American Scientific Research Journal for Engineering, Technology, and Sciences (ASRJETS)

ISSN (Print) 2313-4410, ISSN (Online) 2313-4402

© Global Society of Scientific Research and Researchers

http://asrjetsjournal.org/

Model Defects in Evolving Software Product Lines: A Review of Literature

Amal Khtira [a]*, Anissa Benlarabi [b], Bouchra El Asri [c]

[a,b,c]IMS Team, ADMIR Laboratory, Rabat IT Center, ENSIAS, Mohammed V University, Rabat, Morocco

[a]Email: amalkhtira@gmail.com

[b]Email: a.benlarabi@gmail.com

[c]Email: elasri_b@yahoo.fr

Abstract

Software products lines (SPLs) are long living systems that undergo several evolutions throughout their lifetime due to many reasons related to technology, strategy, business, etc. These evolutions can be the source of several defects that impact the different artefacts of SPLs, namely requirements, models, architecture and code. Many studies in the literature have dealt with the correction of defects in software product lines, but to our knowledge, no reviews have been carried out to provide an extensive overview of these studies. In this paper, we present a literature review of model defects in software product lines. The purpose of this review is to enumerate the different defects discussed in literature and to present the approaches proposed to detect and correct them. The findings of this review reveal new research leads to explore in this issue.

Keywords: Evolution; Literature Review; Model Defects; Software Product Line

1. Introduction

Software Product Line Engineering is a paradigm whose purpose is to reuse a set of core assets to develop specific software products for the benefit of different customers [1]. This paradigm has many advantages, namely the enhancement of software quality, the reduction of production cost, the promotion of reusability and the reduction of Time to Market [2]. Software product lines are long-lived systems that require inevitably permanent evolution. This evolution is mainly caused by new technologies, new customers' needs and new business strategies. Works in literature have dealt with many issues related to SPL Evolution, such as evolution traceability [3,4,5], evolution modeling [6,7,8], co-evolution analysis [9,10,11] and change impact analysis [12,13,14,15].

----

* Corresponding author.

Software product line Evolution presents different challenges and may impact negatively the product quality. McConnell [16] distinguishes two main aspects to take into consideration regarding software quality, a functional aspect that consists of reducing the defect rate in the different artefacts of a software product, and a non-functional aspect that aims at enhancing certain characteristics such as performance, maintainability and robustness. The first aspect occupies an important place in literature. Indeed, a poor defect management may cause indirectly the deterioration of non-functional qualities. In addition, if defects are not detected in an early stage of a project, they propagate throughout the development lifecycle and their correction becomes difficult, costly and time-consuming. Consequently, several studies have dealt with the detection and correction of defects in software products. Specifically, there has been a great deal of research in this area in the context of software product lines.

In this paper, we present a literature review on model defects in software product lines. The objective of this review is to summarize the state of the art in this area between 2010 and 2016 and to identify the quantity and the nature of contributions in the collected works. We believe this study will be a great support for both researchers and practitioners, because it provides an overview of the different defects discussed in literature and discusses the different approaches proposed to handle them. It also highlights the gaps in these approaches and suggests new research trends and some areas for improvement.

The remainder of the paper is organized as follows. Section 2 explains our literature review methodology. In Section 3, we analyze and discuss the results found against the predefined research questions. Section 4 presents some limitations of the study. Finally, Section 5 concludes the paper.

2. Research Methodology

Several solutions have been proposed in relation with model defects in Software product lines. At the aim of collecting and evaluating these solutions, we have carried out a systematic literature review (SLR) by following the protocol described in [17]. This protocol is composed of six main steps: 1) Identification of research questions, 2) Research in Databases, 3) Definition of Inclusion and Exclusion criteria, 4) Data Selection, 5) Data Extraction, and 6) Data Analysis. The first five steps of the followed protocol are detailed in the rest of this section, while the last step is detailed in Section 3.

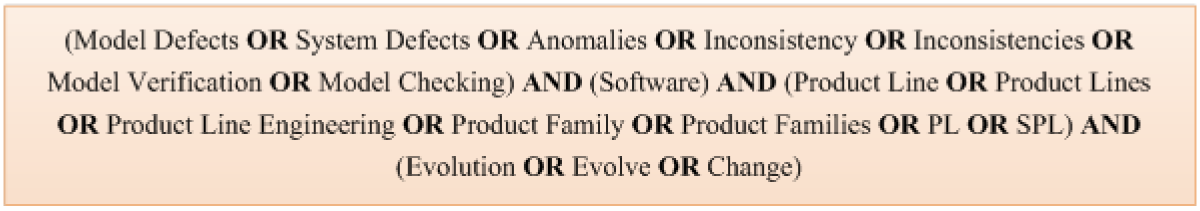
2.1. Search Questions and Search String

The objective of this review is to search and analyze the studies dealing with model defects in evolving software product lines. In order to achieve this objective, we formulated the following research questions:

- **QR1.** What are the different approaches proposed with respect to model defects caused by software product lines evolution?
- **QR2.** What are the types of contributions regarding the detection and correction of model defects in software product lines?
- **QR3.** What are the different artefacts impacted by defects and how are they handled in literature?

- **QR4.** What are the different defects of software product lines addressed in literature?

In order to answer the research questions already defined, we constructed the research string using the keywords related to our topic. The basic keywords are: Software, Product Line, Model Defect and Evolution. To make the research more efficient, we defined a set of synonyms and alternatives for the different keywords. To link the alternative keywords, we used the Boolean “OR” and to interconnect the different parts of the string, we used the Boolean “AND”. As a consequence, we obtained the research string presented in Figure 1.



(Model Defects **OR** System Defects **OR** Anomalies **OR** Inconsistency **OR** Inconsistencies **OR** Model Verification **OR** Model Checking) **AND** (Software) **AND** (Product Line **OR** Product Lines **OR** Product Line Engineering **OR** Product Family **OR** Product Families **OR** PL **OR** SPL) **AND** (Evolution **OR** Evolve **OR** Change)

Figure 1: The search string

2.2. Search in Databases

During this step, we performed a search in the most famous digital libraries using the constructed string. We considered publications retrieved from IEEE Xplore, ACM Digital Library, ScienceDirect, Springer Link and Google Scholar. Since every database has a particular search system, we had to derive many search strings and to carry out additional steps to have equivalent results from the different databases.

- **IEEE Xplore**: In this database, the maximum number of keywords allowed in search strings is fifteen. Since our string contains more than fifteen keywords, we had to divide the base string into several strings then merge the results of the different searches.

- **ACM Digital Library**: The notation used to construct search strings in ACM is different from the other databases. Hence, we had to transform the initial string to respect this notation. The symbol "+" is used instead of the Boolean AND, and the Boolean "OR" is replaced by a white space.

- **ScienceDirect – Elsevier**: This database gives the possibility to refine search through many filters. In our case, we filtered the articles by date, search field and publication title.

- **SpringerLink**: To simplify the search, we needed to overcome two difficulties: i) The filter concerning the search field contains interconnected filters, which makes the search complicated, ii) The maximum number of results that can be extracted is 1000. For this, we applied the same search string for the interconnected fields separately. We also filtered the publications by date to separate results over different periods.

- **Google Scholar**: The search in this database is more complicated in comparison with the other databases for two reasons: i) Google scholar does not give the possibility to write a complete search string. We therefore had to use the basic search tool to conduct a search that is equivalent to the initial string. ii) There is no way in Google Scholar to extract search results. To overcome this gap, we used the open source tool "Outbit Hub" to extract results to Excel.

2.3. Data Selection

As stated before, the objective of a systematic review is to collect relevant approaches proposed regarding a particular area. Hence, we defined a number of inclusion and exclusion criteria for our search to select only relevant articles. A paper is included if it fulfills the following criteria:

- **IC1**: The paper is a full article, a book, a chapter, a technical report, a thesis, a presentation.
- **IC2**: The title or the abstract of the paper contains the keywords related to the search.
- **IC3**: The paper addresses one or more model defect in software product lines.

The paper is excluded if it meets the following criteria:

- **EC1**: The paper is written in a language other than English.
- **EC2**: The publication date is previous to 2010.
- **EC3**: The paper is a short article, a standard, a poster, an editorial, a tutorial.
- **EC4**: The title, the keywords and the abstract do not correspond to the research subject.
- **EC5**: The paper does not deal with model defects in software product lines.
- **EC6**: Another recent paper is published by the same team for the same solution.

The data selection activity was carried out in several stages as described in Figure 2.

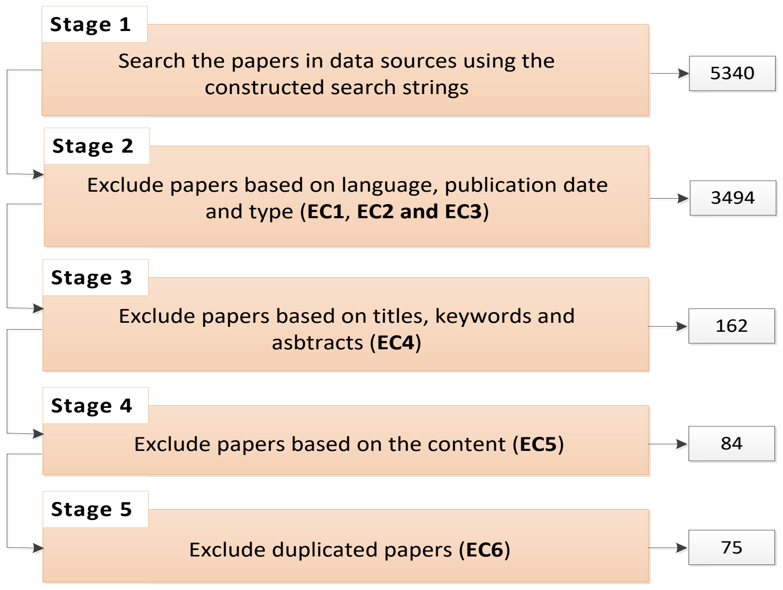

Stage 1
Search the papers in data sources using the constructed search strings
5340
Stage 2
Exclude papers based on language, publication date and type (EC1, EC2 and EC3)
3494
Stage 3
Exclude papers based on titles, keywords and asbtracts (EC4)
162
Stage 4
Exclude papers based on the content (EC5)
84
Stage 5
Exclude duplicated papers (EC6)
75

Figure 2: Data selection process

The total number of papers initially retrieved is 5340, divided as follows: 1201 from IEEE Xplore, 72 from ACM Digital Library, 386 from ScienceDirect, 2685 from SpringerLink and 996 from Google Scholar. In order to ensure the effectiveness of our search, we made a first check on whether our papers published on the subject are retrieved in the performed search or not. At the end of the selection process, we obtained 75 papers.

After this first selection of papers, we carried out another phase of selection based on the quality of the collected studies. Thus, we defined three additional quality assessment criteria: the number of citations, the approach validation and the work continuity.

- **Number of citations**: For each year of publication, we defined the lower number of citations required so that a paper is considered.
- **Approach validation**: A paper is excluded if the approach proposed in it is validated by a case study, a quantitative or a qualitative evaluation, etc.
- **Work continuity**: If the work described in a paper is not detailed or continued in recent papers, the paper is excluded.

After applying the quality assessment criteria on the 75 papers, we finally kept the 48 papers presented later in Table 4.

2.4. Data Extraction

During this step, we designed a personalized form consisting of a number of attributes. Then, the form was filled by researchers for all the selected papers. Table 1 presents the list of these attributes. The objective of this action was to collect and synthesize the data in a way that helps us answer the predefined research questions.

Table 1: Attributes used in data extraction

Title	Title of the paper
Authors	Authors of the paper
Year	Year of publication of the paper
Type of the paper	e. g. Journal paper, conference paper, thesis, book, chapter, workshop paper.
Database	e. g. IEEE, ACM, Springer, Elsevier, Google Scholar
Source	e. g. Journal name, conference name
Research type	e. g. Approach validation, approach evaluation, solution proposal, experiment, case study
Keywords	Keywords specified in the paper
Objective	Objective of the study
Description	Short description of the paper content
Methodology	Methodology followed in the study
Contribution	e. g. Model, Framework, Tool, Method, Algorithm
Model defect	Model defect addressed in the paper
Domain of application	Field in which the study was applied (if it exists)

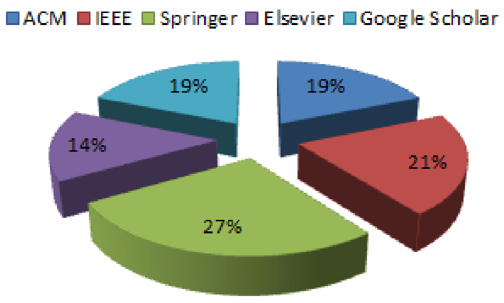
3. Results and Discussions

During this step, the objective was to answer the research questions defined in the beginning of this review. For this, we studied the selected papers from different perspectives. First, we analyzed the metadata of these papers, then we focused on their contents. This section presents and discusses in details the results of this analysis.

3.1. Demographic Data

Demographic data consists of the metadata of the selected papers. In our review, we focused on three types of metadata: the database, the year of publication and the source avenue.

- **Database and year of publication**



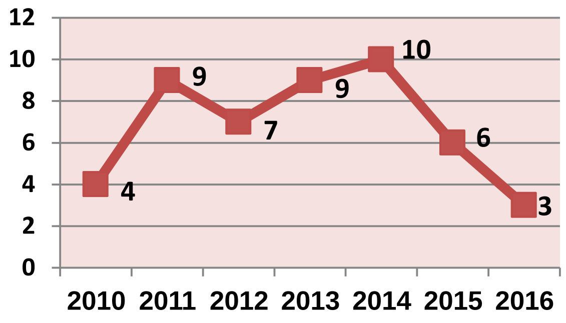


Figure 3: Percentage of papers vs. Database

Figure 4: Number of articles vs. year

Figure 3 presents the percentage of papers in each database. Among the 48 papers retrieved from the five databases used in the systematic review, we identified 7 papers in Elsevier (15%), 9 papers in Google Scholar (19%), 9 papers in ACM (19%), 10 papers in IEEE (21%) and 13 papers in Springer (27%). We notice that Springer is the database that contains the largest number of papers with a percentage of 27%, then IEEE with 21%, which can be explained by the large number of papers retrieved from these two databases (2685 for Springer and 1201 for IEEE). However, this hypothesis may only be rebutted by analyzing the other databases. Indeed, even if ACM contains the fewest papers in the first step with only 1,34 % of papers, the percentage of papers at the end of the review is 19%, which is same as Google Scholar and greater than Elsevier. Similarly, Google Scholar that represents 19% of the initially selected papers represents the same percentage at the end and Elsevier that had more papers than ACM represents at the end the minimum number of papers. Two conclusions are drawn. First, the quality of papers is not proportional to their number. Indeed, a database may contain a large number of papers related to a given area, but most of them can be of poor quality. Second, the search engines used in the digital databases are not perfect, which means that a significant number of papers retrieved in the first stage of the review do not correspond to the search scope.

As mentioned before, the search was carried out for the period 2010-2016. The diagram presented in Figure 4 shows the number of papers per year of publication. We notice that there is a consistent trend regarding the study of model defects in software product lines between 2011 and 2015, with a number of papers that oscillates between 6 and 10. The peak year for model defects publications was 2014 with 10 published papers.

- **Source avenue**

Table 2 presents the different source avenues of the selected papers and their types as well as the number and percent of papers for each source. According to this table, there are 26 source avenues, 12 journals, 9 conferences, 3 workshops and 3 books. The number of papers published in journals is 24, which represents 50% of the papers. The journals that contain the highest number of papers are the Elsevier journals « *Information and Software Technology* » and « *Journal of Systems and Software* » and the Springer journal « *Lectures Notes in Computer Science* ». The number of papers retrieved from conferences is 18 (37% of total papers), with 8 articles published in SPLC (*Software Product Line Conference*). All these findings are logical because the majority of papers are published in important journals and conferences that accept only complete works detailed methods and validated solutions. Regarding the books, they were selected by the review because, besides the fact that they correspond to the search scope, they are also well cited (i. e. 225 citations for the book "*Feature-Oriented Software Product Lines"*).

Table 2: Source avenues

Source avenue	Type	#Papers	%
ACM/IEEE International Conference on Model Driven Engineering Languages and Systems - MODELS	Conference	1	2%
Annual ACM Symposium on Applied Computing	Conference	1	2%
Autonomous Agents and Multi-Agent Systems	Journal	1	2%
Conference on Software Product Line	Conference	8	17%
Euromicro Conference on Software Engineering and Advanced Applications	Conference	1	2%
Expert Systems	Journal	1	2%
Expert Systems with Applications Journal	Journal	1	2%
Feature-Oriented Software Product Lines	Book	1	2%
IEEE Annual Computer Software and Applications Conference	Conference	2	4%
IEEE International Conference on Software Maintenance	Conference	1	2%
IEEE Transactions on Software Engineering	Journal	1	2%
Information and Software Technology	Journal	3	6%
Information Science Journal (ISJ)	Journal	1	2%
International Conference on Distributed Multimedia Systems	Conference	1	2%
International Conference on Software Engineering	Conference	1	2%
International Requirements Engineering Conference	Conference	2	4%
International Workshop on Model-Driven Requirements Engineering - MoDRE	Workshop	1	2%
Journal of Systems and Software	Journal	3	6%
Journal of Universal Computer Science	Journal	2	4%
Lectures Notes in Computer Science	Journal	6	13%
Relating Software Requirements and Architectures	Book	1	2%
Requirements Engineering	Journal	2	4%
Research Journal of Applied Sciences, Engineering and Technology	Journal	1	2%
Software and Systems Modeling	Journal	2	4%
Software Product Line - Advanced Topic	Book	1	2%

Workshop on Emerging Ideas and Trends in Engineering of Cyber-Physical Systems	Workshop	1	2%
Workshop on Variability Modelling of Software-intensive Systems - VaMoS	Workshop	1	2%
Total		**48**	**100%**

3.2. Contributions Analysis

After analyzing the metadata of the selected papers, we focused on the study of the papers contents, the nature of the provided solutions and the proposed contributions. As a result of this study, the first observation we made is that 80% of papers are solution proposals for model defects. The other 20% are shared between case studies, empirical studies and reviews of approaches. In each research type, several contributions are proposed. Table 3 presents the number and the list of papers for the different types of contribution. The contributions can be classified into three categories: Analysis-oriented contributions, Design-oriented contributions and Implementation-oriented contributions.

Table 3: Papers vs. contributions

Category	Type of contribution	#Papers	Papers
Analysis-Oriented Contributions	Questionnaire / Survey / Interviews	3	[37], [57], [60]
	Analysis / Review	2	[20], [60]
	Heuristics	1	[30]
	Checklist	1	[26]
Design-Oriented Contributions	Method	16	[18], [22], [25], [27], [32], [5], [15], [35], [43], [45], [48], [50], [51], [54], [55], [58]
	Process	5	[33], [38], [41], [8], [53]
	Model / Meta-model	6	[15], [8], [49], [34], [21], [42]
	Rules / Formulas	4	[28], [29], [14], [44]
	Algorithm	4	[35], [43], [52], [42]
	Ontology	2	[32], [36]
	Metrics / Measures	3	[10], [53], [30]
	Templates	1	[46]
Implementation-Oriented Contributions	Tool / Tool Extension / Prototype	17	[18], [19], [20], [27], [28], [5], [15], [34], [35], [37], [40], [45], [48], [8], [50], [53], [54], [42]
	Framework	7	[24], [38], [47], [8], [56], [59], [42]
	Language	3	[23], [31], [39]

- **Analysis-Oriented Contributions**

This category corresponds to the papers dealing with the review and analysis of existing studies or the evaluation of new approaches. For example, a quasi-systematic review carried out by [26] revealed that few approaches address software product line inspection. Thus, the paper proposes a checklist-based inspection technique (FMCheck) that assists inspectors in the detection of defects in feature models. Wnuk Farias and his colleagues in [57] present the results of an empirical study based on a survey of 219 participants from different companies. The study focuses on the phenomenon of obsolete requirements and aims at defining this phenomenon from the perspective of industry practitioners. It also analyses the impact of these requirements on projects and discusses how they are handled in industry. Farias and his colleagues in [30] presents an exploratory study whose objective is to assess empirically the impact of stability on the effort of model composition and on the inconsistency rate.

- **Design-Oriented Contributions**

These contributions encompass methods, processes, meta-models, ontologies, etc. Among design studies, [37] introduced the method PUM (Product line Use case modeling Method) for documenting the variability in use case diagrams and specifications and in the associated domain models. Inverardi and Mori [38] propose a model-centric software evolution process of context-aware adaptive systems. In this process, the system behavior is represented using feature diagrams, which supports foreseen and unforeseen evolution. The first type corresponds to the foreseen variations of context, while the second type deals with the new needs of customers that could occur during the execution of unforeseen changes of context.

Within the context of modeling, [49] defined the model PL-CDM that includes the implementations, the features and the dependencies between them. This model can be used by both developers and managers in software product lines to detect inconsistencies between cloned products and synchronize the development and the updates via real-time notifications. In a traceability perspective, [5] aims at enhancing the relations between requirements by assigning them types and providing them semantics expressed in first-order logic formulas. For this, a meta-model for requirements is proposed with formal relation types. In the same vein, [32,36] propose ontologies to specify the concepts related to feature models, which enables to monitor the evolution of these models and facilitates the detection of inconsistencies. Neves Farias and his colleagues in [46] define generic safe evolution templates based on the analysis of the evolution history of different product lines.

- **Implementation-Oriented Contributions**

This category includes tools, prototypes and frameworks proposed regarding the detection and correction of model defects in software product lines. As for tools, Alférez Farias and his colleagues in [19] presents the VCC tool that enables the verification of constraint consistency between feature models and SPL base models through the expression of constraints in the form of propositional formulas. In the same way, [18] proposes the VCC4RE tool that helps verify consistency of semantic relations between features and the associated use scenarios. Kamalrudin Farias and his colleagues in [40] introduced Marama, an automated tracing tool that allows users to capture requirements and generate automatically the corresponding essential use cases (EUC). This tool supports the detection of inconsistencies between textual requirements, abstract interactions and the EUCs. Another tool for the detection of inconsistencies is TRIC proposed by [5] then extended by [15]. At the aim of verifying the conformance between variability in the design level and variability in the requirement level, [45] introduced the SPLEnD tool. The FDDetector tool introduced by [42] enables the detection and correction of duplicated features during the evolution of feature-oriented software product lines.

In addition to tools, several frameworks have been proposed. Indeed, [8] presented SPLEMMA, a generic evolution framework that enables the validation of controlled SPL evolution by following a Model Driven Engineering approach. This study focused on three main challenges: SPL consistency during evolution, the impact on the family of products and SPL heterogeneity. Vierhauser Farias and his colleagues in [56] propose a generic framework for the verification of consistency between heterogeneous artefacts in a product line. The framework was integrated in the DOPLER product line tool suite. Khtira Farias and his colleagues in [42] present a conceptual framework for feature deduplication whose main processes consists of unifying the input artefacts of a derived product and detecting the internal and external duplications using an approach based on natural language processing.

3.3. Artefacts Discussed

During software product line evolution, different artefacts may be impacted. The artefacts that we identified in the studied papers are: requirements, domain models, domain architecture and domain code. Figure 5 shows the aggregation of papers per artefact.

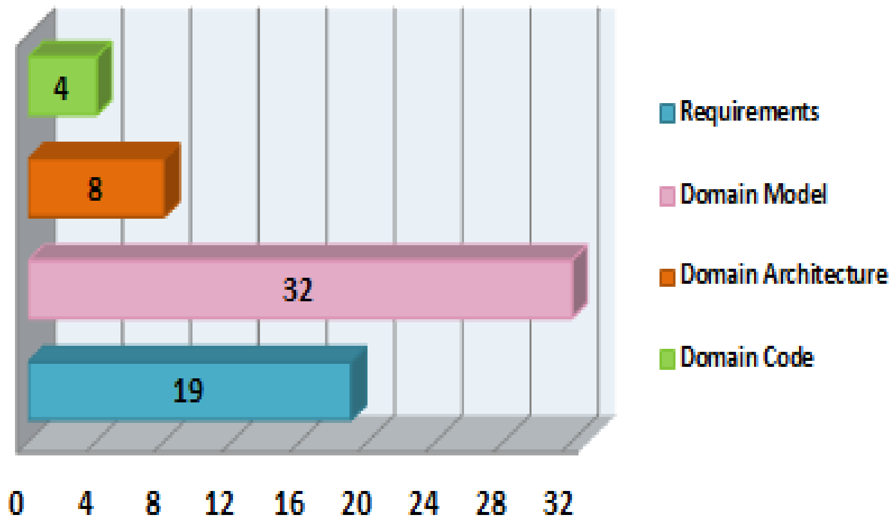


Figure 5 : The number of papers by artefact

Table 4 presents a comparison between papers depending on the artefact they handle and specifies the approach followed to model the problem addressed or the proposed solution.

Table 4: Comparison between approaches according to artefacts

		Requirements	Domain Model	Domain Architecture	Domain Code
[Alférez Farias and his colleagues in , 2011]	[18]	UML (UC, AD)	FM	X	X
[Alférez Farias and his colleagues in , 2014]	[19]	√	FM	X	X
[Apel Farias and his colleagues in , 2013]	[20]	X	FM	X	X
[Bessling and Huhn, 2014]	[21]	√	FM	X	X
[Boutkova and Houdek, 2011]	[22]	NL	FM	X	X
[Cordy Farias and his colleagues in , 2013]	[23]	X	FM	X	X
[Dam and Winikoff, 2011]	[24]	X	X	UML (CD)	X
[Dam Farias and his colleagues in , 2016]	[25]	X	X	UML (SD, CD, AD)	X
[de Mello Farias and his colleagues in , 2014]	[26]	X	FM	X	X
[Dhungana Farias and his colleagues in , 2010]	[27]	X	VM	X	X
[Egyed, 2011]	[28]	X	X	UML (SD, CD, AD)	X
[Elfaki Farias and his colleagues in , 2014]	[29]	X	FM, FOL	X	X
[Elfaki, 2016]	[14]	X	FM, FOL	X	X
[Farias Farias and his colleagues in , 2014]	[30]	X	X	UML (CD, CMD)	X
[Ferreira and Silva, 2012]	[31]	NL	X	X	X
[Filho Farias and his colleagues in , 2012]	[32]	X	FM	X	X
[Gamez and Fuentes, 2013]	[33]	X	FM	X	X
[Goknil Farias and his colleagues in , 2011]	[5]	√	X	X	X
[Goknil Farias and his colleagues in , 2014]	[15]	√	X	X	X
[Greenyer Farias and his colleagues in , 2013]	[34]	√	FM	UML (SD)	X
[Groher Farias and his colleagues in , 2010]	[35]	X	X	UML (SD, CD)	X
[Guo Farias and his colleagues in , 2012]	[36]	X	FM	X	X

[Hajri Farias and his colleagues in , 2015]	[37]	NL, UML (UC, US)	X	X	X
[Hellebrand Farias and his colleagues in , 2014]	[10]	X	FM	X	√
[Inverardi and Mori, 2013]	[38]	√	FM	X	X
[Jureta Farias and his colleagues in , 2010]	[39]	√	X	X	X
[Kamalrudin Farias and his colleagues in , 2010]	[40]	NL, UML (UC)	X	X	X
[Käßmeyer Farias and his colleagues in , 2015]	[41]	X	√	X	X
[Khtira Farias and his colleagues in , 2015]	[42]	NL	FM	X	X
[Manz Farias and his colleagues in , 2014]	[43]	X	√	X	X
[Mazo Farias and his colleagues in , 2011]	[44]	X	FM	X	X
[Millo Farias and his colleagues in , 2013]	[45]	√	FM	√	X
[Neves Farias and his colleagues in , 2015]	[46]	X	FM	X	X
[Noorian Farias and his colleagues in , 2011]	[47]	X	FM	X	X
[Quinton Farias and his colleagues in , 2014]	[48]	X	FM	X	X
[Romero Farias and his colleagues in , 2013]	[8]	√	FM	X	X
[Rubin Farias and his colleagues in , 2012]	[49]	X	FM	X	CC
[Rumpe Farias and his colleagues in , 2015]	[50]	X	X	SM	X
[Salikiryaki Farias and his colleagues in , 2015]	[51]	UML (UC, SD)	√	X	X
[Salinesi and Mazo, 2012]	[52]	X	FM	X	X
[Schmorleiz and Lämmel, 2016]	[53]	X	X	X	CC
[Siedl Farias and his colleagues in , 2012]	[54]	X	FM	X	X
[Stephenson Farias and his colleagues in , 2011]	[55]	√	FM	X	X
[Vierhauser Farias and his colleagues in , 2012]	[56]	X	VM	X	X
[Wnuk Farias and his colleagues in , 2013]	[57]	√	X	X	X
[Yang Farias and his colleagues in , 2011]	[58]	NL	X	X	X
[Zhang and Becker, 2013]	[59]	X	FM, PC	X	X

[Zhang Farias and his colleagues in , 2012]	[60]	X	X	X	CC

NL: Natural Language	UC: Use Cases	FM: Feature Model
VM: Variability Model	PC: Product Configurations	SD: Sequence Diagram
CD: Class Diagram	CMD: Component Diagram	AD: Activity Diagram
SD: State Diagram	FOL: First Order Logic	SM: Simulink Model
CC: Code Cloning	√: Handled	X: Not handled

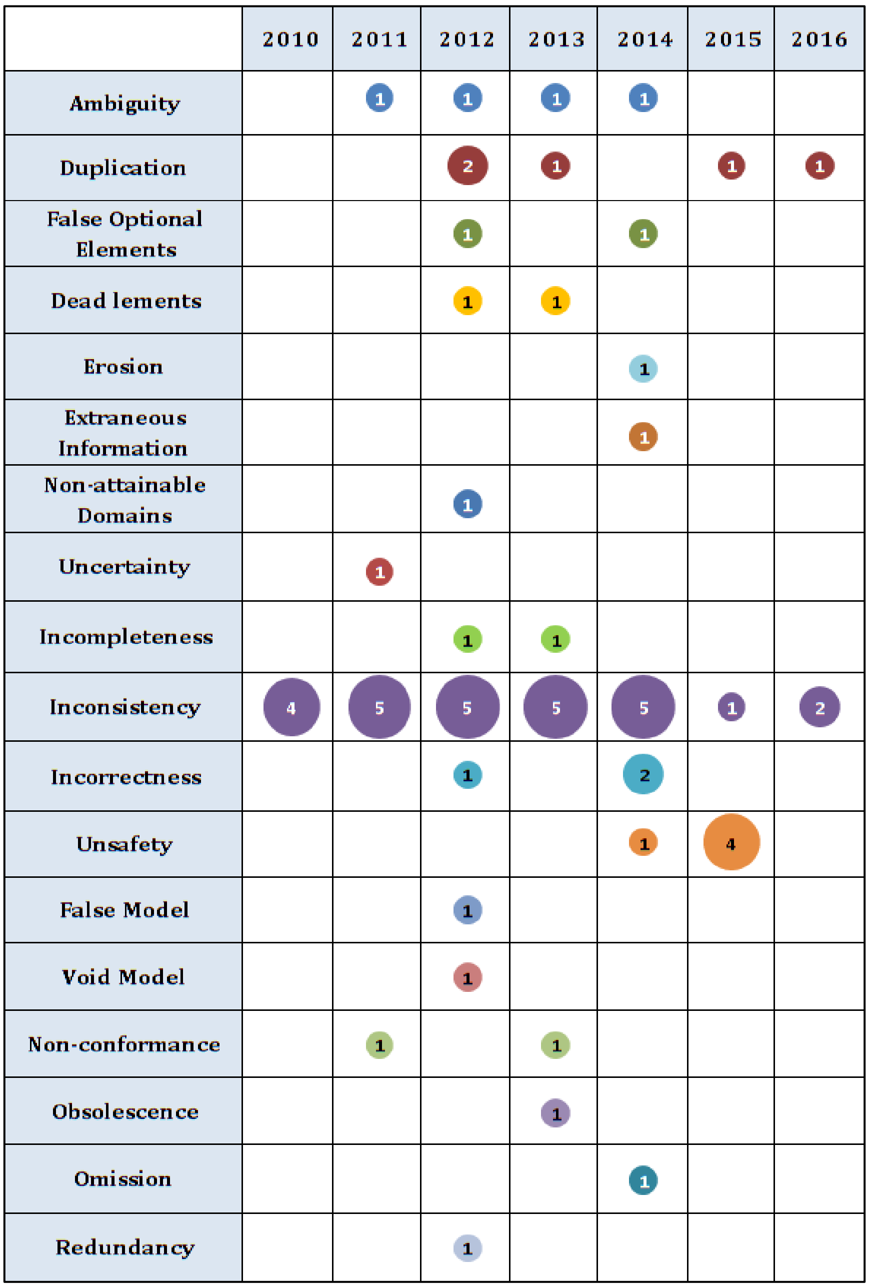
- **Requirement Verification:** In the works dealing with requirements verification, requirements are sometimes expressed as use cases [21,5,38,39,45] or in natural language specifications [42,22,31,37,40,58]. However, some papers do not specify how requirements are expressed [21,5,38,39,45]. Among solutions based on natural language processing, [22] proposes an approach to optimize the identification of features in natural language specifications, based on lexical analysis. The paper describes the results of applying manual and semi-automatic identification of features on the automotive industry. In order to improve the quality and rigor of natural language specifications, Ferreira and Silva [31] present an approach of information extraction based on linguistic patterns. Hajri Farias and his colleagues in [37] propose an NLP-based tool for the verification of use cases and the associated models, whose variability is defined with the method PUM (Product line Use case modeling Method). Yang Farias and his colleagues in [58] introduced an automated system that enables the identification of potential problematic ambiguities among anaphoric ambiguities existing in requirements documents. In addition, many tools of requirements verification have been proposed in literature such as RSLingo [31], TRIC [5] and Marama [40].

- **Domain Model Verification:** Among the 32 papers addressing domain models, 27 focus on feature models such as [42,19,20,22,14,32,46], which is logical because most product lines in literature and industry are feature-oriented. Several solutions have been proposed in this vein, namely tools like VML4RE [18], VCC [19,20], FDDetector [42] and SPLEnD [45], extensions of the DOPLER tool [27,56], frameworks such as SPLEMMA [8], models like PL-CDM [49] or techniques like FMCheck [26].

- **Architecture Verification:** There are 8 papers dealing with architecture among which 6 papers deal with UML design models, namely class diagrams [24,25,28,30,35], sequence diagrams [25,28,34,35], component diagrams [30] and activity diagrams [25,28]. The two other papers address different types of models such as finite state machines [45] and the Simulink models [50].

- **Code Verification:** The number of papers handling code verification is 4. Among these papers, three focus on code cloning [49,60,53] and the fourth deals with co-evolution between feature models and code [10]. Rubin Farias and his colleagues in [49] focus on the management of software product variants realized via cloning. For this purpose, the authors present a framework that consists of seven conceptual operators and validate their efficiency through three case studies from the automotive industry. Schmorleiz and Lämmel [53] describe a process for similarity management of cloned variants during software evolution. This process uses annotations to record developers' intentions and to anticipate automatic change propagation.

At the aim of analyzing and understanding the motivations behind code cloning, [60] presents the results of an industrial study that focuses on the reasons of cloning practices from technical, personal, and organizational perspectives. Hellebrand Farias and his colleagues in [10] address the coevolution between feature models and code. More specifically, it proposes metrics that allow the detection of variability erosion between the two artefacts during SPL evolution. While [49] considers code cloning as a defect that may reduce the product quality and proposes a solution to correct inconsistencies it can generate, we notice that [60][53] see it as a normal phenomenon or even desirable in certain cases, it must only be well managed to benefit from it as much as possible.



	2010	2011	2012	2013	2014	2015	2016
Ambiguity		1	1	1	1		
Duplication			2	1		1	1
False Optional Elements			1		1		
Dead lements			1	1			
Erosion					1		
Extraneous Information					1		
Non-attainable Domains			1				
Uncertainty		1					
Incompleteness			1	1			
Inconsistency	4	5	5	5	5	1	2
Incorrectness			1		2		
Unsafety					1	4	
False Model			1				
Void Model			1				
Non-conformance		1		1			
Obsolescence				1			
Omission					1		
Redundancy			1				

Figure 6: Number of papers for each defect per year

3.4. Model Defects Addressed

Figure 6 illustrates the number of articles dealing with model defects over the studied period. Based on this figure, it is clear that the defect most addressed in literature is inconsistency, either in terms of number or continuity. Indeed, in all years, there is at least a paper about inconsistency (with 4 papers in 2010 and 5 papers in the years between 2011 and 2014). In total, 27 of the 48 selected papers deal with inconsistency. The second defect constantly addressed is ambiguity, with one paper per year between 2011 and 2014. In addition, we notice that unsafety starts getting the attention of researchers since 2014 (the number of papers dealing with this defect increases from one article in 2014 to 4 articles in 2015). Incorrectness was addressed 3 times (one paper in 2012 and 2 papers in 2014). The defects handled two times are dead elements, false-optional elements, incompleteness and non-conformance. The rest of defects, namely erosion, inaccessibility, extraneous information, uncertainty, false models, void models, obsolescence, omission and redundancy do not receive great attention in literature (one paper for each defect since 2010). A large number of these defects are defined by [52] and [26]. As for duplication, it is not constantly addressed, some papers dealt with it in 2012 and 2013 to reappear later in 2015.

After reading the selected papers of the systematic review, we first extracted the different definitions of model defects addressed in these papers. Then, a unique definition for each defect was chosen or formulated as presented in Table 5.

In general, when a model defect is handled in a paper, we start by giving it a definition, either generic or specific to the context of the approach in question. But in certain cases, we simply quote a definition proposed in another work. That’s why we find in Table 5 that some definitions are retrieved directly from the papers concerned by the systematic review, while others are referenced in other papers.

As seen before, the model defect most addressed in literature is inconsistency. Hence, contrarily to other defects to which we gave a single definition, we have formulated different definitions for inconsistency according to the element considered inconsistent. These elements are: requirements, models, cardinalities, dependencies and implementations. Another observation is that inconsistency may be sometimes confused with other defects such as incorrectness [35], non-conformance [30] or duplication [40].

4. Limitations of the Review

In this review, we tried our best to select pertinent studies regarding model defects in software product lines. However, the results of the review could have been affected by some limitations. First, the search systems of the different databases are not accurate, because a great number of papers found by the search do not correspond to the review scope. Therefore, we had to apply many filters to exclude the irrelevant papers. In addition, every database uses a different search system. So we were obliged to construct different search strings and perform additional actions to have equivalent results at the end, but there is always a risk that these strings do not generate the intended results. Moreover, search in databases is generally based on keywords existing in the title or the abstract. If a paper dealing with model defects in software product lines but do not use the same keywords defined in the search string, they could have been missed by the search.

Table 5: Definitions of Model Defects

Model Defect	Definition		Source
Ambiguity	Some Information from the feature model is not clear, allowing multiple interpretations for the specified domain.		[26]
Duplication	To have the same thing expressed in two or more places; duplication can happen in specifications, processes and programs.		[61]
False Optional Elements	A reusable element is false optional if it is included in all the products of the product line despite being declared optional.		[52]
Dead Elements	A reusable element is dead if it cannot appear in any product of the product line.		[52]
Erosion	Erosion means that realization artifacts become overly complex due to unforeseeable changes.		[Zhang Farias and his colleagues in , 2013]
Non-attainable domains	A non-attainable value of a domain is the value of an element that never appears in any product of the product line.		[52]
Uncertainty	Requirements uncertainty refers to changes that occur to requirements during the development of software.		[55]
Incompleteness	The lack of necessary information related to a feature or requirement.		[62]
Inconsistency	Requirement	An inconsistency is a situation where the co-existence of certain relations among requirements causes a contradiction in a specific context.	[5]
	Model	Some feature model element is not consistent with another element from the same feature model.	[26]
	Cardinality	A feature cardinality is considered as range inconsistent if no product exists for some values of its range.	[48]
	Dependency	A dependency is inconsistent when what is being provided by B does not match what entity A needs.	[24]
	Implementation	An implementation of a feature is inconsistent when products contain different sets of functionalities for a given feature.	[49]
Incorrectness	Some information or behavior from the feature model contradicts its domain specification.		[26]
Extraneous information	Some Information in the feature model is outside the domain scope.		[26]
Unsafety	This happens when the behavior of existing products is affected by a new evolution.		[63]
False Model	At most one valid product can be configured with it.		[52]
Void Model	The Product Line Model (PLM) does not define any products.		[52]
Non-conformance	Given a feature f, and a (FSMd, FSMr) pair corresponding to f, we say that the design of f conforms to the requirements of f, if every variant of the FSMd has a corresponding FSMr variant.		[45]

Obsolescence	An obsolete software requirement is a software requirement, implemented or not, that is no longer required for the current release or future releases.	[57]
Omission	Some information from the domain was not properly included in the feature model.	[26]
Redundancy	Redundancy in a PLM is about the presence of reusable elements and variability constraints among them that can be omitted from the PLM without loss of semantic on the PLM.	[52]

In the data selection process, we have defined many exclusion criteria and quality assessment criteria in order to select high quality papers. By applying these criteria, some interesting approaches might have been ignored because they are not well cited, because they are published in short papers, or because the corresponding papers are written in a language other than English, etc.

In our review, the results were retrieved from five databases including Google Scholar. Hence, papers published in other digital libraries like Citeseer or Web of Science or in some specific journals must have been found if they are indexed in Google Scholar, but if this is not the case, they did certainly not appear in the results of the search. Our literature review may also have missed some industrial solutions that have not been published in literature.

5. Conclusions

In the world of today, customers are increasingly demanding, technology advances exponentially and new business strategies emerge. In order to keep up with these changes, all software products and specifically software product lines must evolve constantly, which makes them more and more complex. An inevitable result is the appearance of defects in the different artefacts of the software products lines. The verification process is thus necessary to ensure a satisfactory level of product quality. To make progress regarding the verification of model defects in software product lines, both practitioners and researchers need to have an overview on the work done in this area, the approaches proposed and the gaps to be tackled. The purpose of this paper was to carry out a systematic review that focuses on model defects in evolving software product lines.

As a result of searching studies from 2010 and 2016 in five digital libraries, we identified at the beginning 5340 papers. Based on a set of exclusion criteria and quality assessment criteria, 48 relevant papers were selected. The main objectives of this review was to investigate the different approaches proposed with respect to model defects in SPLs, to identify the nature of contributions in this area, to determine the different artefacts concerned by this issue and to enumerate all the model defects addressed in literature.

Several findings have been drawn from this review. First, we noticed that contributions cover all the phases of a product, from analysis to implementation, but the focus was mainly on design and modeling. We believe that frameworks and tools must be given more attention because without proper tools, it is difficult to validate the proposed approaches. Concerning the artefacts handled by the selected papers, a large number of studies focus on the verification of feature models, which is normal in the case of software product lines, but it would be good to propose generic solutions that can be applied in all kinds of models. Similarly, the approaches proposed regarding the verification of architecture deal mainly with UML design models. Thus, more work remains to be done to cover other types of design models. As for code, little attention has been given to this artefact in research and the defects that have been discussed in relation with it are limited (all the papers dealing with code focus on code cloning). Research should be encouraged in this way especially in the projects that concentrate on implementation where the code is the central artefact.

An analysis of the defects addressed in the papers selected by the systematic review has shown that the most discussed model defect is inconsistency (27 out of 48 papers), while other defects are not thoroughly treated, such as erosion, inaccessibility, extraneous information, uncertainty, obsolescence and duplication. Researchers can therefore propose solutions to verify and correct these defects in the different artefacts of a software product line.

During our review, many works have been eliminated because they do not validate their contributions using an industrial SPL, even if the solutions they propose can be promising. To overcome this problem, researchers and practitioners need to collaborate seriously to work on industrial problems and develop new tools and solutions, which would be of interest for both stakeholders. Moreover, the scope of this review was limited to software product lines, but a more general review of literature could help bring ideas and solutions from other fields and apply them to SPLs to verify model defects.

References

[1] L. M. Northrop, “SEI’s Software Product line Tenets.” *IEEE Software*, vol. 19, no. 4, pp. 32-40, 2002.

[2] K. Pohl, G. Böckle, and F. Van Der Linden, *Software Product Line Engineering Foundations, Principles, and Techniques*, Berlin, Germany: Springer-Verlag, 2005.

[3] D. Yu, P. Geng, and W. Wu, “Constructing traceability between features and requirements for software product line engineering,” in *Proc. 19th Asia-Pacific Software Engineering Conference (APSEC),* IEEE, 2012, pp. 27-34.

[4] N. Anquetil, U. Kulesza, R. Mitschke, Farias and his colleagues in , “A model-driven traceability framework for software product lines.” *Software and Systems Modeling*, vol. 9, no. 4, pp. 427-451, 2010.

[5] A. Goknil, I. Kurtev, K. van den Berg, and J. Veldhuis, “Semantics of trace relations in requirements models for consistency checking and inferencing, Software Systems Modeling.” *Springer*, vol. 10, no. 1, pp. 31-54, 2011.

[6] S. Lity, S. Nahrendorf, T. Thüm, Farias and his colleagues in , “175% Modeling for Product-Line Evolution of Domain Artifacts,” in *Proc. International Workshop on Variability Modelling of Software-Intensive Systems*, ACM, 2018, pp. 27-34.

[7] A. Pleuss, G. Botterweck, D. Dhungana, Farias and his colleagues in , “Model-driven support for product line evolution on feature level.” *Journal of Systems and Software*, vol. 85, no. 10, pp. 2261-2274, 2012.

[8] D. Romero, S. Urli, C. Quinton, Farias and his colleagues in , “SPLEMMA: A generic framework for controlled-evolution of software product lines,” in *Proc. 17th International Software Product Line Conference,* 2013, pp. 59-66.

[9] L. Passos, K. Czarnecki, S. Apel, Farias and his colleagues in , “Feature-oriented software evolution,” in *Proc. 7th International Workshop on Variability Modelling of Software-intensive Systems,* ACM, 2013, p. 17.

[10] R. Hellebrand, A. Silva, M. Becker, Farias and his colleagues in , "Coevolution of variability models and code: an industrial case study," in *Proc. 18th International Software Product Line Conference*, ACM, Sept. 2014, vol. 1, pp. 274-283.

[11] A. Benlarabi, A. Khtira, and B. El Asri, "CASPL: A Coevolution Analysis Platform for Software Product Lines," in *Handbook of Research on Investigations in Artificial Life Research and Development*, IGI Global, pp. 380-396, 2018.

[12] M. Bhushan, S. Goel, and K. Kaur, "Analyzing inconsistencies in software product lines using an ontological rule-based approach." *Journal of Systems and Software*, 2017.

[13] K. Shumaiev and M. Bhat, "Automatic Uncertainty Detection in Software Architecture Documentation," in *Proc. International Conference on Software Architecture Workshops (ICSAW)*, April 2017, pp. 216-219.

[14] A. O. Elfaki, "A rule-based approach to detect and prevent inconsistency in the domain engineering process." *Expert Systems*, vol. 33, no. 1, pp. 3-13, 2016.

[15] A. Goknil, I. Kurtev, K. Van Den Berg, Farias and his colleagues in , "Change impact analysis for requirements: A metamodeling approach." *Information and Software Technology*, vol. 56, no. 8, pp. 950-972, 2014.

[16] S. McConnell, "Software quality at top speed." *Software Development*, vol. 4, no. 8, pp. 38-42, 1996.

[17] S. B. Kitchenham, *Charters, Guidelines for Performing Systematic Literature Reviews in Software Engineering*, Keele University, UK EBSE-2007-1, 2007.

[18] M. Alférez, R. E. Lopez-Herrejon, A. Moreira, Farias and his colleagues in , "Supporting consistency checking between features and software product line use scenarios," in: Schmid K. (eds) Top Productivity through Software Reuse (ICSR 2011), *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, vol. 6727, pp. 20-35, June 2011.

[19] M. Alférez, R. E. Lopez-Herrejon, A. Moreira, Farias and his colleagues in , "Consistency Checking in Early Software Product Line Specifications-The VCC Approach." *Journal of Universal Computer Science*, vol. 20, no. 5, pp. 640-665, 2014.

[20] S. Apel, D. Batory, C. Kästner, and G. Saake, "Analysis of Software Product Lines," in *Feature-Oriented Software Product Lines*, Berlin: Springer, 2013, pp. 243-282.

[21] S. Bessling and M. Huhn, "Towards formal safety analysis in feature-oriented product line development," in: Gibbons J., MacCaull W. (eds) Foundations of Health Information Engineering and Systems (FHIES 2013), *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, vol. 8315, pp. 217-235, Aug. 2013.

[22] E. Boutkova and F. Houdek, "Semi-automatic identification of features in requirement specifications," in *Proc. 19th International Requirements Engineering Conference (RE)*, IEEE, Aug. 2011, pp. 313-318.

[23] M. Cordy, P. Y. Schobbens, P. Heymans, and A. Legay, "Beyond boolean product-line model checking: dealing with feature attributes and multi-features," in *Proc. International Conference on Software Engineering, IEEE*, May 2013, pp. 472-481.

[24] H. K. Dam and M. Winikoff, "An agent-oriented approach to change propagation in software maintenance." *Autonomous Agents and Multi-Agent Systems*, vol. 23, no. 3, pp. 384-452, 2011.

[25] H. K. Dam, A. Egyed, M. Winikoff, Farias and his colleagues in , "Consistent merging of model versions." *Journal of Systems and Software*, vol. 112, pp. 137-155, 2016.

[26] R. M. de Mello, E. Nogueira, M. Schots, Farias and his colleagues in , "Verification of Software Product Line Artefacts: A Checklist to Support Feature Model Inspections." *Journal of Universal Computer Science*, vol. 20, no. 5, pp. 720-745, 2014.

[27] D. Dhungana, P. Grünbacher, R. Rabiser, and T. Neumayer, "Structuring the modeling space and supporting evolution in software product line engineering." *Journal of Systems and Software*, vol. 83, no. 7, pp. 1108-1122, 2010.

[28] A. Egyed, "Automatically detecting and tracking inconsistencies in software design models." *IEEE Transactions on Software Engineering*, vol. 37, no. 2, pp. 188-204, 2011.

[29] A. Elfaki, S. Fong, P. Vijayaprasad, Farias and his colleagues in , "Using rule-based method for detecting anomalies in software product line." *Research Journal of Applied Sciences, Engineering and Technology*, vol. 7, no. 2, pp. 275-81, 2014.

[30] K. Farias, A. Garcia, and C. Lucena, "Effects of stability on model composition effort: an exploratory study." *Software & Systems Modeling*, vol. 13, no. 4, pp. 1473-1494, 2014.

[31] D. A. Ferreira and A. R. da Silva, "RSLingo: An information extraction approach toward formal requirements specifications," in *Proc. Model-Driven Requirements Engineering Workshop (MoDRE)*, IEEE, Sept. 2012, pp. 39-48.

[32] J. B. F. Filho, O. Barais, B. Baudry, Farias and his colleagues in , "An approach for semantic enrichment of software product lines," in *Proc. 16th International Software Product Line Conference-Volume 2,* ACM, Sept. 2012, pp. 188-195.

[33] N. Gamez and L. Fuentes, "Architectural evolution of FamiWare using cardinality-based feature models." *Information and Software Technology*, vol. 55, no. 3, pp. 563-580, 2013.

[34] J. Greenyer, A. M. Sharifloo, M. Cordy, and P. Heymans, "Features meet scenarios: modeling and consistency-checking scenario-based product line specifications." *Requirements Engineering*, vol. 18, no. 2, pp. 175-198, 2013.

[35] I. Groher, A. Reder, and A. Egyed, "Incremental consistency checking of dynamic constraints," in: Rosenblum D.S., Taentzer G. (eds) Fundamental Approaches to Software Engineering (FASE 2010), *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, vol. 6013, pp. 203-217, 2010.

[36] J. Guo, Y. Wang, P. Trinidad, Farias and his colleagues in , "Consistency maintenance for evolving feature models." *Expert Systems with Applications*, vol. 39, no. 5, pp. 4987-4998, 2012.

[37] I. Hajri, A. Goknil, L. C. Briand, and T. Stephany, "Applying product line use case modeling in an industrial automotive embedded system: Lessons learned and a refined approach," in *Proc. 18th International Conference on Model Driven Engineering Languages and Systems (MODELS)*, IEEE, Sept. 2015, pp. 338-347.

[38] P. Inverardi and M. Mori, "A software lifecycle process to support consistent evolutions," in: Software Engineering for Self-Adaptive Systems II, *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, vol. 7475, pp. 239-264, 2013.

[39] I. J. Jureta, A. Borgida, N. A. Ernst, and J. Mylopoulos, "Techne: Towards a new generation of requirements modeling languages with goals, preferences, and inconsistency handling," in *Proc. 18th IEEE International Requirements Engineering Conference (RE)*, IEEE, Sept. 2010, pp. 115-124.

[40] M. Kamalrudin, J. Grundy, and J. Hosking, "Managing consistency between textual requirements, abstract interactions and Essential Use Cases," in *Proc. IEEE 34th Annual Computer Software and Applications Conference (COMPSAC)*, IEEE, July 2010, pp. 327-336.

[41] M. Käßmeyer, M. Schulze, and M. Schurius, "A process to support a systematic change impact analysis of variability and safety in automotive functions," in *Proc. 19th International Conference on Software Product Line*, ACM, July 2015, pp. 235-244.

[42] A. Khtira, A. Benlarabi, and B. El Asri, "Duplication Detection when evolving Feature Models of Software Product Lines." *Information Science Journal (ISJ)*, vol. 6, no. 4, pp. 592-612, Oct. 2015.

[43] C. Manz, M. Schulze, M. Reichert, "An approach to detect the origin and distribution of software defects in an evolving cyber-physical system," in *Workshop on Emerging Ideas and Trends in Engineering of Cyber-Physical Systems*, Apr. 2014.

[44] R. Mazo, R. E. Lopez-Herrejon, C. Salinesi, Farias and his colleagues in , "Conformance checking with constraint logic programming: The case of feature models," in *Proc. IEEE 35th Annual Computer Software and Applications Conference (COMPSAC)*, IEEE, July 2011, pp. 456-465.

[45] J. V. Millo, S. Ramesh, S. N. Krishna, G. K. Narwane, "Compositional verification of software product lines," in: Johnsen E.B., Petre L. (eds) Integrated Formal Methods (IFM 2013), *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, vol. 7940, pp 109-123, June 2013.

[46] L. Neves, P. Borba, V. Alves, Farias and his colleagues in , "Safe evolution templates for software product lines." *Journal of Systems and Software*, vol. 106, pp. 42-58, 2015.

[47] M. Noorian, A. Ensan, E. Bagheri, Farias and his colleagues in , "Feature Model Debugging based on Description Logic Reasoning," in *Proc. 17th International Conference on Distributed Multimedia Systems*, Aug. 2011, vol. 11, pp. 158-164.

[48] C. Quinton, A. Pleuss, D. L. Berre, Farias and his colleagues in , "Consistency checking for the evolution of cardinality-based feature models," in *Proc. 18th International Software Product Line Conference-Volume 1*, ACM, Sept. 2014, pp. 122-131.

[49] J. Rubin, A. Kirshin, G. Botterweck, and M. Chechik, "Managing forked product variants," in *Proc. 16th International Software Product Line Conference-Volume 1*, ACM, Sept. 2012, pp. 156-160.

[50] B. Rumpe, C. Schulze, M. Von Wenckstern, Farias and his colleagues in , "Behavioral compatibility of simulink models for product line maintenance and evolution," in *Proc. 19th International Conference on Software Product Line*, ACM, July 2015, pp. 141-150.

[51] A. Salikiryaki, I. Petrova, and S. Baumgart, "Graphical approach for modeling of safety and variability in product lines," in *Proc. 41st Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, IEEE, Aug. 2015, pp. 410-417.

[52] C. Salinesi and R. Mazo, *Defects in Product Line Models and how to Identify them*, InTech editions, 2012, p. 50.

[53] T. Schmorleiz and R. Lämmel, "Similarity management of 'cloned and owned' variants," in *Proc. 31st Annual ACM Symposium on Applied Computing*, ACM, Apr. 2016, pp. 1466-1471.

[54] C. Seidl, F. Heidenreich, and U. Aßmann, "Co-evolution of models and feature mapping in software product lines," in *Proc. 16th International Software Product Line Conference-Volume 1*, ACM, Sept. 2012, pp. 76-85.

[55] Z. Stephenson, K. Attwood, and J. McDermid, "Product-Line Models to Address Requirements Uncertainty, Volatility and Risk," in *Relating Software Requirements and Architectures*, Springer Berlin Heidelberg, pp. 111-131, 2011.

[56] M. Vierhauser, P. Grünbacher, W. Heider, Farias and his colleagues in , "Applying a consistency checking framework for heterogeneous models and artifacts in industrial product lines," in: France R.B., Kazmeier J., Breu R., Atkinson C. (eds) Model Driven Engineering Languages and Systems (MODELS 2012), *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, vol. 7590, pp. 531-545, 2012.

[57] K. Wnuk, T. Gorschek, and S. Zahda, "Obsolete software requirements." *Information and Software Technology*, vol. 55, no. 6, pp. 921-940, 2013.

[58] H. Yang, A. De Roeck, V. Gervasi, Farias and his colleagues in , "Analysing anaphoric ambiguity in natural language requirements." *Requirements engineering*, vol. 16, no. 3, p. 163, 2011.

[59] B. Zhang and M. Becker, "Mining complex feature correlations from software product line configurations," in *Proc. 7th International Workshop on Variability Modelling of Software-intensive Systems*, ACM, Jan. 2013, p. 19.

[60] G. Zhang, X. Peng, Z. Xing, and W. Zhao, "Cloning practices: Why developers clone and what can be changed," in *Proc. 28th IEEE International Conference on Software Maintenance (ICSM)*, IEEE, Sept. 2012, pp. 285-294.

[61] A. Hunt and D. Thomas, *The pragmatic programmer: from journeyman to master*, Addison-Wesley Professional, 2000.

[62] G. Lami, S. Gnesi, F. Fabbrini, M. Fusani, G. Trentanni, "An automatic tool for the analysis of natural language requirements." *Informe técnico*, CNR Information Science and Technology Institute, Pisa, Italia, Sept. 2004.

[63] L. Neves, L. Teixeira, D. Sena, Farias and his colleagues in , "Investigating the safe evolution of software product lines." *ACM SIGPLAN Notices,* vol. 47, no. 3, pp. 33-42, 2012.