Search-Based Migration of Model Variants to Software Product Line Architectures

Wesley Klewerton Guez Assunção*†
*DINF - Federal University of Paraná, CP: 19081, CEP: 81531-980, Curitiba, Brazil
†COINF - Technological Federal University of Paraná, CEP: 85902-490, Toledo, Brazil
Email: wesleyk@inf.ufpr.br, Website: http://www.inf.ufpr.br/wesleyk/

***Abstract*—Software Product Lines (SPLs) are families of related software systems developed for specific market segments or domains. Commonly, SPLs emerge from sets of existing variants when their individual maintenance becomes infeasible. However, current approaches for SPL migration do not support design models, are partially automated, or do not reflect constraints from SPL domains. To tackle these limitations, the goal of this doctoral research plan is to propose an automated approach to the SPL migration process at the design level. This approach consists of three phases: detection, analysis and transformation. It uses as input the class diagrams and lists of features for each system variant, and relies on search-based algorithms to create a product line architecture that best captures the variability present in the variants. Our expected contribution is to support the adoption of SPL practices in companies that face the scenario of migrating variants to SPLs.**

***Index Terms*—Reuse; Migration; Re-engineering; Software Product Line; Search-Based Software Engineering.**

I. Introduction

There is an increasing demand for software in all segments of the society. Rather than developing systems from scratch, software engineers reuse artefacts, libraries, etc. A common ad hoc reuse strategy is referred to as *clone-and-own* methodology [1], where existing products are copied to be the basis of new variants which are then modified. But when the number of variants increases, their individual maintenance becomes a complex, if not infeasible, activity. In such cases, a systematic reuse approach is paramount. *Software Product Line Engineering (SPLE)* is a software development paradigm that has been proven effective for this scenario [2]. The goal of SPLE is to produce a *Software Product Line (SPL)*, which is a set of software products that share a set of common features [3]. A *Product-Line Architecture (PLA)* is the main artefact of an SPL and provides the design that is common to all the products of the SPL, hence it describes all the mandatory and varying features of its SPL domain [4].

SPLE can be undertaken using different strategies [5]. The *extractive approach* uses existing systems as the starting point and hence it is more suitable for companies with many systems already in production. The migration process of the extractive strategy is composed of three phases [6]: *(i) identification phase*, where the relevant information such as traceability links between features and artefacts is gathered; *(ii) analysis phase* where the similarities and variabilities are identified; and *(iii) transformation phase* where the common and variable parts of an SPL are realized in the corresponding artefacts.

To gather information on the migration of system variants into SPLs, we performed a systematic mapping study [7]. We found a growing interest in this research topic. However, we identified the following gaps that our work will address:

1) *Limitations on the artefacts used*. Most of the research has focused on source code artefacts and the use of design models, in particular within the context of *Model-Driven Software Development (MDSD)* [8], has not been extensively explored.
2) *Limitations related to the migration process*. Many approaches require extensive human expertise or lack adequate tool support. In addition, most studies focus on specific phases of the migration process, but in practice it must be considered in its entirety.
3) *Limitations related to feature management*. Feature management is challenging because of the typical large number of possible feature combinations in an SPL. Current approaches generally do not consider domain constraints among features, which hinders the consistent instantiation of products with certain properties such as minimal cost and maximum benefit.

Software architecture has an important role in software development because it works as a bridge between the requirements and the implementation. In the context of SPLs, PLAs enable the variability management in the design process and provide a high level of abstraction for their understanding. Because the critical role PLAs play in SPL development, and the limitations we identified, the goal of the doctoral research is to find whether it is possible to automate the extraction of a PLA from a set of variant models associated to existing systems while considering the domain constraints of an SPL.

The migration process necessary to achieve this goal is clearly a complex activity. For example, in the analysis phase many possible feature combinations and their properties must be considered. Problems like this have been successfully addressed in the *Search-Based Software Engineering (SBSE)* [9] field, where complex software engineering problems are tackled by using search-based algorithms. SBSE techniques offer

ways to efficiently explore large and complex search spaces characteristic of SPLs [9]. For this reason we chose to investigate the use of SBSE techniques to tackle our goal.

II. Proposed Approach

Our goal is to provide complete and automated support for the entire migration process, that encompasses the phases of identification, analysis, and transformation. In this section, we first describe the input required by our approach, the steps it consists of, and the output it produces. In addition, we present the plan to evaluate our work.

A. Input, Steps, and Output

The input of our approach is a set of existing model variants. Each variant consists of two parts: *(i)* a set of *UML class diagrams*, and *(ii)* a set of *feature sets* which denote the combination of features provided by the variant. Figure 1 presents an example with three product variants. For instance, Product 2 has classes `A`, `D` and `E`, and realizes features `F1` and `F4`. Because we aim to support MDSD, we chose class diagrams as they are the most common type of models [10].

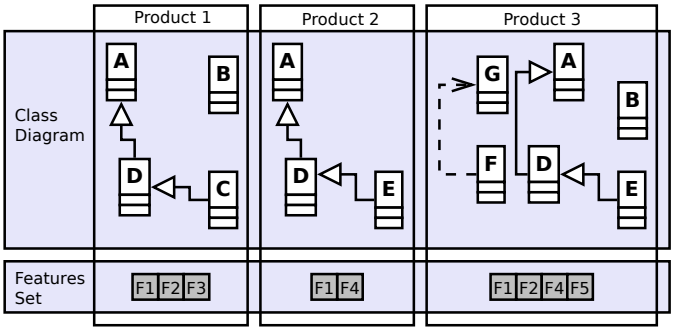

Fig. 1. Example of input for the approach

Figure 2 presents the three steps, one for each migration phase, of our approach. We now describe each step in more detail.

1) Step 1 - Feature Traceability: The goal of this step is to automatically extract traceability links between each feature or feature interaction and the elements in the class diagram that realizes them. This step corresponds to the identification phase of the migration process. Furthermore, relevant information, such as the relationships (e.g. dependencies) existing among the elements that implement the features, becomes explicit. This information is considered in the next step. To carry out this task, we propose a solution based on the work of Linsbauer et al. [11], but using the class diagram instead of the source code. In the work of Linsbauer et al. [11] the traceability links are extracted by matching source code overlaps and feature overlaps. Our proposal is to use class diagrams instead of source code. For example, considering Figure 1, matching the features sets and diagrams of products 1 and 2 it is possible infer that feature `F1` is realized by class `A`, class `D`, and the relationship between them. To the best of our knowledge, our work is the first approach that studies

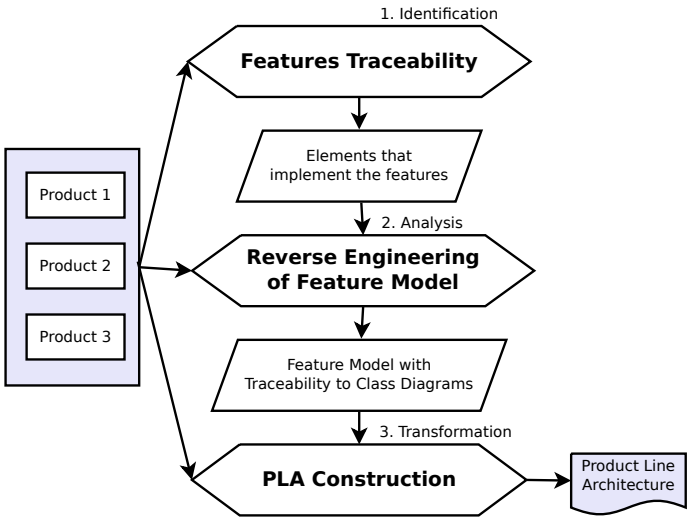

Fig. 2. Proposed approach for migrating systems to SPL

the extraction of traceability links between class diagrams and features in multiple existing variants with the goal to obtain an SPL.

2) Step 2 - Reverse engineering of Feature Models: Feature Models (FMs) are the de facto standard to represent all valid combinations of features of SPLs [12]. The goal of this step is to reverse engineer a FM from the set of features sets, and the traceability links to the class diagrams extracted in the first step. This is a complex and error prone task to be performed manually, for instance it is very frequent to derive FMs that contain many more combinations than those required by the software engineer. An approach to obtain FMs from features sets is the work by Linsbauer et al. [13] which relies on genetic programming, a search-based technique, to reverse engineer FMs using information retrieval metrics as objective functions. We plan to expand on this work to consider traceability information. This expansion consists on identifying the relationships among elements that realize different features and to use them in the search-based technique. The goal is to assess the value of the traceability from features to the class diagrams for the reverse engineering of FMs. Our motivation to use traceability information comes from the work of Peng et al. [14], who argue that the structure of the artefacts should resemble the structure of features. In addition, having a FM that closely resembles the structure of the traced artefacts, class diagrams in our case, may simplify the transformation phase. The output of this step is then the derived FM and the traceability links from the first step.

3) Step 3 - PLA construction: The goal of this step is to generate a class diagram with SMarty notation [15] that corresponds to a PLA for the model variants. The proposed strategy is to generate PLAs using multi-objective evolutionary optimization techniques. The basis for this step is our previous work [16]. We plan to study similarity measures to find PLAs that have the most similar structure with all the model variants considered as input. At the end of the search process, a set of

PLAs with a good trade-off between the similarity measures is expected to be available for the user. Figure 3 presents an PLA constructed using the input presented in Figure 1.

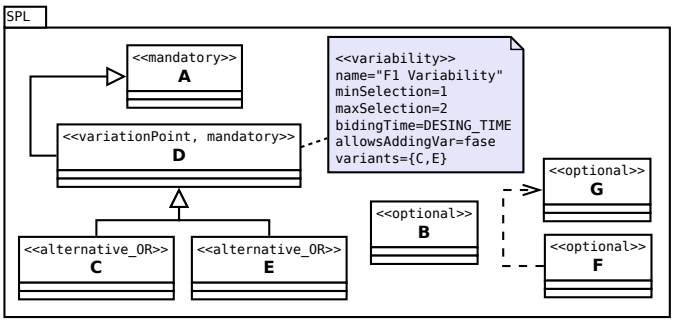

Fig. 3. Example of class diagram with SMarty notation

B. Evaluation Plan

We currently plan to use variants from five systems [17]: Draw Product Line (18 variants), Video On Demand (32 variants), ArgoUML (256 variants), ZipMe (32 variants), and Game Of Life (65 variants). The Java source code of these systems variants is available. The class diagrams of each variant will be obtained from source artefacts using tools such as Astah UML[1]. With the class diagrams of each variant, we plan to reverse-engineer a PLA for each system.

The success criteria of the approach consist in obtaining a PLA that enables an easy migration of the existing systems to a SPL. To asses this, we will use similarity measures such as topological similarity and semantic similarity to evaluate how similar is the created PLA to the existing class diagram variants. Moreover, we consider to use other quality metrics, such as *Variability model metrics* proposed by Berger and Guo [18] and *Design metrics* available in tools like SDMetrics[2].

III. Expected Contributions

The proposed plan tackles several practical and open issues that are relevant for the adoption of SPLs in industry. The main expected contributions are:

- *Support the SPL migration process at the design level.* By supporting the migration process at the design level, software engineers and architects can attain better visualization and understanding of SPLs, potentially reducing the effort needed for performing evolution and maintenance tasks. This is so because such changes are made at a higher level of abstraction than the current practice at the source code level. Our contribution is to support the entire migration process of class diagrams of system variants into PLAs. This type of diagrams are among the most commonly used to model systems [10], so by using them we expect to reach a wider use for our approach.
- *Study of search-based techniques for SPL migration.* Search-based techniques can efficiently explore large search spaces and achieve optimal, or near optimal, solutions and have been studied in the context of SPLs [9]. However, their use for SPL migration at design level has not been explored. We expect that our work will address this research gap and in addition support software engineers in their decision making process by providing them with multiple and alternative solutions from which they can select the best options according to their migration context.
- *Support for managing domain constraints.* Products instantiated from SPLs must not only satisfy constraints that are represented in FMs (e.g. valid feature combinations) but also they must satisfy constraints imposed by the artefacts that realize their features and feature interactions (e.g. dependence calls in source code). Our contribution is that we consider both sources of domain constraints and use them to guide the search in SBSE techniques, in particular for creation of FMs and PLAs.

IV. Related Work

Our systematic mapping [7] revealed a few studies that propose search-based techniques for the SPL migration process. Segura et al. [19] propose the application of an evolutionary algorithm to optimize the generation of FM. These authors took into account computational resources, such as execution time or the memory consumption, to generate computationally-hard FMs. Lopez-Herrejon et al. [20] extended this work to reverse engineer FMs from feature sets of existing variants. Linsbauer et al. [13] also addressed this reverse engineering task but instead they employed information retrieval metrics to define an objective function and genetic programming. None of these two studies consider other models such as class diagrams. Lohar et al. [21] addressed only the identification phase using search-based techniques to explore variability model. Ali et al. [22] used a multi-objective algorithm to *miniaturize* software so that it fits into different resource-constrained hardware platforms. In this case, only the identification and analysis phases are considered. Ullah et al. [23] used a genetic algorithm that considers business and technical perspectives. In contrast with our work, their focus was on migrating a single product to a portfolio of products, and they supported identification and analysis phases. To the best of our knowledge, our work is the first to propose using search-based algorithms encompassing the three SPL migration phases.

V. Work Progress

This doctoral work began on September 2012. Initially, two studies were performed whose goal was to use search-based strategies to recover similar class diagrams [24], [16]. These studies served as a way to plan how to represent and deal with class diagrams using optimization algorithms with focus on reuse. Then we conducted a systematic mapping on the subject of migrating systems into SPL [7]. This study helped us to identify gaps and open opportunities in this research area which became the foundation of the doctoral proposal.

[1]http://www.astah.net/
[2]http://www.sdmetrics.com/

Currently, from September 2014 to July 2015, I am a visiting doctoral student hosted by Prof. Alexander Egyed, the head of the Institute for Software Systems Engineering (ISSE) at Johannes Kepler University (JKU) in Linz, Austria. During this stay, our plan is to develop a tool automating Steps 2 and 3 of the proposed approach. For Step 2, we are currently conducting a study on using multi-objective search-based algorithms for the reverse engineering FMs considering dependencies between artefacts. After this study and to achieve Step 1, we plan to extend current work on traceability extraction, developed by the hosting group, for dealing with class diagrams. Step 3 will be the focus of the last leg of the stay at JKU. We should comment that the order of addressing our research steps was scheduled to align with the host institute research goals and with the intended publications calendar.

For the last year of the doctoral work, August 2015 to June 2016, we plan to perform a thorough and robust empirical evaluation of the proposed approach. The final reporting of the approach details, experimental evaluation, results and analysis, will materialize in the doctoral dissertation that I expect to defend towards the end of 2016.

VI. Publications

In the first stage of the doctoral work two papers have been published. The first paper [24] presents a comparison between a Genetic Algorithm and a Particle Swarm Optimization using a mono-objective solution to retrieve class diagrams from repositories composed with design models of open source software. The second paper [16] presents a multi-objective solution with two similarity measures as objectives. The initial results of the systematic mapping are in [7]. An extended journal version of this study is under development. We expect the first results of Step 2 to be submitted for publication in January 2015.

Acknowledgment

The author would like to thank Brazilian Federal Agency for the Support and Evaluation of Graduate Education (CAPES) for the scholarship in Brazil and the financial support for the internship abroad, my advisor Prof. Silvia Regina Vergilio, my co-advisor Dr. Roberto Erick Lopez-Herrejon, and Prof. Alexander Egyed.

References

[1] K. Pohl, G. Böckle, and F. J. v. d. Linden, *Software Product Line Engineering: Foundations, Principles and Techniques*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2005.

[2] F. J. v. d. Linden, K. Schmid, and E. Rommes, *Software Product Lines in Action: The Best Industrial Practice in Product Line Engineering*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2007.

[3] P. Clements and L. Northrop, *Software Product Lines: Practices and Patterns*. Boston, MA, USA: Addison-Wesley, 2001.

[4] J. Contieri, AntonioC., G. Correia, T. Colanzi, I. Gimenes, J. Oliveira, EdsonA., S. Ferrari, P. Masiero, and A. Garcia, "Extending uml components to develop software product-line architectures: Lessons learned," in *Software Architecture*, ser. LNCS, I. Crnkovic, V. Gruhn, and M. Book, Eds. Springer Berlin Heidelberg, 2011, vol. 6903, pp. 130–138.

[5] C. W. Krueger, "Easing the transition to software mass customization," in *Software Product-Family Engineering*. Springer, 2002, pp. 282–293.

[6] V. Anwikar, R. Naik, A. Contractor, and H. Makkapati, "Domain-driven technique for functionality identification in source code," *SIGSOFT Software Engineering Notes*, vol. 37, no. 3, pp. 1–8, May 2012.

[7] W. K. G. Assunção and S. R. Vergilio, "Feature location for software product line migration: A mapping study," in *2nd International Workshop on Reverse Variability Engineering, 18th International Software Product Line Conference*. New York, USA: ACM, 2014, pp. 52–59.

[8] M. Völter, T. Stahl, J. Bettin, A. Haase, and S. Helsen, *Model-driven software development: technology, engineering, management*. John Wiley & Sons, 2013.

[9] M. Harman, Y. Jia, J. Krinke, B. Langdon, J. Petke, and Y. Zhang, "Search based software engineering for software product line engineering: a survey and directions for future work," in *18th International Software Product Line Conference*, 2014, pp. 1–14.

[10] U. Fahrenberg, M. Acher, A. Legay, and A. Wasowski, "Sound merging and differencing for class diagrams," in *Fundamental Approaches to Software Engineering*, ser. Lecture Notes in Computer Science, S. Gnesi and A. Rensink, Eds. Springer, 2014, vol. 8411, pp. 63–78.

[11] L. Linsbauer, E. R. Lopez-Herrejon, and A. Egyed, "Recovering traceability between features and code in product variants," in *17th International Software Product Line Conference*, ser. SPLC 2013. New York, NY, USA: ACM, 2013, pp. 131–140.

[12] K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak, and A. S. Peterson, "Feature-oriented domain analysis (foda) feasibility study," Carnegie-Mellon University Software Engineering Institute, Tech. Rep., November 1990.

[13] L. Linsbauer, R. Lopez-Herrejon, and A. Egyed, "Feature model synthesis with genetic programming," in *Search-Based Software Engineering*, ser. Lecture Notes in Computer Science, C. Le Goues and S. Yoo, Eds. Springer International Publishing, 2014, vol. 8636, pp. 153–167.

[14] X. Peng, Z. Xing, X. Tan, Y. Yu, and W. Zhao, "Improving feature location using structural similarity and iterative graph mapping," *Journal of Systems and Software*, vol. 86, no. 3, pp. 664–676, 2013.

[15] E. A. O. Junior, I. M. Gimenes, and J. C. Maldonado, "Systematic management of variability in uml-based software product lines," *Journal of Universal Computer Science*, vol. 16, no. 17, pp. 2374–2393, 2010.

[16] W. K. G. Assunção and S. R. Vergilio, "A multi-objective solution for retrieving class diagrams," in *Brazilian Conference on Intelligent Systems*, 2013, pp. 249–255.

[17] S. Fischer, L. Linsbauer, R. E. Lopez-Herrejon, and A. Egyed, "Enhancing clone-and-own with systematic reuse for developing software variants," in *30th International Conference on Software Maintenance and Evolution*, 2014.

[18] T. Berger and J. Guo, "Towards system analysis with variability model metrics," in *8th International Workshop on Variability Modelling of Software-Intensive Systems*. New York, USA: ACM, 2014, pp. 1–8.

[19] S. Segura, J. Parejo, R. M. Hierons, D. Benavides, A. Ruiz-Cortés, and E. d. l. J. de Andalucía, "ETHOM: An evolutionary algorithm for optimized feature models generation," Applied Software Engineering Research Group, Department of Computing Languages and Systems, University of Sevilla, Tech. Rep. ISA-2012-TR-01, 2012.

[20] R. E. Lopez-Herrejon, J. A. Galindo, D. Benavides, S. Segura, and A. Egyed, "Reverse engineering feature models with evolutionary algorithms: An exploratory study," in *Search Based Software Engineering*, ser. Lecture Notes in Computer Science, G. Fraser and J. Teixeira de Souza, Eds. Springer Berlin Heidelberg, 2012, vol. 7515, pp. 168–182.

[21] S. Lohar, S. Amornborvornwong, A. Zisman, and J. Cleland-Huang, "Improving trace accuracy through data-driven configuration and composition of tracing features," in *9th Joint Meeting on Foundations of Software Engineering*. New York, USA: ACM, 2013, pp. 378–388.

[22] N. Ali, W. Wu, G. Antoniol, M. Di Penta, Y. Guéhéneuc, and J. Hayes, "Moms: Multi-objective miniaturization of software," in *27th IEEE International Conference on Software Maintenance*, 2011, pp. 153–162.

[23] M. I. Ullah, G. Ruhe, and V. Garousi, "Decision support for moving from a single product to a product portfolio in evolving software systems," *Journal of Systems and Software*, vol. 83, no. 12, pp. 2496–2512, 2010.

[24] W. K. G. Assunção and S. R. Vergilio, "Class diagram retrieval with particle swarm optimization," in *25th International Conference on Software Engineering and Knowledge Engineering*, 2013, pp. 632–637.