Selecting SPL Modeling Languages: a Practical Guide

Asmaa Achtaich, Ounsa Roudies, Nissrine Souissi, Camille Salinesi

Selecting SPL Modeling Languages: a Practical Guide

Achtaich Asmaa (*)(1), Roudies Ounsa (1)
(1) Univ. Mohammed V - Rabat,
EMI , SIWEB Team
Rabat, Morocco.
asmaaachtaich@research.emi.ac.ma,
roudies@emi.ac.ma

Souissi Nissrine (1) (2)
(2) Ecole Nationale Supérieure des Mines de Rabat
Computer Science Department
Rabat, Morocco
souissi@enim.ac.ma

Salinesi Camille
Université Paris 1 Panthéon – Sorbonne
Centre de Recherche en Informatique
Paris, France
camille.salinesi@univ-paris1.fr

***Abstract*—Software product lines engineering decreased the complexity of the development of products that share common features, and variability modeling helped define and manage the commonalities and differences between family products. That's why, through the years, many SPL languages have been proposed, tested, extended, experimented in case studies, and then developed even more. The proliferation of the modeling languages has made it difficult for engineers to select the appropriate one, depending on the domain context and on the user requirements. This paper first presents a panorama of the Software product line (SPL) modeling languages that have been proposed in the last two decades. A survey of few selected modeling languages is given in order to clarify their processes and the difference between the notations they use to specify requirements and to express commonality and variability. The article then provides software product line engineer with a guide that helps selecting the appropriate SPL modeling language, depending on the projects' constraints and requirements. The proposed practical guide is composed of a list of criteria that represent a basis for a comparative survey.**

Keywords— Software product lines; complexity; variability; features; language; domain; requirement; reusability

I. INTRODUCTION

Software product Line (SPL) engineering consists of designing and creating an assembly of elements sharing features, functionalities or common architectures, while meeting the needs of a specific category of stakeholders [1]. Both researchers and engineers are interested in this paradigm for it solves the complexity of the design and the development of software families. Reusability is a key concept in the SPL engineering: it captures the information available in the environment of the application and uses it in the development of related products while managing variability. Consistently with [2] [3] [4] [5] variability is defined in the rest of this paper, as the variation between systems belonging to a Product Line (PL) in terms of properties and qualities.

Since their introduction by Kang et al. [6], there has been a continuously growing number of SPL modeling languages. Through the years, several research papers and articles have reviewed and analyzed them from different aspects, and for different purposes. For instance, [34] proposes an evaluation framework for comparing Product Line Architecture design methods; [35] synthesizes and assesses the evidence regarding the effectiveness of proposed solutions, which resulted into a 12 categories comparative study; [13] presents a comparative framework for evaluating notations for requirements variability modeling. An evaluation of SPL modeling languages was held in [37], where Moreno-Rivera and Navarro report a systematic review (SR) of SPL approaches. Sinnema and Deelestra [38] introduce a classification framework of six variability-modeling techniques [39] provides an overview on the topics and trends of software variability management. Czanecki et al [41] compare feature models to decision models in a 10 dimension comparative study. Chen & Babar [42] propose a structured systematic literature review of variability management approaches [40] summarizes major research achievements in the field of SPL engineering and variability modeling using a standardized software product line framework.

This paper relies on these research papers to elucidate a wide range of selection criteria for SPL languages. The goal is to provide a guide for product families' engineers and developers that helps them choose a SPL modeling language according to a list of significant requirements defined along those criteria. We discuss and analyze how does each method scores, in terms of maturity, variability modeling, ease of use and update.

Section 2 describes the methodology and presents a panorama of SPL modeling languages. Section 3 presents the proposed practical guide for the selection of SPL modeling language. The paper concludes with a structured discussion on the strengths and weaknesses of each modeling language analyzed in the paper according to a group of criteria selected from the guide.

II. METHODOLOGY

The methodology we used to select SPL modelling languages and analyzed them is a mix of literature review and subjective qualitative analysis. As Figure 1 shows it, the practical guide was produced in 5 steps, the first 3 for selecting modeling languages, the last 2 to analyze them.

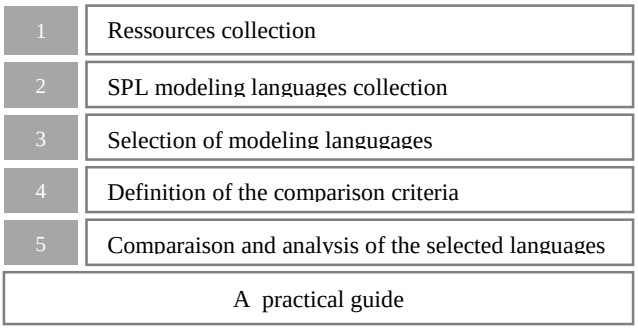

Figure 1 : Paper methodology

Conference and journal papers were selected from a wide range of computer science databases. The first goal was to identify primary studies of the software product lines modeling languages. During this activity we explored the IEEE Digital Library, ACM Digital Library, Science@Direct, MetaPress, Wiley InterScience, and Google Scholar. The work was not

978-1-4673-9669-1/15/$31.00 ©2015 IEEE

conducted as a systematic literature review as exhaustivity was not a major concern at this stage of the research project.

In order to get a meta view on SPL modeling languages, we selected papers that report state of art, literature (systematic or mapping) reviews, and surveys on SPLs as long as they were relevant to our preliminary reading and study. These papers were used as a reference (a) to detect a large collection SPL modeling languages, and (b) to get a first understanding on how they can be compared with each other.

The second step of our work aimed at describing representative modeling languages. In addition to the papers found in the earlier stage, PhD theses were used as input references. Our approach for filtering these documents were:

- We selected methods from different paradigms (Feature oriented, Object oriented and Family Oriented)
- We filtered the methods that had the most citations in state of art papers, surveys and literature review papers
- We selected methods that address both domain engineering and application engineering…

Last, we constructed a collection of comparison criteria for SPL modeling languages. 4 categories of criteria were defined: maturity, variability modeling, ease of use and update possibilities, and applied to the SPL modeling languages identified during the earlier stages of the work.

III. Survey of SPL Modeling Languages

In the past 25 years, more than SPL 50 modeling languages have been developed. Figure 2 presents a panorama of some of these notations in a 2 dimensional framework structured around time (vertical dimension) and conceptual proximity (horizontal dimension). We chose to put forward 6 particular languages that we found presentative either for their basic concepts, or for the engineering processes or tools that go along with them.

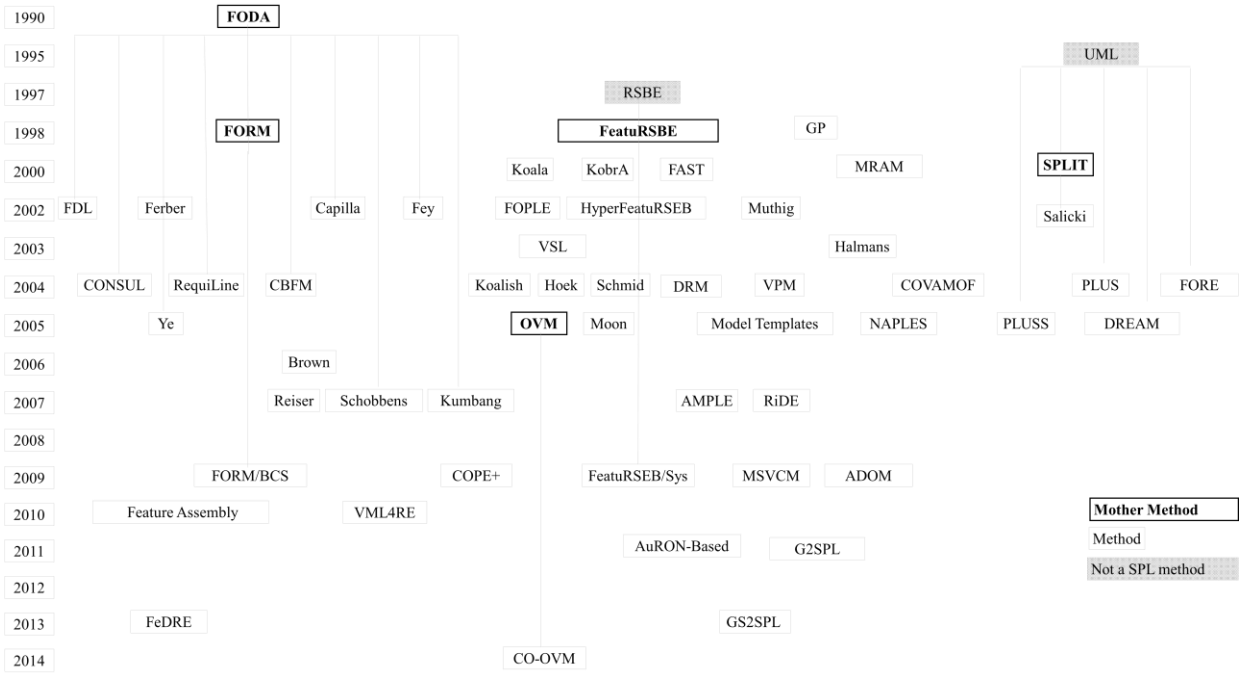

Figure 2 : SPL modeling languages

A. FODA

Feature Oriented Domain Analysis (FODA) is a domain analysis method [6] in which requirements are expressed under the form of features, which are used as the basic building blocks to specify similarities and differences between products of a family. In FODA features are used both to describe domains and to configure products. The extent of a domain is defined through domain analysis where the target collection of products is specified *by intent* in a structured diagram that combines features with dependencies.

Domain models make an explicit representation of some common features and all variable features of a family of products. Domain entities are selected from the application and domain knowledge, through a context analysis. They are then described in feature models with constraints that are specified either by dependencies described as feature-to-feature links, or textually when the graphical representation is not powerful enough to express them. In order to achieve this, variability and commonality are modeled in a functional model. Then, the structure of the software implementations is established in the architecture modeling phase, where a set of architectural models for building application are shown in the form of packages.

B. FORM

Feature Oriented Reuse Method (FORM) [7] is a method that focuses on capturing commonalities and differences in terms of features and uses the analysis results to develop architectures and domain components. Domain objects are

collected in domain engineering phase and can be prepared for reuse in the development of applications for a given domain in the architecture modeling phase.

C. FeatuRSEB

FeatuRSEB [8] is a method that combines FODA and RSEB (Reuse-Driven Software Engineering Business) by integrating the features model of FODA and the use cases of RSEB [9]. RSEB is a process that guides systematic reuse based on UML, which is used for modeling variability. The domain scoping and the choice and definition of the requirements are determined during the Domain Analysis phase. The process of FeatuRSEB includes the construction of a use case diagram for the product line simultaneously with feature model development. The use case diagram includes the list of the domain actors and the features model.

D. PLUSS

Product Line Use case modeling for System and Software Engineering (PLUSS) [10] is also an approach that combines the use of features models with use case diagrams to illustrate the point of view of a high-level product family. In PLUSS, all changes are managed using a common feature model that provides a complete overview of all the variability is the use case model. After collecting information through questionnaires or documentation, the requirements analysis comes, to describe the interactions between the system and actors. The product family requirements are captured in a use case diagram, and variability within this family is injected in the features model. Information concerning the types of features are then added, for example in use case scenarios.

E. ODM

Organization Domain Modeling (ODM) [11] focuses primarily on domain engineering for existing systems. However, it can be applied to the specification of requirements for new systems too. The engineering process of ODM starts with scoping and planning the domain. The possible combinations of features are specified in the domain modeling phase. First by capturing the semantics, then by modeling features that indicate commonalities and differences within a family of products, and finally by generating a list of profiles.

F. FAST

The purpose of Family-Oriented, Abstraction, Specification, and Translation (FAST) [12] is to make the software engineering process more effective by reducing the work redundancies and decreasing production costs and access times. Domain Engineer develops and acquires the basic assets of the product line, then the application engineer generates application systems that adapt to the customer needs.

The next chapter presents the propose guide. We first present our original list of criteria, then show how they can be used by comparing the 6 languages selected and presented above.

IV. THE PROPOSED GUIDE

The guide was constructed around a list of criteria similar to [13]. This list is the basis of our analysis and comparison; it discusses the maturity of the SPL modeling languages, the guidance of the modeling activity, the ease of use and the possibilities of evolution.

A. Criteria

The performance of a SPL modeling languages can be evaluated according to different perspectives. In the following, we elaborate a list of criteria that provides a common ground for comparing and analyzing these languages.

1) Category:Maturity

This category aims to indicate the maturity of a particular modeling language according to:

- Fields of application: to distinguish the ones designed for a specific area from more generic ones.
- Extensions: to determine if the modeling language aroused the interest of other researchers.
- Roots of the method: The maturity also depends on the research organization from which it originated. Some modeling languages are more developed than others because they are carried out by SPL specialists.

2) Category: Variability modeling

Variability is the keys concepts of SPLs, that is why the ways this variability is modeled by the different languages is something to take into consideration. In this category, we will shed some light on the following:

- Dependencies: The features of a product line are linked and each influences the behavior related features. Adding or changing these features must be managed so as not to affect the simplicity and relevance of the full model.
- Identification of variability points: The method uses diagrams or models in which the modeling of the variability is supported
- Conflict Resolution: The method must be able to manage conflicts between the requirements. And must be able to prove, at any time, the choice of a particular feature among others.

3) Category: Using the approach

The availability of information about a SPL is crucial for its comprehension, tool support is also needed for implementation. This category exposes:

- Readability: The feature model is a reference. It is consulted by the various stakeholders, thus, it should be easy to read, understand and apply.
- Simplicity: The user needs should be represented and comprehensible with the minimum of objects
- Technical Support: The modeling language must manage requirements and model the variability through mature tools.
- Documentation: The documentation is paramount in choosing the appropriate language. This manual should be consistent and clear in describing the language.

4) Category: Update

Due to its constant evolution, a SPL language has to take into consideration these parameters:

- Evolution: An SPL has a very dynamic nature, the requirements are constantly redefined. The template update method should be possible via the addition of new requirements if necessary.
- Adaptability: The model must be able to meet the specific needs of each context.
- Scalability: The model must allow for large scale systems modeling

B. Analysis and discussion

In the following chapter, we will compare FODA, FORM, FeatuRSEB, ODM, PLUSS and FAST. This comparison will be

based on maturity, variability modeling, ease of use and update. Figures 3, 4, 5 and 6 recapitulate the results.

The notations +++, ++, +, / and – carry the values 3, 2, 1, 0 and -1. The most advanced modeling language following a certain sub-criteria is rewarded with a +++, and so on until the least one represented with a -. The / is a neutral judgment, It is the equivalent of 0. The argumentation behind this notation is undertaken in the discussion.

1) Maturity

	Domaines of application	Extensions	Roots of the method	Global rates
FAST	++	/	+++	5
PLUSS	++	/	++	4
ODM	+	/	+++	4
FeatuRSEB	+	++	+++	4
FORM	++	+	++	5
FODA	+++	+++	+++	9

Figure 3 : Matrix of maturity

Since its initial development, the FODA method has boosted productivity and research in the field of product lines. The relevance of this modeling language and its ability to model, manage and make use of reusable elements has been shown through several projects [14] [15] [16] [17] [18] [19] [20] [21] [22]. Other methods have emanated from FODA. One of these methods is FROM, which fundamental observation is that FODA is very focused on domain engineering, but does not gie much details on actual reuse. FORM was also applied to several projects [23] [24] [25] [26].

FeatuRSEB is also a descendant of FODA. It was the first to introduce UML concepts in the expression of variability. Since its creation, in 1998, FeatuRSEB gave birth to several extensions including HyperFeatuRSEB and FeatuRSEB/sys, and also inspired researchers to complement features models and FODA notations by the UML diagram [27]. PLUSS was one of the many methods that came afterwards, and used UML notions to complete the expression of variability. This method was also applied to some projects [28] [29].

ODM was originally developed as a design phases of the Reuse Library Framework (RLF) [30], it was then developed and applied on a small project within Hewlett -Packard, as part of the Air Force CARDS program. This method was further refined at Unisys Corporation and was applied to the STARS program [31]. Several stakeholders contributed to the development of this method making it one of the most mature.

2) Variability identification and representation

FODA models variability through mandatory, optional and alternatives features, and describes the importance of each of these features via a “Rationale”, thus determining their priorities and justifying their selection. The "Rationale" is an information related to features under the form of an attribute, which allows for the selection or not of the feature in a given context. Conflict resolution through "Rationale" also takes place in the ODM, in the interpretation model, but is not considered in the remaining methods. The relations which connect the features of a product line are described in the features model. Vertical dependencies (Requires and exclude) that connect features belonging to different levels of the tree of the feature model are supported by FODA, FORM, FeatuRSEB

	Dependencies	Variability identification	Conflict resolution	Global rates
FAST	/	+	/	1
PLUSS	++	++	-	3
ODM	/	++	+	3
FeatuRSEB	+++	++	-	4
FORM	+++	+	-	3
FODA	++	+++	+	6

Figure 4 : Variability Matrix

and PLUSS. The horizontal dependencies which describe the relations between features that belong to the same level are operated by FORM (Composed of and generalization) and by FeatuRSEB (refinement). FODA is particularly suitable for the construction of reusable elements and modeling variability through the features and functional models. Features models are effective in modeling variability points, that’s why all the methods make use of is. It is often accompanied by additional diagrams. It has even been proven in [32] that the FODA original features diagrams called OFD, are the most complete and that the proposed extensions, even though complete the basic model, do not add much expressiveness.

The variability can also be expressed through UML diagram, like in FeatuRSEB and PLUSS. In FeatuRSEB, the variability points are expressed in the use case diagrams. A features model is connected to the UML diagram thought, offering a panoramic view of the SPL. In PLUSS, the features model provides a high-level view of the variability in the product family. FAST documents the variability in a written document.

3) Use of the approach

The FODA formalism is intuitive, precise and unambiguous, as confirmed by Gliss [33]. The great popularity of the method is in part due to its ease of use. All methods facilitate the expression of user needs and the overall understanding of the product line through diagrams with which the user is familiar, such as the feature model, the use case diagram or the concept model.

In terms of simplicity, the FODA features model is the simplest and lightest. FORM is simple, if not the lack of clarity in the mapping between the features model and the final architecture of the SPL. FeatuRSEB and PLUSS combine the use of features models with UML diagrams, something that facilitates modeling on one side and the understanding of it by

	Lisibility	Simplicit	Technical support	Documentation	Global rates
FAST	/	/	++	+	3
PLUSS	++	++	++	+	9
ODM	++	++	++	++	9
FeatuRSEB	++	++	++	+	9
FORM	++	+	++	++	7
FODA	+++	+++	++	+++	11

Figure 5 : Ease of use Matrix

the final user on the other side. Finally, ODM is very comprehensive but complex; several models are connected, one complementing the other which makes the reading of the overall a bit complicated. The use of the methods and their extensions is simplified by numerous support tools. They manage the features models, the features combinations and configurations. The available documentation for each method is a considerable asset. The better the process of a method is explained and detailed, the easier it is for a user to build the SPL. In our research, we found that FODA, FORM and ODM are clearly described, handy and very complete. The diagrams used in the modeling are defined and applied to some examples. Same goes for the engineering process followed and the key concepts.

FeatuRSEB, PLUSS and FAST are described in specialized articles, concise but short. The detail required for a proper assimilation is hardly collected.

4) Update

	Evolution	Adaptability	Scalability	Global rates
FAST	+	+	/	2
PLUSS	+	/	+	2
ODM	+	-	/	0
FeatuRSEB	-	-	+	-1
FORM	+	+	+	3
FODA	-	+	+	1

Figure 2 : Update Matrix

As features may change, features models are also subject to change by including new features or changing the nature of existing ones. FODA and FeatuRSEB do not keep track of the changes reported to the features model, evolution is thus not supported by these methods. In FORM, ODM and PLUSS, new features can be injected through additional models. In ODM for example, after completion the domain modeling, further measures of development can be injected to the integrated domain model. In PLUSS, the product instantiation is done by adding new requirements to the use case. In FAST, the evolution plan is deduced from the analysis of similarities.

For its setting, FODA and FORM keep a generic model, applicable to different contexts. The components are adapted by instantiating the parameter values. To be able to keep the same performance for all levels of magnitude, FODA and FORM allow vertical scalability, the latter method also supports horizontal scalability. In FeatuRSEB and PLUSS, scalability is supported because the configuration is not made in the use cases model, thereby avoiding congestion and allowing the modeling of large systems.

V. CONCLUSION

The number of SPL modeling languages has multiplied consequently, making the task of choosing a particular one very complex. The purpose of this paper is to provide the SPL engineer with a practical guide, which helps select the appropriate modeling language depending on his interests and final goals. Also, to help him in his search for existing work on SPL, we presented a panorama of more than 50 SPL modeling languages in their chronological appearance. 6 among them were filtered for a brief review: FODA, FORM, FeatuRSEB, ODM, PLUSS and FAST. For each method we described the global goal, the engineering process and the variability modeling diagrams.

Our practical guide composed of a 4 groups criteria: maturity, variability modeling, ease of use and update. Each one of these is once again composed of 3 to 4 criteria, subject to discussion. We analyzed and compared the selected methods cited above, and were able to point out the strengths and weaknesses following a distinct criterion, hence attributing for the SPL modeling language, a score according to a specific criterion. The outcome of this analysis is collected and represented in a Radar Chat, as presented in figure 7.

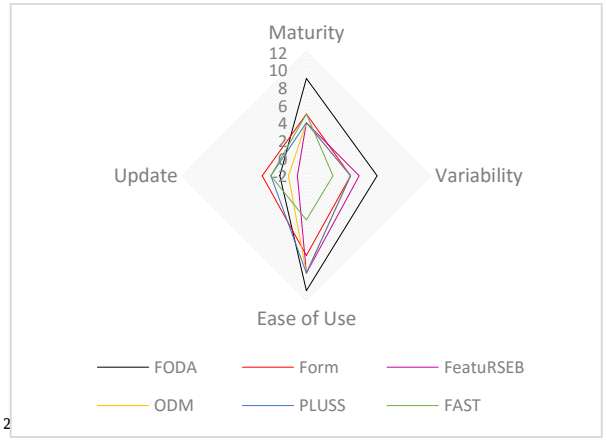

Figure 7 : Results of the analysis

Depending on the final user requirements and expectations, the SPL engineer can inject weights into a certain axis. For example if the user intends to evolve and expend the SPL over time, then the update axis takes over with a representative coefficient. If he wants to model variability using the lightest and most straightforward approach, in this case, the ease of use will be represented with the highest weight, and so on.

As a result of this work, the engineer can now easily choose methods from the panorama we presented in the first chapter, and compare them following our guide for better performances for the end user.

REFERENCES

[1] Donohoe, P. (2014). Introduction to Software Product Lines (Slides). Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst.

[2] Babar, M. A., Chen, L., & Shull, F. (2010). Managing variability in software product lines. Software, IEEE, 27(3), 89-91.

[3] Stephen Creff .(2013) Une modélisation de la variabilité multidimensionnelle pour une évolution incrémentale des lignes de produits. Software Engineering. Universit´e Rennes 1. French.

[4] Bosch, J., Capilla, R., & Hilliard, R. (2015). Trends in Systems and Software Variability. IEEE Software, (3), 44-51.

[5] Galster, M., Weyns, D., Tofan, D., Michalik, B., & Avgeriou, P. (2014). Variability in Software Systems—A Systematic Literature Review. Software Engineering, IEEE Transactions on, 40(3), 282-306.

[6] Kang, K. C., Cohen, S. G., Hess, J. A., Novak, W. E., & Peterson, A. S. (1990).Feature-oriented domain analysis (FODA) feasibility study (No. CMU/SEI-90-TR-21). Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst.

[7] Kyo C. Kang, Sajoong, K., Jaejoo, L., Kijoo, L., Euiseob, S. Moonhang, H. (1998) "FORM: A feature-; oriented reuse method with domain-; specific reference architectures." Annals of Software Engineering 5.1: 143-168.

[8] Griss, Martin L., John Favaro, and Massimo d'Alessandro. (1998) Integrating feature modeling with the RSEB. Software Reuse, 1998. Proceedings. Fifth International Conference on. IEEE.

[9] Jacobson, I., Griss, M., & Jonsson, P. (1997). Software reuse: architecture process and organization for business success (Vol. 285, p. 286). New York: acm Press..

[10] Eriksson, M., Börstler, J., & Borg, K. (2005). The PLUSS approach–domain modeling with features, use cases and use case realizations. In Software Product Lines (pp. 33-44). Springer Berlin Heidelberg.

[11] Simos, M., Creps, D., Klingler, C., Levine, L., & Allemang, D. (1996). Organization domain modeling (ODM) guidebook version 2.0.

[12] Weiss, D. M., "Commonality Analysis: A Systematic Process for Defining Families," Second International Workshop on Development and Evolution of Software Architectures for Product Families, 1998.

[13] Djebbi, O., & Salinesi, C (2006, September). Criteria for comparing requirements variability modeling notations for product lines. In Comparative Evaluation in Requirements Engineering, 2006. CERE'06. Fourth International Workshop on (pp. 20-35). IEEE.

[14] Zawoad, S., Mernik, M., & Hasan, R. (2014). Towards building a forensics aware language for secure logging. Computer Science and Information Systems, (00), 51-51.

[15] Georghiou, A. E., Davis, A., Eskandari, F., & Paulin, M. (2015, March). Extending Conventional Pipe-Soil Interaction Models to Include Bundle Effects for Arctic Subsea Pipeline Design. In OTC Arctic Technology Conference. Offshore Technology Conference.

[16] Kaur, P. K. (2014). Mobile Media SPL creation by Feature IDE using FODA.Global Journal of Computer Science and Technology, 14(3).

[17] Barreiro, P. S., García-Saiz, D., & Pantaleon, M. E. Z. (2014). Building Families of Software Products for e-Learning Platforms: A Case Study.Tecnologias del Aprendizaje, IEEE Revista Iberoamericana de, 9(2), 64-71.

[18] Karol, S., Heinzerling, M., Heidenreich, F., & Aßmann, U. (2010, September). Using feature models for creating families of documents. In Proceedings of the 10th ACM symposium on Document engineering (pp. 259-262). ACM.

[19] Ge, X., Paige, R. F., & McDermid, J. A. (2009, October). Domain analysis on an electronic health records system. In Proceedings of the First International Workshop on Feature-Oriented Software Development (pp. 49-54). ACM.

[20] Huron, M. A. (1997). The Army Tactical Command and Control System. Naval Postgraduate School Monterey CA.

[21] Vici, A. D., Argentieri, N., Mansour, A., d'Alessandro, M., & Favaro, J. (1998, June). FODAcom: an experience with domain analysis in the Italian telecom industry. In Software Reuse, 1998. Proceedings. Fifth International Conference on (pp. 166-175). IEEE.

[22] Mathias Filho, I., de Oliveira, T. C., & de Lucena, C. J. (2002). Domain Oriented Framework Construction. Enterprise Information III, 3, 171.

[23] Horváth, L., Rudas, I. J., & Tar, J. K. (2003, June). Robot assembly trajectory generation using form feature driven robot process model. In Industrial Electronics, 2003. ISIE'03. 2003 IEEE International Symposium on (Vol. 2, pp. 707-711). IEEE.

[24] Sarfraz, M. (Ed.). (2004). Geometric modeling: techniques, applications, systems and tools. Springer Science & Business Media.

[25] Choi, B. W., Jang, K. B., Kim, C. H., Wang, K. S., & Kang, K. C. (1999). Development of software for the hard real-time controller using feature-oriented reuse method and case tools. In Computer Aided Control System Design, 1999. Proceedings of the 1999 IEEE International Symposium on(pp. 126-131). IEEE.

[26] Roško, Z. (2014) Case Study: Refactoring of Software Product Line Architecture-Feature Smells Analysis. In Central European Conference on Information and Intelligent Systems (pp. 326 of 344)

[27] Yu, W., Zhang, W., Zhao, H., & Jin, Z. (2014, September). TDL: a transformation description language from feature model to use case for automated use case derivation. In Proceedings of the 18th International Software Product Line Conference-Volume 1 (pp. 187-196). ACM.

[28] Bosch, J. (2000). Design and use of software architectures: adopting and evolving a product-line approach. Pearson Education.

[29] Kollu, K. R. (2005). Evaluating The PLUSS Domain Modeling Approach by Modeling the Arcade Game Maker Product Line (Doctoral dissertation, Umeå University).

[30] McDowell, R., & Solderitsch, J. (1990, January). The Reusability Library Framework. In Proceedings of the Unisys Defense Systems Software Engineering Symposium.

[31] Simos, M., Creps, R., Klingler, C., & Lavine, L. (1995). Software Technology for Adaptable Reliable Systems (STARS). Organization Domain Modeling (ODM) Guidebook, Version 1.0 (No. STARS-VC-A023/011/00). Unisys Defense Systems Reston VA.

[32] Bontemps, Y., Heymans, P., Schobbens, P. Y., & Trigaux, J. C. (2004, August). Semantics of FODA feature diagrams. In Proceedings SPLC 2004 Workshop on Software Variability Management for Product Derivation–Towards Tool Support (pp. 48-58).

[33] Griss, M. , Favaro, J. d'Alessandro, M. (1997) Featuring the Reuse-Driven Software Engineering Business. CASE Methodologist.

[34] Matinlassi, M. (2004, May). Comparison of software product line architecture design methods: COPA, FAST, FORM, KobrA and QADA. In Proceedings of the 26th International Conference on Software Engineering (pp. 127-136). IEEE Computer Society.

[35] Babar, M. A., Chen, L., & Shull, F. (2010). Managing variability in software product lines. Software, IEEE, 27(3), 89-91.

[36] Ferré, X., & Vegas, S. (1999, June). An evaluation of domain analysis methods. In Proceedings 4th CAiSE Workshop on Exploring Modelling Methods for Systems Analysis and Design

[37] Moreno-Rivera, J. M., & Navarro, E. (2011, January). Evaluation of SPL approaches for WebGIS development: SIGTel, a case study. In System Sciences (HICSS), 2011 44th Hawaii International Conference on (pp. 1-10). IEEE.

[38] Sinnema, M., & Deelstra, S. (2007). Classifying variability modeling techniques. Information and Software Technology, 49(7), 717-739.

[39] Bosch, J., Capilla, R., & Hilliard, R. (2015). Trends in Systems and Software Variability. IEEE Software, (3), 44-51.

[40] Metzger, A., & Pohl, K. (2014, May). Software product line engineering and variability management: achievements and challenges. In Proceedings of the on Future of Software Engineering (pp. 70-84). ACM

[41] Czarnecki, K., Grünbacher, P., Rabiser, R., Schmid, K., & Wąsowski, A. (2012, January). Cool features and tough decisions: a comparison of variability modeling approaches. In Proceedings of the sixth international workshop on variability modeling of software-intensive systems (pp. 173-182). ACM.

[42] Chen, L., & Babar, M. A. (2011). A systematic review of evaluation of variability management approaches in software product lines. Information and Software Technology, 53(4), 344-362.