch, D. Streitferdt, "Modeling Variability for Object-Oriented Product Lines", ECOOP, 2003.
[8] F. Linden, "Software Product Families in Europe: The Esaps & Café Projects", IEEE Software, 2002
[9] S. Buehne, K. Lauenroth, and K. Pohl, "Modeling Features for Multi-Criteria Product-Lines in Automotive Industry", ICSE 2004
[10] www.stago.fr
[11] O. Djebbi, C. Salinesi, "RED-PL, a Method for Deriving Product Requirements from a Product Line Requirements Model", CAISE, Norway, 2007.
[12] O. Djebbi and C. Salinesi, "Criteria for Comparing Requirements Variability Modeling Notations for Product Lines", workshop CERE in RE'06, USA, September 2006.
[13] D. Stuart, W. Sull, S. Pruitt, D. Cobb, F. Waskiewicz, and T. Cook, "The SSEP Toolset for Product Line Development: An XML Based Architecture Centric Approach", In Proceedings of SPLC, Colorado, 2000.
[14] V. Cechticky, A. Pasetti, O. Rohlik, W. Schaufelberger, "XML-Based Family Modelling", Software Reuse: Methods, Techniques, and Tools, Springer-Verlag, 2004.
[15] D. Dhungana, P. Grünbacher, R. Rabiser, "Decision King: A Flexible and Extensible Tool for Integrated Variability Modeling", VAMOS workshop, 2007.
[16] T. Maßen and H. Lichter, "RequiLine: A Requirements Engineering Tool for Software Product Lines", 2004.
[17] K. Czarnecki et al., "fmp and fmp2rsm: Eclipse Plug-Ins for Modeling Features Using Model Templates", OOPSLA, California, 2005.
[18] Seminar 'TREK for Product Lines', organized by Thales, Télelogic, Alcatel and Crescendo, Paris, 2007.
[19] http://www.sse.uni-due.de/wms/en/index.php?go=139
[20] http://www.biglever.com/
[21] https://sourceforge.net/projects/captainfeature/
[22] http://www.pure-systems.com/3.0.html
[23] M. Antkiewicz, K.Czarnecki, "FeaturePlugin: Feature Modeling Plug-In for Eclipse", OOPSLA, Canada, 2004.
[24] http://www.pnp-software.com/XFeature/
[25] K. Kim et al.,"ASADAL: A Tool System for Co-Development of Software and Test Environment based on Product Line Engineering", ICSE, 2006.
[26] http://www-lufgi3.informatik.rwth-aachen.de/TOOLS/requiline/