2018 International Conference on the Quality of Information and Communications Technology

A Systematic Reuse-based Approach for Customized Cloned Variants

Karam Ignaim, João M. Fernandes
Dep. Informatics, Universidade do Minho
Braga, Portugal
{e6870@alunos, jmf@di}.uminho.pt,

André L. Ferreira, Jana Seidel
Bosch Car Multimedia Portugal
Braga, Portugal
{Andre.Ferreira2@pt, Jana.Seidel@de}.bosch.com

***Abstract*—Systematic reuse often becomes possible only after a number of customized cloned variants have already been delivered. Transforming from customized cloned variants to a systematic reuse with the explicit management of variability is beneficial. Hence, industrial companies prefer to adopt a reuse-based approach. We propose in this research work an approach that supports the re-engineering of existing customized cloned variants towards systematic software reuse. The approach also eases the process of adding a new variant to a set of customized cloned variants, whenever there is enough implementation similarity between the existing customized cloned variants and the new one. We plan to evaluate our approach in an industrial case study, specifically in a set of customized cloned variants of software applications used by automotive companies. As an initial validation effort, we already have presented our first results to software developers at Bosch. They provided us a positive feedback about the ability of our approach to give an overview of the commonality and the variability of the customized cloned variant. For more robust validation, we propose to use a structured demonstration for the same industrial case study environment with qualitative and quantitive evaluation of the impact.**

Keywords— cloning, systematic reuse, variant, variability, commonality, Feature Model

I. INTRODUCTION

Some companies need to handle customized variants that have some characteristics in common [1]. Generally, companies apply an approach called cloning, in which a new customized cloned variant is built by copying and adapting existing customized cloned variants [2, 3]. Cloning requires no major upfront investments and is natural, which makes it common for industrial use [2]. Unfortunately, cloning does not favor reuse. The developing organization faces a choice between building reuse-based systems from scratch or transforming the existing customized cloned variants into a systematic reusable form [2, 4, 5]. A systematic reuse-based approach can be introduced, but it requires explicit information about commonality and variability over the customized cloned variants [4, 6, 7]. Moreover, it needs a mapping between features and artifacts (e.g. requirement document or code base) into customized cloned variants [3, 8–11].

Feature Models (FMs) are one of the most popular abstraction forms for modeling commonality and variability of customized cloned variants [9, 12]. FMs are used broadly to help generate and validate individual variant configurations (Fig. 1) and to provide support for domain analysis [9, 10]. However, a successful transformation is challenging, since it requires precise and detailed information about the distribution of implementation similarity and difference between the product variants. This information is usually not available, as the product variants were modified independently of each other [4]. Our approach organizes the discovered variability in a design model, which is called Variability Design Model (VDM). This model is a design that expresses variations between customized variants. The purpose of our approach is to support systematic reuse for a set of customized cloned variants by delivering detailed similarities and differences among them. For that purpose, we study customized variants related to the sensor-based software product family, these variants are cloned and customized by the developers at Bosch company. Our research contributes to the work related to the systematic reuse of a customized cloned variant, by providing the following characteristics:

- A feature-based design for a set of customized cloned variants at requirement and design level.
- A difference analysis specific for requirement level of customized cloned variants.
- An approach to identify commonality and variability among customized cloned variants.
- A novel mapping method to trace features to their place in the implementation code through requirements specification of customized cloned variants.

The contributions of our approach are reflected in the research questions, shown in Table I. In an industrial case study, we asked the participating developers to assess the impact/influence of the models and information produced by our approach on the available set of customized cloned variants. The feedback has shown that our approach was appreciated as an explicit way to express variability and to go through a systematic reuse of variants family.

II. STATE OF THE ARTE

In this section, we summarize the state of the art of the field addressed by our research. Thus, we review some of the previous methods, techniques, and tools that tackle the problem of systematic reuse. Systematic reuse often takes place only

978-1-5386-5841-3/18/$31.00 ©2018 IEEE
DOI 10.1109/QUATIC.2018.00051

after a number of family variants have already been delivered. Practically, a new variant is often created by cloning the code of an existing variant and changing it according to the new requirements. For example, in [1], authors conducted an empirical study involving three companies and analyzed in detail the development activities these companies perform on existing cloned variants. Then, they provided activities to support the transitions to a structured Software Product Line (SPL) based approach. In the short term, cloning is a common and simple way to create a new customized variant. A novel approach to enhance cloned variants is proposed in [2]. The authors evaluated the approach on six case studies, and they covered 402 variants. As a result, they reached an excellent percentage of composed variants reuse.

Analysis of customized cloned variants for systematic reuse requires precise and detailed information about how implementation similarities are distributed among the customized variants. In [3, 4], authors support a research work close to our research. They support techniques and tools to identify similarities, which can be identified to be common for all variants. However, in our research work, we propose an approach that identifies not only the similarities, but also the differences among customized cloned variants. Seven variability mechanisms are characterized and compared in [13]. As a variability mechanism, authors in [14] present a novel feature-oriented programming approach to migrate multiple cloned variants into an SPL.

FMs have been widely used to model commonality and variability in the context of variant families. The key requirement for using FMs is to derive a product configuration that satisfies all business and customer requirements [5]. The authors of this research work present their own tool, called SPLConfig, to support product configuration in Software Product Line.

A novel approach named SPLEVO for supporting the consolidation of customized product copies into a Software Product Line is presented in [6]. Several researchers tackle the problem of extracting FMs or variability of existing customized variants [7, 12, 13, 15–18]. Several kinds of artifacts can be considered, including variants descriptions [9, 10, 19–21], models [22, 23], and code bases [8, 24–26] or the combination of them. The approach under development is suitable for variability management focusing on artifacts resulting from requirements engineering and code base implementations practices.

Existing software reuse approaches often develop and evolve new variants independently, which makes it difficult to manage the relationship between features or variability to other artifacts. Many researches present approaches and tools that support mapping features to variant artifacts [8] [12], specifically to the source code [11] [25].

Initially, we evaluated our approach on a sensor-based software product family. The variants of this family are cloned and customized in the software development department at Bosch Company [27–32]. We investigated and analyzed in detail the development activities performed by the software team. In addition, we access the software variants artifacts to apply our approach.

A systematic literature review was conducted by [33] to assess research quality and to identify research trends, open problems, and areas for one of the most important reused-based approach (i.e. SPL). In this research, authors concluded that there is a clear need for conducting studies comparing alternative methods for SPL development. Moreover, they recommended more future research to invest in tool support and in SPL adoption strategies.

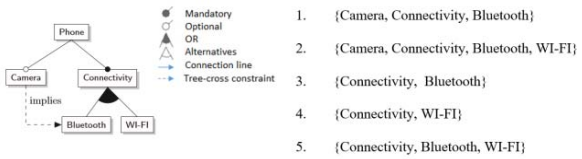


Figure 1. Feature Model and valid individual variant configurations.

III. RESEARCH OBJECTIVES AND METHODOLOGICAL APPROACH

The main objective of this research proposal is to define a reuse-based approach that is practicable in transforming customized cloned variants towards systematic reuse. To fulfill the main objective, we aim to provide an extractive approach (1) to support an explicit management of commonality and variability, and (2) to derive an FM and a VDM for a set of customized cloned variants. In addition, we aim to provide a reactive approach (3) to evolve the current set of available customized cloned variants with a new one. A secondary objective is to identify and trace the variability, which is scattered across all the customized cloned variant code bases. To work towards this objective, we plan to provide an approach that can trace features to their places in the requirements and source code, and update the FM to maintain its coherence when features changes occur.

TABLE I. RESEARCH QUESTIONS FOR OUR RESEARCH PROPOSAL

RQ1	How can we transform customized cloned variants into a systematic reuse- based approach?
RQ2	How can we analyze and identify similarities and differences between independent customized cloned variants?
RQ3	How can we derive FM and VDM, which support explicit variability over the customized cloned variants?
RQ4	What is an acceptable approach to extract commonality and variability of the customized cloned variants?
RQ5	How can we map a feature to its place(s) in the implementation code bases?

Regarding the input artifacts, variants specifications (i.e., requirement specifications) and code bases are the most important inputs for our approach. In addition, variants family architecture is taken into consideration, since it contains variability information. Different types of artifacts are produced, including FMs, VDM and traceability matrix (which indicates the relations among features and code bases). As shown in Fig. 2, our approach has three major phases:

(1) reverse engineering, (2) forward engineering, and (3) mapping.

A. Reverse Engineering

This phase compares the requirement specifications of two variants to detect the similarities and the differences between them and then uses this information to derive a FM and a VDM. These models support an explicit management of variability for the family variants.

B. Forward Engineering

This phase evolves the set of customized cloned variants with a new variant upon a new customer request. It is supposed to receive the specific requirements for the new customer and then update the overall FM, in order to include the new features that relate to the new customized cloned variant.

C. Mapping

In the context of programming, the problem space relates to the requirements or needs of a domain and describes the features provided by variants family from a customer perspective. The solution space relates to the implementation of variants and describes the variability in the code bases from the perspective of developers [34]. Based on these definitions, our approach maps requirements and domain space analysis (i.e., FM) into the problem space. Additionally, it maps code bases (i.e. software elements) to the solution space. This phase helps to trace features to their locations in the implementation code. The novelty in this phase is that it maps each feature to its implementation in the code bases through the software requirements. This phase delivers the traceability matrix (a model that maps the features to the code bases).

To evaluate our approach, we will apply it to an industrial case study related to a variants family of automotive sensors. These variants are currently cloned and customized by software team to satisfy the needs of different car manufacturers [27]. We have investigated and analyzed in detail the development activities, which are performed there. Developers use configuration management branching techniques to create a new customized cloned variant to support a customer request [28, 29, 32]. The branching approach, which is mainly based on cloning and customization of the variant in order to satisfy customer needs, has many benefits [30, 31]. At the same time, it has some challenges, which are summarized in Table II. In addition to this, we are going to use a structured demonstration technique to assess the usefulness of our approach, specifically, to preserve the benefits and overcome the challenges of the approach adopted at Bosch.

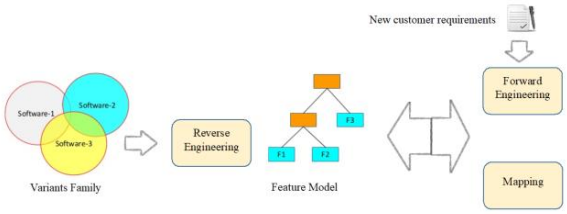


Figure 2. The three mais phases of our approach.

IV. PAST WORK AND PRELIMINARY RESULTS

In this section, we present work and results of our research. This work satisfies our aim to support an explicit management of commonality and variability (i.e. almost phase one of our approach). We have derived FM and a partial VDM for the customized cloned variants. Gradually, different studies on reverse engineering to derive FMs from different artifacts of customized cloned variants were presented [9, 10, 20, 21]. Many of the existing approaches are designed to reverse engineering FM from high-level models, such as variant description and requirements [35].

For systematic reuse of customized cloned variants, domain analysis consists in identifying similarities and differences among the family variants [20]. The explicit identification of commonality and variability is the starting point for systematic reuse of family variants. One of the most efficient and popular ways to explicitly present this is by using FM. FM is a domain abstraction for program functionality and at the same time, it is a compact way to define all features and their valid combinations [24, 25].

The first phase of our approach derives FM and VDM from artifact related to the customized cloned variant, specifically the requirement specifications. Our approach identifies commonality and variability in requirement specifications, which are written in natural language (English) and may contain some additional information in tabular formats. Phase 1 of our approach is divided into steps as next discussed:

Step 1. Define transformation rules to rewrite requirement specifications for each customized cloned variant in single requirement line statements (i.e. Requirement Variant / No. Document). Also, normalize the input of tabular data and facilitate its interpretation.

Step 2. Identify common and different parts among requirements for each variant using difference analysis. Similarities and differences are extracted initially from some customized cloned variants, specifically from their Requirement Variant / No Documents. Our approach uses text-based comparison and natural language processing to identify similarities and differences among variants artifacts [6].

Step 3. Similarities represent commonality among customized cloned variants and differences represent variability. As a result, mandatory features appear on the common requirements, and optional features appear on the varied requirements. Our approach relates the requirements that represent one feature each other. In addition, the difference between requirements represents a variation point among customized cloned variants. A variation point identifies one or more locations at which the variation occurs [15].

Step 4. Finally, variability and commonality are represented in terms of features, such as root node, mandatory/optional features, parent features, and an alternative/OR group of features (Fig. 3 and Fig. 4). In addition, constraints are

extracted. Our approach uses the variation points to derive a VDM.

TABLE II. BENEFITS AND CHALLENGES OF THE BRANCHING APPROACH

Benefits	Challenges
Fast and low effort in the creation of a new customized cloned variant.	Lower maintainability of the new customized cloned variant resulting code. Since maintaining, the duplicated code involves repetitive tasks.
Branched a new customized cloned variant lead to no risk to the already existing customized cloned variants.	Porting similar problems, which will be carried over from one new customized cloned variant to another.
There is no need to coordinate modifications and configurations, which were introduced to the new-branched variant to other variants. This reduces the coding effort.	This approach takes some time considering branching, freezing, updating branched variant, checking and testing.
In case of testing fails, the new customized cloned variant can be easily modified due to its independence from other customized cloned variants. This means low development effort.	The developer would harm the platform software with customer related code in case of wrong or miss step in branch handling.

Fig. 5 shows an example of a variation point in the VDM. We adopt the technique presented in [5] and adapt it to work with requirements specifications. Using our approach, we have derived five relevant features (Fig. 3) and other sub-features (Fig. 4) from classic sensor customized cloned variants [a] resulting in a systematic domain abstraction of the sensor customized cloned variants. Up to now, we have derived a VDM that refers to requirements implemented by their variation points. Our approach can use this initial design to build VDM that refers to features implemented by their variation points (Fig. 5). Making result available for the classic sensor developers, we expect it can be used to give a comprehensive understanding of the classic sensor domain space and to help them to transform into the systematic reuse-based approach. Moreover, our approach will use this result during the progress of our research work.

This section presents the part of our research work that already has been performed using phase 1 of our approach. This phase receives the Variant Specifications, as an input, and produces the FM and, partially, the VDM.

V. FUTURE WORK AND EXPECTED RESULTS

The research work and the results of the previous section satisfy some but not all of our objectives and research questions. We answered partially the RQ1, RQ2, and RQ3. There are several lines of research work still need to be addressed. Thereby, as a future work, we plan to incrementally use other customized cloned variants until we derive FM that covers a set of the available customized cloned variants. Moreover, we plan also to use the derived FM to evolve the set of available customized cloned variants with new variants. Finally, software developers noted that annotation of features in the code bases to be a time- consuming and repetitive task. Therefore, and to justify RQ4 and RQ5, we intend to investigate our proposed method to map features to their implementation on the code bases and to produce the traceability matrix. For more robust validation, we are going to evaluate our approach among available variants family of the classical sensor, for that, we use a structured demonstration with qualitative and quantitive evaluation of the impact.

Our approach is perceived as integrated successfully within the existing customized cloned variants based on the decision of both the software project manager and software developer team. We aim to satisfy them with a systematic reuse-based approach that is flexible enough to customize the intended product variants and to provide a manageable amount of configurations. The expected results of our future research work are:

- Practicable systematic reuse-based approach for customized cloned variants.
- FMs that cover a set of customized cloned variants.
- Defined concept to evolve the FM with a new variant.
- VDM that identifies separate difference points to support continues variability management.
- Mapping/Tracing features to code bases through requirements specification of customized cloned variants. Moreover, for usability, it is important to store this information in the traceability matrix.

CONCLUSIONS

In this manuscript, we present an approach for supporting the reuse of customized cloned variants in software engineering. We propose a systematic reuse-based approach that delivers part of our research work. To evaluate our approach, we have conducted a case study related to a sensor-based software product family. We have successfully derived an FM and a partial VDM for some customized cloned variants. Our future work will be focused on improving the results even further considering the execution of a second structured demonstration case study by adding more customized cloned variants.

ACKNOWLEDGMENT

We thank our colleagues from Bosch company in sensor software team (João Santos and Hélder Vilas-Boas) and we also like to show our gratitude to Matthias Renninger (Development, Mangement of projects) for his assistance. We thank the European Structural and Investment Funds in the FEDER component, through the Operational Competitiveness and Internationalization Programme (COMPETE 2020) Project nº 002797; Funding Reference: POCI-01-0247-FEDER-002797.

[a] For readability, we split the FM into Fig. 3 and 4.

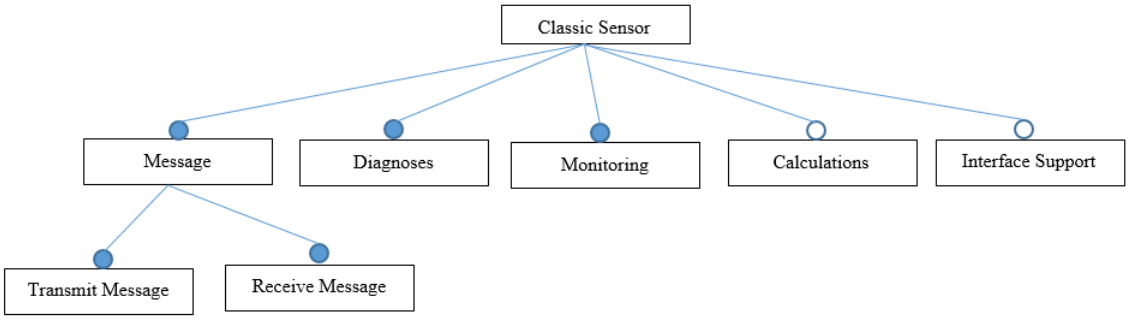


Figure 3. Feature Model for classical sensor variants family.

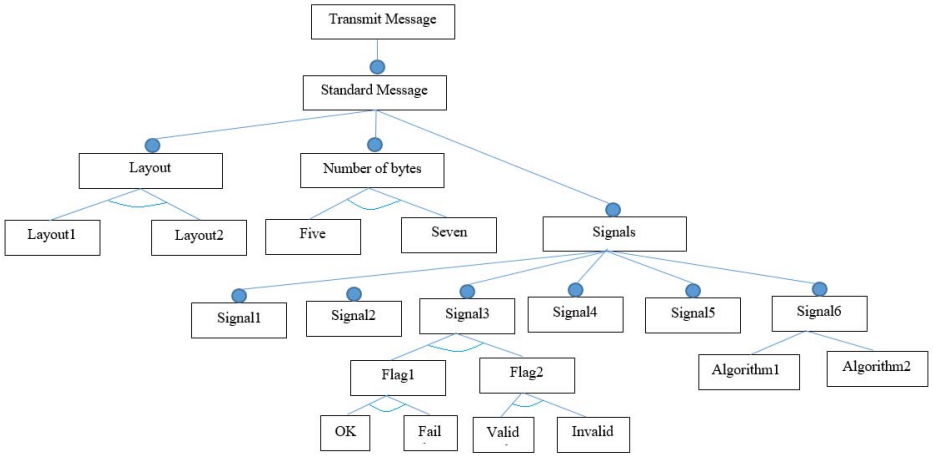
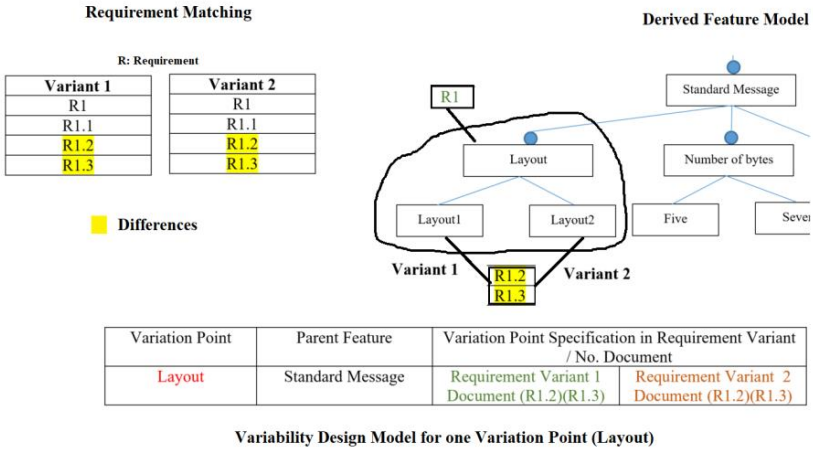


Figure 4. Feature Models for classical sensor variants family.



Variation Point	Parent Feature	Variation Point Specification in Requirement Variant / No. Document	
Layout	Standard Message	Requirement Variant 1 Document (R1.2)(R1.3)	Requirement Variant 2 Document (R1.2)(R1.3)

Variability Design Model for one Variation Point (Layout)

Figure 5. Example of variation point in the Variability Design Model.

REFERENCES

[1] J. Rubin, K. Czarnecki, and M. Chechik, "Managing cloned variants: A framework and experience," in *17th International Software Product Line Conference (SPLC 2013)*, ACM, pp. 101–110, 2013.

[2] S. Fischer, L. Linsbauer, R. E. Lopez-Herrejon, and A. Egyed, "Enhancing clone-and-own with systematic reuse for developing software variants", in *IEEE International Conference on Software Maintenance and Evolution (ICSME 2014)* , IEEE, pp. 391–400, 2014.

[3] T. Mende, F. Beckwermert, R. Koschke, and G. Meier, "Supporting the grow-and-prune model in of software product lines evolution using clone detection," *in 12th European Conference on Software Maintenance and Reengineering (CSMR 2008)* , IEEE, pp. 163–172, 2008.

[4] S. Duszynski, "*Analyzing similarity of cloned software variants using hierarchical set models*," Ph.D dissertation, University of Kaiserslautern, Germany, 2015.

[5] L. Machado, J. Pereira, L. Garcia, and E. Figueiredo, "Splconfig: Product configuration in software product line," in *Brazilian Congress on Software (CBSoft),* pp. 1–8, 2014.

[6] B. Klatt, "*Consolidation of customized product copies into software product lines*," KIT Scientific Publishing, 2016.

[7] J. Martinez, T. Ziadi, T. F. Bissyande, J. Klein, and Y. Traon, "Bottom-up technologies for reuse: Automated extractive adoption of software product lines", in *39th International Conference on Software Engineering Companion (ICSE 2017)*, IEEE, pp. 67–70, 2017.

[8] L. Linsbauer, R. E. Lopez-Herrejon, and A. Egyed, "Recovering traceability between features and code in product variants," in *17th International Software Product Line Conference (SPLC 2013)*, ACM, pp. 131–140, 2013.

[9] J. Davril, E. Delfosse, N. Hariri, M. Acher, J. Cleland-Huang, and P. Heymans, "Feature model extraction from large collections of informal product descriptions", in *9th Joint Meeting on Foundations of Software Engineering (FSE 2013)*, ACM, pp. 290–300, 2013.

[10] M. Acher, A. Cleve, G. Perrouin, P. Heymans, C.Vanbeneden, P. Collet, and P. Lahire, "On extracting feature models from product descriptions," in *6th International Workshop on Variability Modeling of Software-Intensive Systems*, ACM, pp. 45–54, 2012.

[11] Y. Zheng, C. Cu, and H. U. Asucion, "Mapping features to source code through product line architecture: Traceability and conformance," in *IEEE International Conference on Software Architecture (ICSA 2017)*, IEEE, pp. 225–234, 2017.

[12] K. Czarnecki, P. Grünbacher, R. Rabiser, K. Schmid, and A. Wąsowski, "Cool features and tough decisions: A comparison of variability modeling approaches," in *6th International Workshop on Variability Modeling of Software-intensive Systems*, ACM, pp. 173–182, 2012.

[13] B. Zhang , S. Duszynski, and M. Becker, "Variability mechanisms and lessons learned in practice", in *IEEE/ACM International Workshop on Variability and Complexity in Software Design (VACE 2016),* IEEE, pp. 14–20, 2016.

[14] W. Fenske, J. Meinicke, S. Schulze, S. Schulze, and G. Saake, "Variant-preserving refactorings for migrating cloned products to a product line", in *Software Analysis, Evolution and Reengineering (SANER)* , pp. 316–326, IEEE, 2017.

[15] D. L. Webber and H. Gomaa, "Modeling variability in software product lines with the variation point model," *Science of Computer Programming* 53(3):305–331, 2004.

[16] J. Bosch, R. Capilla, and R. Hilliard, "Trends in systems and software variability [Guest editors introduction]," *IEEE Software* 32(3):44–51, 2015.

[17] V. Alves, P. Matos Jr. , L. Cole, P. Borba, and G. Ramalho, "Extracting and evolving mobile games product lines," in *International Conference on Software Product Lines (SPLC 2005)*, Springer, pp. 70–81, 2005.

[18] M. V. Couto, M. T. Valente, and E. Figueiredo, "Extracting software product lines: A case study using conditional compilation," in *European Conference on Software Maintenance and Reengineering (CSMR 2011),* IEEE, pp. 191–200, 2011.

[19] N. Bakar, Z. Kasirun, and N. Salleh, "Feature extraction approaches from natural language requirements for reuse in software product lines: A systematic literature review," *Journal of Systems and Software* 106:132–149, 2015.

[20] F. Wanderely, D. S. Silveira, J. Araujo, and M. Lencastre, "Generating feature model from creative requirements using model driven design," in *16th International Software Product Line Conference-Volume 2*, ACM, pp. 18-25, 2012.

[21] N. Itzik and I. Reinhartz-Berger, "Generating feature models from requirements: Structural vs. functional perspectives," in *18th International Software Product Line Conference (SPLC 2014)*, ACM, pp. 44–51, 2014.

[22] H. Casalánguida and J. Durán, "Automatic generation of feature models from UML requirement models," in *16th International Software Product Line Conference (SPLC 2016),* ACM, pp. 10–17, 2012.

[23] J. Martinez, T. Ziadi, T. F. Bissyande, J. Klein, and Y. Le Traon, "Automating the extraction of model-based software product lines from model variants," in *30th IEEE/ACM International Conference on Automated Software Engineering (ASE 2015),* IEEE, pp. 396–406, 2015.

[24] Y. Xue, Z. Xing, and S. Jarzabek, "Feature location in a collection of product variants," in *19th Working Conference on Reverse Engineering (WCRE 2012)*, IEEE, pp. 145–154, 2012.

[25] R. F. Al-Msie'Deen, A. Serial, M. Huchard, C. Urtado, S. Vauttier, and H. E. Salman, "Feature location in a collection of software product variants using formal concept analysis," in *International Conference on Software Reuse (ICSR 2013)*, Springer, pp. 302–307, 2013.

[26] B. Zhang and M. Becker, "Code-based variability model extraction for software product line improvement," in *16th International Software Product Line Conference (SPLC 2012)*, ACM, pp. 91–98, 2012.

[27] Bosch, Classic Sensor Software Process Introduction. Bosch Global Network (Internal Information). Bosch Car Multimedia Portugal, S.A. Braga. Classic Sensor Software Process Introduction, 2018.

[28] Bosch, J. Seidel, Personal interview. Bosch Global Network (Internal Information). Bosch Car Multimedia Portugal, Braga. Classic Sensor Software Process Introduction, 2018.

[29] Bosch, Sensor Software Process Variant 1. Bosch Global Network (Internal Information). Bosch Car Multimedia Portugal, Braga, 2018.

[30] Bosch, SW Architecture. Bosch Global Network (Internal Information). Bosch Car Multimedia Portugal, Braga, 2018.

[31] Bosch, Creating and handling SW variants. Bosch Global Network (Internal Information). Bosch Car Multimedia Portugal, Braga, 2018.

[32] Bosch, SAS team. Group meeting. Bosch Global Network (Internal Information). Bosch Car Multimedia Portugal, Braga, 2018.

[33] V. Alves, N. Niu, C. Alves, and G. Valença, "Requirements engineering for software product lines: A systematic literature review," *Information and Software Technology* 52(8):806–820, 2010.

[34] K. Czarnecki, U. Eisenecker, and K. Czarnecki. "*Generative programming: methods, tools, and applications*," Reading: Addison Wesley, 2000.

[35] R. F. Al-Msie'Deen, "*Reverse engineering feature models from software variants to build software product lines*," Ph.D. dissertation, University of Montpellier, France, 2014.