Initial Evidence for Understanding the relationship between Product Line Architecture and Software Architecture Recovery

Crescencio Rodrigues Lima Neto*, Mateus Passos Soares Cardoso,
Christina von Flach Garcia Chavez, Eduardo Santana de Almeida
Computer Science Department - Federal University of Bahia (DCC/UFBA)
∗ Federal Institute for Education, Science, and Technology of Bahia (IFBA)
{crescencio, mateuspsc, flach, esa}@dcc.ufba.br

Abstract—Context: **Over the years, the interest on software architecture recovery has increased. Due to software product line inherent complexity, the recovery of product line architecture is crucial to alleviate difficulties and enable benefits during the SPL development.** ***Objective:*** **In order to gather data and evidence about the relationship between product line architecture and software architecture recovery,** ***(Method:)*** **we performed a literature survey using some steps of systematic literature review method and an exploratory study.** ***Results:*** **We identified the appearance of more elaborated studies over the years and the majority of solution proposals are used to recover the architecture from legacy systems source code to provide the SPL reference architecture.** ***Conclusion:*** **Most of the studies presented solution proposal combined with case studies. But, only a few of them address empirical evaluation. With the software architecture recovery evolution, more research combining recovery of product line architecture and empirical evaluation still necessary.**

Keywords**—Software Product Lines; Software Reuse; Software Architecture; Survey**

I. Introduction

Software Architecture Recovery (SAR) is the process that extracts architectural information from a software system [1]. As defined by Kazman et al. [2] and Deursen et al. [3], SAR is the process to obtain the architecture of an implemented system from the existing legacy system to support the understanding of the recovered system.

The Software Product Lines (SPL) approach provides the development of software systems based on reusable parts [4]. SPL products share a set of common features (commonalities) and have variabilities that distinguish the applications [5]. Because the development of an SPL involves the implementation of different structures, processes, interfaces and activities, it is relevant for product line practitioners to pay sufficient attention to its architecture.

In this sense, according to Gomaa [6], an Product Line Architecture (PLA) is an architecture for a family of products, which describes the mandatory, optional, and variable components in the SPL, and their interconnections. Moreover, the PLA can be defined as the "core" architecture that capture high level design for the SPL, including variation points and variants documented in the variability [5].

Despite the importance of PLA, Souza Filho et al. [7] raised the lack of guidelines for architecture recovery in SPL domain. SAR can help in commonality and variability identification within the products [8]. The PLA recovery process focus on representative members of the product line resulting in the architectural description for each product line member [9]. The combination of SAR with PLA is critical to maintain the assets (including architectural documentation and design artifacts) up-to-date [10], manage the variability at the architectural level [11], and maintain the architectural conformance among the products [12].

In this context, we performed a survey to gather data and evidence about the relationship between Product Line Architecture and the Software Architecture Recovery field. We could not identify any reviews, which investigated software architecture recovery focusing on SPL. Finally, we also performed an exploratory study to extend the investigation about the relationship between SAR and PLA.

A first step towards addressing product line architecture recovery is to identify current methods, techniques and processes used to recover product line architectures. Therefore, we define the goal of our study using the Goal-Question-Metric (GQM) approach [13] as follows:

Analyze the literature on product line architecture and software architecture recovery *for the purpose of* characterization *with respect to gather* information *from the point of view of* researchers and practitioners *in the context of* software product line research area.

Thus, the contributions of this paper are (i) review the literature about product line architecture and software architecture recovery relationship and its evolution over the years, (ii) investigate how the existing solution proposal support PLA recovery, and (iii) identify product line architecture recovery research trends.

The remainder of this paper is organized as following: Section II presents related work. Section III details the research method. Section IV presents the outcomes. In Section V, we discuss the findings and describes the threats to validity of this work. Section VI presents the exploratory study. Finally, in Section VII, we present the concluding remarks and future work.

II. Related Work

In 2007, Pollet et al. [14] presented a process-oriented software architecture reconstruction taxonomy. The authors proposed a classification based on the life time of SAR approaches. Later on, in 2009, Ducasse and Pollet [15] extended the research. The authors analyzed briefly some aspects of SAR in the SPL context.

For example, they affirm that reconstructed architecture is the basis for reuse investigation and migration for product lines. Due to its relevance and because they synthesized information from previously published studies, in our review, we considered these studies as secondary studies.

Souza Filho et al. [7] presented a systematic review on domain design approaches to understand and summarize empirical evidence about their activities, identifying directions, strengths, and weaknesses.

Although, these studies are relevant evidence, they did not consider product line architecture recovery deeply. Thus, in contrast, we aim to review the PLA recovery state-of-the-art and its evolution over the years, analyze the relationship between SAR and PLA, and identify SAR trends adaptable to PLA recovery. These research areas can work together and benefit from each other.

III. Research Method

The research design for this study was based on systematic literature review guidelines [16] to aim at a credible and fair evaluation of studies on product line architecture recovery. An important step is the protocol development. The protocol maps systematically the steps performed in the review and increases its rigor allowing study replication.

A. Research Questions (RQs)

Based on the study main goal, our review objective is to answer the following research questions:

- RQ 1: How does the relationship between product line architecture and software architecture recovery evolve over the years?
- RQ 2: How does the existing solution proposal support product line architecture recovery?
- RQ 3: What are the product line architecture recovery trends?

We define RQ1 to obtain an overview of the research area. We intend to understand the relationship between software architecture recovery and product line architectures. Moreover, we defined RQ2 to identify how the solution proposal support product line architecture recovery. Finally, RQ3 helps researchers to assess the product line architecture recovery trends.

B. Search Strategy

The search strategy enables the inclusion of relevant studies in the search results. The search was based on (i) preliminary searches in key venues such as Software Product Line Conference (SPLC), European Conference on Software Architecture (ECSA), and Working IEEE / IFIP Conference on Software Architecture (WICSA), (ii) trial searches using various combinations of search terms derived from the research questions, (iii) automatic search by executing search strings on search engines of electronic data sources, and (iv) the "snow-balling" process [17], in which the identified studies references were analyzed.

When manually searching the venues, we considered title, keywords, and abstract. On the other hand, the search string for automatic search consisted of three parts: recovery AND software architecture AND software product line. The words were combined through logical OR to form a string, as can be seen on following search string.

("recovery" OR "recover" OR "reconstruction" OR "reconstruct" OR "refactoring") AND ("architecture" OR "architecting" OR "design" OR "designing") AND ("product line" OR "product lines" OR "product-line" OR "product family" OR "product families" OR "SPL" OR "SPL" OR "PLA" OR "SPLA")

We changed the search string according to search features of electronic sources. For this reason, we used different search strings for different sources. However, these strings were semantically and logically equivalent.

C. Selection Criteria

In order to filter the studies and eliminate irrelevant ones (e.g. do not address the research questions), we adopted the following inclusion and exclusion criteria.

1) Inclusion:: The study must explore software architecture recovery in the software product line context, or present SAR solution proposals and how it supports product line architecture, and finally, presents product line architecture recovery trends. Moreover, we only included studies written in English.

2) Exclusion:: We excluded studies if their focus, or main focus, was not software architecture recovery or if they did not present topics related with reuse and product line architecture recovery. Finally, studies available as abstracts or presentations were excluded.

D. Data Collection

Based on the selection criteria, we selected 28 studies. They have been read in details to extract the data that address each research questions. We created an extraction form to collect data composed by the following information: author(s), year, title, source, research type, answer for each research question, and information to categorize the study.

During the extraction, we classified some studies in more than one category. Thus, primary studies can appear more than once. An extracted data record was kept in Excel spreadsheet for analysis. For each study, we collected the following fields:

Author(s), Year, Title, Venue, Venue Type, Research Type, Contribution Type, Study address PLA, Study address SAR, Recovery Type, Tool Support, Empirical Study. For each paper, data was extracted by one researcher and checked against the paper by another researcher. Finally, disagreements were resolved by discussions between researchers or by consulting an additional researcher. More details can be seen in the survey website[1].

E. Data Analysis

We summarized data from primary studies to answer the research questions. Therefore, we reviewed data manually to perform a classification scheme that helped to categorize the results of this study in tabular form using the research type facet defined by Wieringa et al. [18].

Thus, we classified the studies according to contribution type (Approach, Framework, Method, Process, Taxonomy, and Tool) and empirical research (Experiment, Case study, Survey, Ethnography, Action Research, and Mixed-Method Approach) [19].

We also defined the Recovery Type according to [15]. First, Bottom-up processes start with low-level knowledge to recover architecture. Second, Top-Down processes start with high-level knowledge to discover architecture by formulating conceptual hypotheses. Finally, Hybrid processes combine bottom-up with top-down processes.

IV. Outcomes

We used the extracted data to answer the research questions. Moreover, we describe the main findings and group the results according to each research question. The following tables and figures contain the studies number and classification. First, we give an overview of the identified studies and extracted information. Then, we answered the research questions by analyzing the data.

A. Characteristics of the studies

The 28 selected studies encompass the years 1998 through the first semester of 2015. They were gathered from 13 conferences, 1 workshop, and 4 journals. The remaining 5 studies were published as book chapter (2 studies), one technical report, and two Ph.D. Thesis.

As expected, the greater amount of studies in a single vehicle was found in the SPLC (4 studies), considered the most representative conference for the SPL engineering area, followed by the WICSA (2 studies), the European Conference on Software Maintenance and Reengineering (2 studies), and the International Conference on Software Engineering (2 studies).

Such a distribution give us the initial impression that most relevant studies in the field were only found in recent publications, i.e., as of the year 2009. Regardless the number of studies (93) removed from our final list for matching any of the exclusion criteria. Thus, we may notice a trend curve in the data, showing an increasing attention on the use of scientifically rigorous evaluation methods as a means to assess and make explicit the value of the proposed approaches for the SAR field.

[1]http://goo.gl/FJbsup

B. Results

Table I presents the list of 28 reviewed studies (26 primary studies and 2 secondary studies – S11 and S16). Based on the data, we can consider product line architecture recovery as a recent research area. For this reason, the findings discussed here should be considered as initial evidence or tendencies.

TABLE I
List of Reviewed Studies

ID	Ref.	Year	PLA	Rec.	R.T.	T.S.	E.R.
S1	[8]	1998	●	●	hybrid	-	case study
S2	[20]	2000	●	●	hybrid	-	case study
S3	[9]	2000	●	●	hybrid	-	case study
S4	[21]	2001	●	⊖	hybrid	●	case study
S5	[2]	2003	⊖	●	bottom-up	⊖	-
S6	[10]	2003	●	●	hybrid	⊖	-
S7	[22]	2004	⊖	●	hybrid	●	case study
S8	[23]	2004	⊖	⊖	top-down	-	-
S9	[24]	2005	●	●	bottom-up	-	case study
S10	[25]	2006	-	●	top-down	⊖	case study
S11	[14]	2007	⊖	●	All	⊖	survey
S12	[26]	2007	⊖	●	hybrid	-	case study
S13	[27]	2007	⊖	●	hybrid	⊖	experiment
S14	[28]	2007	●	●	hybrid	●	case study
S15	[12]	2009	●	●	hybrid	●	case study
S16	[15]	2009	⊖	●	All	⊖	survey
S17	[29]	2010	⊖	●	hybrid	●	experiment
S18	[30]	2011	●	-	-	-	-
S19	[31]	2011	●	-	-	-	-
S20	[32]	2011	●	●	top-down	●	case study
S21	[33]	2012	-	●	bottom-up	-	-
S22	[34]	2012	●	●	bottom-up	●	case study
S23	[35]	2013	●	-	-	-	case study
S24	[36]	2013	-	●	bottom-up	⊖	survey
S25	[37]	2014	●	●	hybrid	⊖	case study
S26	[38]	2014	-	●	bottom-up	●	-
S27	[11]	2015	●	●	bottom-up	⊖	experiment
S28	[39]	2015	-	●	bottom-up	●	experiment

Legend: [●] Characteristic clearly addressed by the study, [⊖] The study encourages the use of such characteristic, but do not provide any implementation (e.g., it states an external tool is used, but no detail is provided), [-] Characteristic not mentioned in the study, [PLA] Product Line Architecture, [Rec.] Recovery, [R.T.] Recovery Type, [T.S.] Tool Support, and [E.R.] Empirical Research.

Most of the studies (71%) discuss PLA, SAR, and their relationship. Although some of them (17%) did not address Product Line Architecture, these studies were included because (i) they considered at least one SPL aspect such as software reuse or variability, or (ii) they described research trends adaptable to PLA.

Regarding recovery type, 32% of the studies defined bottom-up processes, 11% top-down, 39% hybrid, 2 studies considered these three types, and 3 studies did not address any of them. A total of 9 studies proposed the development of tools to support software architecture recovery process, 9 of them use external tools (developed by third parties), but no detail is provided, and 10 studies did not mention tool support.

Regarding empirical research, 50% of the studies presented case studies, 14% presented experiments, 11% presented surveys, and 25% did not present empirical studies. The numbers indicate the necessity of more empirical evaluation.

In combination with Table I, we present a timeline in Figure 1 that gathers additional information. It shows papers distribution over the years until 2015. Moreover, Figure 1 shows how many papers were found in journals, conferences, book chapter, technical reports, PhD thesis, or workshops.

C. RQ 1: How does the relationship between product line architecture and software architecture recovery evolve over the years?

To answer this question, we continue the timeline analysis. As can been seen in Figure 1, the timeline summarizes the studies evolution over the years. We identified three phases: (i) definition of basic concepts – initial studies considering PLA and SAR; (ii) field consolidation – first journals addressing PLA and SAR; and, (iii) appearance of new research trends.

Eixelsberger performed the first studies combining PLA and SAR (S1, S2, and S3). The author started investigating software architecture recovery of embedded software [1] and architectural structure [40]. Later, the research evolved to architecture recovery of product lines.

S6 proposed the architecture recovery of individual systems from the same domain and compared them to allow the SPL Design Reference Architecture (DSSA) creation. The study presented techniques along with tools integrated into the reconstruction process to reduce the manual effort. Similarly, S9 recovered the conceptual architecture from legacy applications of the same domain. The authors identified services based on the extracted object relationship diagram.

Further, two studies (S14 and S15) extended the reflexion method to compare product implementations to the product architecture and to compare product architectures to the SPL architecture. The studies used clustering algorithms to consolidate software variants into product lines, assuming that implementations and architectures of the variants are similar.

From 2011 and on, studies presented trends (further discussed in RQ 3) and demonstrated the maturity on the development of SAR solution proposals. The studies are mostly focused on recover the variability at the requirement level (S3, S6, S9, S20, S22). Few works aim at PLA recovery focusing the variability at the architectural level (S9, S14, S15). Only one study (S27) proposed fully automated recovery approach, but it recovered the products architecture.

D. RQ 2: How does the existing solution proposal support product line architecture recovery?

Most studies (18 of 28, 64%) presented solution proposal. Table II organizes the solution proposal per research contribution. Studies presented eight tools, five approaches, five methods, four processes, and two frameworks.

Studies S1, S2, and S3 presented a set of methods that uses information from members of a family of legacy software systems to recovery the architectural properties in the systems and build architectural descriptions. S4 proposed a bottom-up approach to recover architectural representations of existing systems and a top-down approach to map known architecture architectural styles and attributes onto the recovered architecture.

TABLE II
SOLUTION PROPOSAL PER RESEARCH CONTRIBUTION

Classes	Acronym	Studies	Count
Tool	[To.]	S5, S7, S14, S15, S17, S20, S22, S26	8
Approach	[Ap.]	S8, S12, S13, S17, S27	5
Method	[Me.]	S3, S4, S7, S14, S15	5
Process	[Pr.]	S5, S6, S18, S20	4
Framework	[Fr.]	S1, S21	2

Moreover, S5 describes the process of architecture reconstruction using the Architecture Reconstruction and Mining (*ARMIN*) tool. The representation can be used as a way for identifying components for reuse or for establishing an architecture-based software product line. S6 combines solution proposals (techniques and tools) to analyze related systems, determine common assets, and integrate these assets into the design of a reference architecture. On the other hand, S7 proposes a method, called *NIMETA*, for architecture reconstruction based on the recovery of the architecturally significant views.

Not always tools implement software architecture recovery techniques. For example, S12, S13, and S17 presented *ArchMine*, an architecture recovery approach based on dynamic analysis and data mining. S14 and S15 described a method and supporting tools to compare software variants at the architectural level. The method consists of the specification of the module view and the mapping of implementation components onto the module view.

Furthermore, S20 presented a tool-supported approach to reverse engineer architectural feature models. Similarly, S22 proposed a history-sensitive heuristics (supported by a tool *RecFeat*) for the recovery of features in code of degenerate program families.

S21 proposed to explore the use of machine learning techniques to automatically recover software architecture from software artifacts, S26 presented *ArchViz* a tool that partially automate the analysis of recovered architectures. Finally, S27 proposed an approach to reverse engineer the architecture of a set of product variants. The study relies on Formal Concept Analysis (FCA) to analyze the variability.

E. RQ 3: What are the product line architecture recovery trends?

We analyzed PLA recovery trends and SAR used to recovery single systems architecture adaptable to PLA recovery. Primarily, the research community demands more studies and evidence presenting empirical studies such as performed in S10, S24, and S28. Majority of studies consist of solution proposal (64%), next step suggests the development of studies describing experiments, surveys, and empirical evaluation.

S21 indicates a trend towards software architecture recovery automation. The study goal is to develop techniques allowing automatic architecture recovery from source code. The study

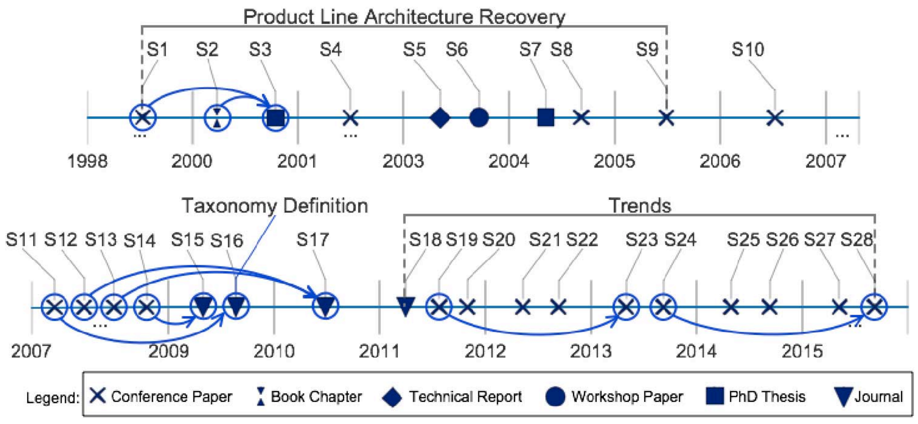
Product Line Architecture Recovery
S1 S2 S3 S4 S5 S6 S7 S8 S9 S10
1998 2000 2001 2003 2004 2005 2006 2007
Taxonomy Definition
Trends
S11 S12 S13 S14 S15 S16 S17
S18 S19 S20 S21 S22 S23 S24 S25 S26 S27 S28
2007 2009 2010 2011 2012 2013 2014 2015
Legend: Conference Paper, Book Chapter, Technical Report, Workshop Paper, PhD Thesis, Journal

Fig. 1. Timeline

proposes the use of machine learning techniques to automatically recover software architecture. On the other hand, S22 proposes history-sensitive heuristic for features recovery in source of degenerating program families.

Another trend is related to design product line architecture based on reference architectures (S19 and S23). In S23, the authors present a process, named *ProSA-RA2PLA*, that uses reference architectures for building product line architectures systematically.

There is a trend related to software architecture recovery tools development (S22 and S26). New tools are necessary to support solution proposal (e.g. approaches, frameworks, heuristic, methods, etc) in product line architecture recovery. One brief example might clarify this concept, it is feasible to use SAR tool to recover reference architectures and enable product line architecture construction (as proposed in S19 and S23).

Finally, S25 characterizes a research trend related to architectural bad smells in the context of PLA through an exploratory study. And, recently, S27 presents research trend related to identify variability and dependencies among architectural variants at the architectural level.

V. Discussion

In the following, we provide a summary of the main findings, limitations to the review, and threats to validity.

A. *Main Findings*

The goal of this review is to identify the studies to define PLA recovery by analyzing the relationship between PLA and SAR over the years, verify whether the existing solution proposal are being used in product line architecture recovery, and identify the product line architecture recovery trends.

While most of the identified studies are solution proposal (see Figure 2 and Table III), few studies provide details on validation. Thus, it is difficult to draw general conclusions. Moreover, according to Table III, a small number of opinion or theoretical studies investigates the area.

TABLE III
Number of studies per research type

Classes	Acronym	Studies	Count
Solution Proposal	[S.P.]	S1, S3, S4, S5, S6, S7, S8, S12, S13, S14, S15, S17, S18, S20, S21, S22, S26, S27	18
Experience Papers	[E.P.]	S2, S9, S10, S19, S25	5
Evaluation Research	[E.R.]	S24, S28	2
Philosophical Papers	[P.P.]	S11, S16	2
Validation Research	[V.R.]	S23	1

Figure 2 shows the relationship among contribution type, research type, and research questions.

Regarding RQ1, we identified that some studies evolved from conference papers to journals (e.g. S11 evolved to S16; S12 and S13 evolved to S17). We believe that journals are more rigorous, in other words, studies discussed their research topic in depth. In S19, the authors explored the use of reference architectures in the development of product line artifacts. Further, the authors evolved the research to S23, presenting a process to systematize the use of existing reference architectures to build PLA.

During our review, we also identified the evolution of software architecture recovery techniques (S24 and S28), processes (S18 and S23), and tools (S26 and S27). They became more complex in solving specific issues of SAR field. Moreover, we found tools developed to support heuristics (S22), approaches (S17 and S19), and frameworks (S21).

Even with the evolution in this aspect, there is no standardization regarding inputs and outputs. The majority of the solution proposals work independently. There is no reuse of tools, each study developed tools focused to their context.

Finally, the software architecture recovery field supports product line architecture recovery by providing solution proposal adaptable to SPL context (e. g. adapted processes, tools, approaches, frameworks, and so on). Few works focused on

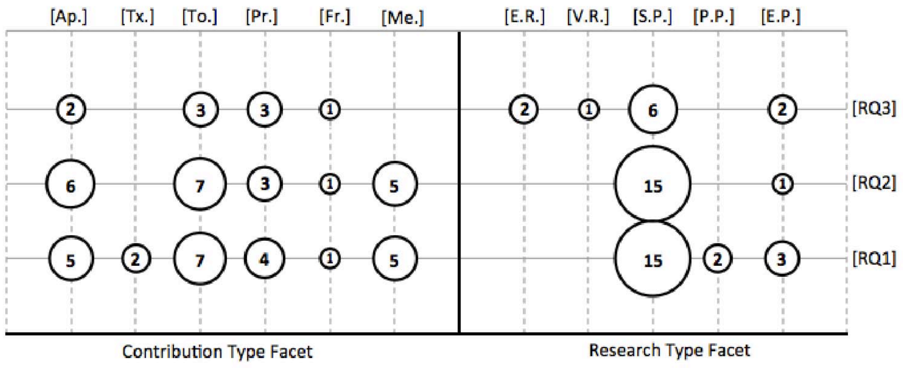

Fig. 2. Contribution Type versus Research Type versus Research Questions

fully automated recovery of variability at the architectural level.

Answering RQ2, the majority of solution proposals are used to recover the architecture from legacy systems source code to provide the SPL reference architecture (S1, S2, S3, S4, S5, S6, S12, S13, S17, S21). In other words, these solution proposals support the migration from legacy code into SPL and five of them (S4, S5, S12, S13, and S17) presented architecture mining techniques to migrate legacy systems into product line.

As a matter of fact, new approaches, tools, and techniques were developed during these years addressing specific problems such as recovering architectural variability (S21 and S27), recovery of feature models (S20 and S22), and static architecture variants reconstruction allowing module view reconstruction (S14 and S15).

To the best of our knowledge, we believe that SAR solution proposals used to recover single systems architecture could be used to recover software architecture of product line projects. For example, it is feasible to adapt the architecture recovery techniques presented in S24 and S28 to solve product line architecture recovery challenges.

For this reason, we performed an exploratory study further detailed in Section VI. Finally, another example, S26 developed *ArchViz*, a tool to compare recovery architectures that also could be used to support product line architecture.

Regarding RQ3, we identified the following research trends:

- T1: Architectural variability recovery (S21 and S27);
- T2: Development of empirical studies (S19, S23, S24, S25, and S28);
- T3: PLA reconstruction based on reference architecture (S19 and S23);
- T4: Web-based tools implementation to support PLA recovery (S20, S22, and S26);
- T5: PLA recovery to identify architectural bad smells in SPL context (S25).

Furthermore, the appearance of new development trends such as Cloud Computing (CC)[41], [42], [43] enables the development of new solutions in product line architecture recovery research field. As we mention before, there is an opportunity of extending these new technologies to solve some product line architecture recovery challenges. Because CC eliminates the platform dependency, the development of web based tools (S26) could solve the SAR input and output standardization problem.

Finally, there is a number of studies that did not explicitly answer the research questions set out in this study. Table IV presents the mapping between the studies and the research question. Even with some studies addressing the research questions, more evidence is required to consolidate the research field on product line architecture recovery. This topic is still an emerging research area and further investigation is necessary.

TABLE IV
NUMBER OF STUDIES PER RESEARCH QUESTION

RQ	Studies	Count
RQ1	S1, S2, S3, S4, S5, S6, S7, S8, S9, S11, S12, S13, S14, S15, S16, S17, S20, S22, S25, S27	20
RQ2	S1, S3, S4, S5, S6, S7, S8, S10, S12, S13, S14, S15, S17, S20, S22, S27	16
RQ3	S18, S19, S20, S21, S22, S23, S24, S25, S26, S27, S28	11

B. Limitations of the Review and Threats to Validity

During automatic search, our main objective was to ensure the selected papers completeness. As mentioned, we searched a number of key venues. We also excluded irrelevant papers to reduce the researchers bias that affect the paper selection process. There are some threats to the validity of our survey, which we briefly describe along with the mitigation strategy for each threat.

- *Publication bias*: We cannot guarantee that all relevant primary studies were selected. It is possible that some relevant studies were not chosen during the search process. We mitigated this threat as much as possible, by following references in the primary studies.
- *Research questions*: The research questions we defined cannot provide complete coverage of the SAR field. We considered this as a threat, however, we discussed with researchers from our group and with an expert in the area to validate the questions. We also performed an

exploratory study (see Section VI) in order to expand the analysis and mitigate this threat.
- *Search conducted*: Although digital databases have incompatible search rules, we adapt our search strings for each digital database.

VI. Exploratory Study

The exploratory study aims at characterizing the relationship between PLA and SAR through the extension of RQ 2: How does the existing solution proposal support product line architecture recovery?

We are particularly interested in investigating whether the same SAR techniques often used to recover single systems architecture (S11, S16, S24, S26, and S28) can also be used to recover PLAs. Further, we discuss the implications on using these recovery solution proposals in the SPL context.

A. Study Settings

In order to address the aforementioned issues, we selected an open source SPL project, Prop4J[2]. Prop4J is a library for arbitrary propositional formulas used to solve boolean satisfaction and optimization problems based on Sat4J[3] that has been transformed into a product line. Because there was no architectural artifacts related to this project, we undertook a bottom-up recovery process to identify the descriptive architecture of 3 products (P1, P2, and P3).

To answer the research question (RQ2), we compared the clustering results obtained from the architecture recovery of the SPL products. Our study was performed in four main stages as shown in Figure 3; for each product, all of the steps from Figure 3 were performed.

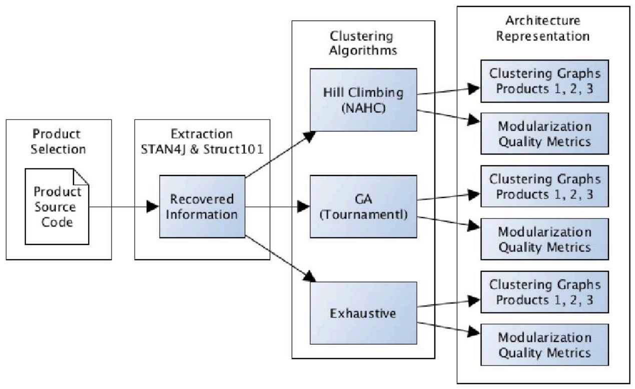
Product Selection
Product Source Code
Extraction STAN4J & Struct101
Recovered Information
Clustering Algorithms
Hill Climbing (NAHC)
GA (Tournamentl)
Exhaustive
Architecture Representation
Clustering Graphs Products 1, 2, 3
Modularization Quality Metrics
Clustering Graphs Products 1, 2, 3
Modularization Quality Metrics
Clustering Graphs Products 1, 2, 3
Modularization Quality Metrics

Fig. 3. Study Design

In summary, the study workflow consisted in (1) instantiating an SPL product, (2) extracting its descriptive architecture from the source code using STAN4J[4] and Struct101[5] extracting tools, (3) applying the clustering algorithms, and (4) analyzing the generated graphs and metrics. Due to space constraints, in this paper we focus on the description of Prop4J SPL products.

The exploratory study supplemental material such as complete descriptive statistics, metrics, and recovered graphs can be reached in the paper resource website[6].

B. Selected Recovery Technique

For our evaluation, we selected the architecture recovery technique called Bunch [44]. The technique uses dependencies to determine clusters and transforms the architecture recovery problem into an optimization problem.

An optimization function called Modularization Quality (MQ) represents the quality of a recovered architecture. It measures the *trade-off* between inter-connectivity (i.e., connections between the components of two distinct clusters) and intra-connectivity (i.e., connections between the components of the same cluster) [45].

Bunch provided the following metrics:

- M1 – IncrementalMQ: It is calculated by summing the Cluster Factor (CF) for each cluster. The CF is defined as a normalized ratio between the total weight of the internal edges (edges within the cluster) and half of the total weight of external edges (edges that exit or enter the cluster) [45];
- M2 – TurboMQ: it is based on a variation of *IncrementalMQ* metric;
- M3 – TurboMQIncrW: It is based an variation of *TurboMQ* metric;
- M4 – ITurboMQ: It is also based an variation of *TurboMQ* metric; and,
- M5 – BasicMQ: It demonstrates the trade-off between inter-connectivity and intra-connectivity by rewarding the creation of highly-cohesive clusters, while penalizing the creation of too many inter-edges. The value of the BasicMQ is bounded between -1 (no cohesion within the subsystems) and 1 (no coupling between the subsystems) [45].

Furthermore, Bunch offers five distinct clustering algorithms. In this study, we used 3 of them: Hill Climbing (Nearest Ascent Hill Climbing – NAHC), Exhaustive, and Genetic Algorithm (GA). More details about these algorithms can be found in [45]. Because the other 2 algorithms (SAHC and Roulette Wheel) are variations, we did not include them in the study.

Exhaustive - It is the only deterministic algorithm provided by bunch. The algorithm tests all the partition possibilities and returns the one with higher MQ value. It is only recommended to systems containing 17 elements at maximum [45]. When elements above 17 are used, the algorithm starts to loose performance.

NAHC - Unlike Exhaustive, this algorithm did not test all possible partitions. Initially, a starting point is generated randomly, from this point its neighboring partitions are analyzed.

[2]http://spl2go.cs.ovgu.de/projects/1
[3]http://www.sat4j.org/
[4]http://stan4j.com/
[5]http://structure101.com/

[6]http://goo.gl/gFQObI

Mitchell [45] stated that a neighboring partition is made by reallocating a member from one partition to another.

Thus, the algorithm calculates the MQ value for the neighboring partitions. If a higher value is found, an iteration is made and continues until it converges to a maximum value. Because the start point is always randomly generated, it is not possible to obtain optimal results all times.

For this reason, Bunch allows the user to choose the percentage of neighboring partitions that will be analyzed ranging from 0% (NAHC) where the first neighboring partition with higher MQ will be chosen and 100%(SAHC) where all neighboring partitions will be analyzed.

GA - Bunch also provides genetic algorithms to search for the optimal results. This algorithm is too complex and extensive. For this reason, we focused on important details, further explanation can be found in [45].

We used the tournament method to choose random partitions (individuals) and reproduce the next generation based on the MQ value. This algorithm should be calibrated to each analyzed system because it only search for the best result of a fixed number of generations. Mitchell [45] recommends that safe numbers of iterations and population size are 100 times the number of elements and 10 times the number of elements respectively.

Finally, Bunch uses hill-climbing and genetic algorithms to find a partition (i.e., a grouping of software entities into clusters) that maximizes MQ. Due to the non-determinism of the clustering algorithms, we executed each algorithm twenty times and reported the mean values of those twenty executions.

C. Descriptive Statistic Analysis

Table V presents the data collected from the SPL products architecture recovery. We gathered the information from three products (P1, P2, and P3) using three clustering algorithms organized according to the five metrics presented previously.

Figure 4 shows the boxplot for the clustering algorithms. The horizontal lines crossing the boxplot indicates the average, while the dots show how the values spread over the Y-axis. The Exhaustive and GA algorithms presented similar results. On the other hand, the Hill Climbing algorithm presented similar mean value, but it shows different results.

In order to evaluate if any of the clustering algorithms is significantly different from the others, we tested the results statistically through ANOVA (Analysis of Variance) [13].

The *p-value* for the ANOVA was 3,11e-08, giving us sufficient evidence to conclude that at least one clustering algorithm has average performance significantly different from the others. To identify the specific means that are different, we performed a multiple comparison procedure called Tukey's test.

Regarding GA and Exhaustive algorithms, there is no significant differences between them (*p-value*=0.9906). On the other hand, Tukey's test indicates that Hill Climbing algorithm is better than GA (*p-value*=0.0000006) and Exhaustive (*p-value*=0.0000011) algorithms.

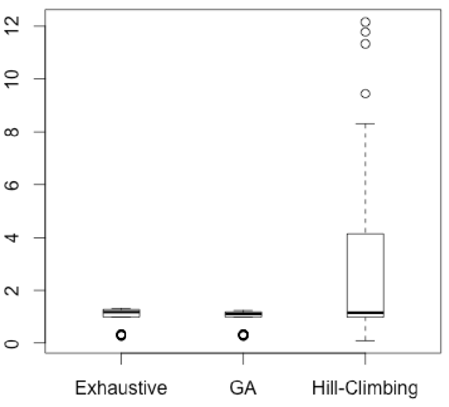

Fig. 4. Clustering Algorithms Boxplot

D. Findings

At first, we used STAN4J and Struct101 to obtain the products descriptive architectures. We noticed some minor differences between these extracted architectures. In the end, the results were almost similar (as can be seen on Table V). There are no significant differences among the collected information.

Similarly to S24 and S28, the Hill Climbing (NAHC) algorithm showed the best performance. It was the fastest in achieving higher MQ values. Moreover, the higher MQ value was acquired combining NAHC with iTurboMQ, represented by M3 on Table V.

We also proved statistically that NAHC presented best results. On the other hand, the exhaustive algorithm was the most stable because it generated partitions in a deterministic way. However, it was the slowest of the three even with incremental metrics.

Regarding the metrics used, the genetic algorithm showed fewer variations between its results. Even though, it stills slower than the NAHC algorithm and could not generate higher values of MQ.

Finally, RQ2 tried to comprehend if the software architecture recovery techniques can be applied on the SPL context. This preliminary study showed us that it is possible. However, we found some problems.

Because those techniques were made to recover single systems architectures, we could not recover the PLA reference architecture. Therefore, we recovered the descriptive architecture from the products.

In this way, some SPL characteristics such as variability could not be seen clearly. Nevertheless, when we compared the products architecture manually, it was possible to visualize the PLA reference architecture including its commonalities and variabilities.

E. Exploratory Study Threats to Validity

There are some threats to the validity of our exploratory study, which we briefly describe along with the mitigation strategy for each threat.

TABLE V
METRICS - STAN4J VS. STRUCT101 - WITH WEIGHT

SPL Product	**Algorithm**	**STAN4J - with weight**					**Struct101 - with weight**				
		M1	M2	M3	M4	M5	M1	M2	M3	M4	M5
P1	Hill Climbing	0.9894	0.9788	0.9936	0.9851	0.1067	0.9975	0.9981	0.9979	0.9974	0.1348
	GA	1	1	1	1	0.3333	1	1	1	1	0.3333
	Exhaustive	1	1	1	1	0.3333	1	1	1	1	0.3333
P2	Hill Climbing	11.7781	1.2905	6.4470	4.4865	0.1382	12.1511	1.2898	6.8072	11.3286	0.1481
	GA	1.1839	1.1954	1.1861	1.1890	0.3254	1.22	1.2142	1.2068	1.2030	0.3254
	Exhaustive	1.3094	1.3094	1.3094	1.3094	0.3254	1.3164	1.3264	1.3264	1.3264	0.3254
P3	Hill Climbing	3.5727	1.1853	4.8965	3.3984	0.1753	3.8010	1.1523	4.5923	5.5311	0.1594
	GA	1.1316	1.1298	1.1367	1.1382	0.29	1.1103	1.1049	1.1307	1.116	0.29
	Exhaustive	1.194	1.194	1.194	1.194	0.29	1.1787	1.1787	1.1787	1.1787	0.29
SPL Product	**Algorithm**	**STAN4J - without weight**					**Struct101 - without weight**				
		M1	M2	M3	M4	M5	M1	M2	M3	M4	M5
P1	Hill Climbing	1.0792	1.0559	1.0623	1.0136	0.1754	1.0803	1.0623	1.0375	1.0439	0.1707
	GA	1.1111	1.1111	1.1111	1.1111	0.3333	1.1111	1.1111	1.1111	1.1111	0.3333
	Exhaustive	1.1111	1.1111	1.1111	1.1111	0.3333	1.1111	1.1111	1.1111	1.1111	0.3333
P2	Hill Climbing	7.9837	1.1608	8.3056	9.4429	0.1558	7.8040	1.1296	6.3180	7.5521	0.1166
	GA	1.1334	1.1439	1.1428	1.1341	0.3254	1.1464	1.1492	1.1473	1.1491	0.3254
	Exhaustive	1.2647	1.2647	1.2647	1.2647	0.3254	1.2517	1.2517	1.2517	1.2517	0.3254
P3	Hill Climbing	2.4653	1.2367	3.2085	2.2013	0.1734	3.2241	1.2466	1.9191	2.4317	0.1545
	GA	1.2302	1.2307	1.2316	1.2379	0.29	1.2198	1.2349	1.2334	1.2283	0.29
	Exhaustive	1.2759	1.2759	1.2759	1.2759	0.29	1.2759	1.2759	1.2759	1.2759	0.29

Legend: [M1] IncrementalMQ, [M2] TurboMQ, [M3] TurboMQIncrW, [M4] iTurboMQ, [M5] BasicMQ

- *Selection of Prop4J open source SPL project*: For the initial analysis, we selected a project with a small number of generated clusters. To mitigate this threat we generated three products based on this product line, each product had different features, thus different sizes and components, increasing the analysis scope.
- *Bunch generates random results*: The algorithms implemented on Bunch are non-deterministic, each execution generates a different result. To mitigate this issue, we executed each algorithm twenty times for each metric. The results shown in Table V are the mean values of those twenty executions.
- *The dependency graphs may not be equal*: Some graphs obtained by STAN4J and Struct101 presented differences in relation to weight of dependency relationships of graph elements. To mitigate this threat, we compared the graphs and the results showed no significant difference between those tools, that means that whichever tool is used the results will be the same.
- *Only the products architecture was recovered*: Due to Bunch algorithms limitations, it was not possible to recover the PLA, to solve this problem, we recovered the products architectures and perform a manual analysis in search of common elements between the products as well as the variability points.

VII. CONCLUSION

In this paper, we presented a literature survey and an exploratory study to gather initial evidence for understanding how product line architecture relates to software architecture recovery.

Regarding the survey, the selected studies fell into three thematic groups: understand the relationship between product line architecture and software architecture recovery, characterize how the existing solution proposals support PLA, and identify the product line architecture recovery trends.

The PLA and SAR relationship evolved from the definition of basic concepts to provide solution proposals that solve specific problems. The solution proposals majority are used to recover the SPL reference architecture based on legacy systems. Only few studies address empirical evaluation such as experiments, surveys, mixed-methods, and so on. In this context, more empirical research is still necessary. There is a clear need to establish standardization for future field studies.

Regarding the exploratory study, we investigated how to use the same single systems SAR techniques to recover PLAs. Hence, to adapt the recovery techniques to SPL context, we recovered the descriptive architecture from SPL products and identified that NAHC clustering algorithm obtained the best performance results.

As contribution for researchers, the survey summarized the information providing a list of primary studies and identified initial evidence for understanding the relationship between PLA and SAR. As contributions for the practitioners, we created the study design during the exploratory study execution to allow the study replication.

As future work, we aim to automatically perform the "comparison" between the product architectures to provide the identification of variability points in the PLA. We also intend to develop a metamodel for standardizing the PLA creation. Finally, we intend to perform a controlled experiment extending the exploratory study.

ACKNOWLEDGMENT

The authors are grateful to the anonymous reviewers for the thorough feedback. This work was funded by IFBA grants

EDITAL Number BP003-04/2014/PRPGI.

References

[1] W. Eixelsberger, L. Warholm, R. Klösch, and H. Gall, "Software architecture recovery of embedded software," in *Proceedings of the 19th International Conference on Software Engineering*. ACM, 1997.

[2] R. Kazman, L. O'Brien, and C. Verhoef, "Architecture Reconstruction Guidelines," Carnegie Mellon University and Software Engineering Institute, Tech. Rep., 2003.

[3] A. van Deursen, C. Hofmeister, R. Koschke, L. Moonen, and C. Riva, "Symphony: view-driven software architecture reconstruction," in *Fourth Working IEEE/IFIP Conference on Software Architecture*, 2004, pp. 122–132.

[4] S. Apel, D. Batory, C. Kastner, and G. Saake, *Feature-Oriented Software Product Lines*. Springer, 2013.

[5] K. Pohl, G. Böckle, and F. J. v. d. Linden, *Software Product Line Engineering: Foundations, Principles and Techniques*. Springer-Verlag, 2005.

[6] H. Gomaa, *Designing Software Product Lines with UML: From Use Cases to Pattern-based Software Architectures*. Addison-Wesley, 2005.

[7] E. de Souza Filho, R. de Oliveira Cavalcanti, D. S. Neiva, T. B. Oliveira, L. Lisboa, E. de Almeida, and S. de Lemos Meira, *Evaluating Domain Design Approaches Using Systematic Review*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2008, vol. 5292, ch. 6, pp. 50–65.

[8] W. Eixelsberger, M. Ogris, H. Gall, and B. Bellay, "Software architecture recovery of a program family," in *Proceedings of the 20th International Conference on Software Engineering*. ACM, 1998, pp. 508–511.

[9] W. Eixelsberger, "Software architecture recovery of product lines," Ph.D. dissertation, Universität Klagenfurt Fakultät für Wirtschaftswissenschaften und Informatik, 2000.

[10] M. Pinzger, H. Gall, J.-F. Girard, J. Knodel, C. Riva, W. Pasman, C. Broerse, and J. Wijnstra, *Architecture Recovery for Product Families*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2003, vol. 3014, ch. 26, pp. 332–351.

[11] A. Shatnawi, A. Seriai, and H. Sahraoui, "Recovering Architectural Variability of a Family of Product Variants," in *Software Reuse for Dynamic Systems in the Cloud and Beyond*, ser. Lecture Notes in Computer Science. Springer International Publishing, 2015, vol. 8919, pp. 17–33.

[12] R. Koschke, P. Frenzel, A. Breu, and K. Angstmann, "Extending the reflexion method for consolidating software variants into product lines," *Software Quality Journal*, vol. 17, no. 4, pp. 331–366, 2009.

[13] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, and B. Regnell, *Experimentation in Software Engineering*. Springer, 2012.

[14] D. Pollet, S. Ducasse, L. Poyet, I. Alloui, S. Cimpan, and H. Verjus, "Towards a process-oriented software architecture reconstruction taxonomy," in *11th European Conference on Software Maintenance and Reengineering*, 2007, pp. 137–148.

[15] S. Ducasse and D. Pollet, "Software architecture reconstruction: A process-oriented taxonomy," *IEEE Transactions on Software Engineering*, vol. 35, no. 4, pp. 573–591, 2009.

[16] B. Kitchenham and S. . Charters, "Guidelines for performing systematic literature reviews in software engineering," Keele University and University of Durham, Tech. Rep., 2007.

[17] D. Budgen, M. Turner, P. Brereton, and B. Kitchenham, "Using Mapping Studies in Software Engineering," in *Proceedings of PPIG 2008*. Lancaster University, 2008, pp. 195–204.

[18] R. Wieringa, N. Maiden, N. Mead, and C. Rolland, "Requirements engineering paper classification and evaluation criteria: A proposal and a discussion," *Requir. Eng.*, vol. 11, no. 1, pp. 102–107, 2005.

[19] S. Easterbrook, "Empirical research methods for software engineering," in *Proceedings of the Twenty-second IEEE/ACM International Conference on Automated Software Engineering*. ACM, 2007.

[20] W. Eixelsberger and H. Beckman, *The TCS Experience with the recovery of Family Architecture*. Addison-Wesley, 2000, pp. 209 – 231.

[21] C. Stoermer and L. O'Brien, "Map - mining architectures for product line evaluations," pp. 35–44, 2001 2001.

[22] C. Riva, "View-based Software Architecture Reconstruction," Ph.D. dissertation, Vienna University of Technology, 2004.

[23] C. Chiang and R. Y. Lee, "Developing tools for reverse engineering in a software product-line architecture," pp. 42 – 47, 2004.

[24] K. C. Kang, M. Kim, J. Lee, and B. Kim, "Feature-oriented reengineering of legacy systems into product line assets: A case study," in *Proceedings of the 9th International Conference on Software Product Lines*. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 45–56.

[25] J. Knodel, M. Lindvall, D. Muthig, and M. Naab, "Static evaluation of software architectures," in *Proceedings of the 10th European Conference on Software Maintenance and Reengineering*, March 2006, pp. 10 pp.–294.

[26] A. P. V. Vasconcelos and C. M. L. Werner, "Architectural elements recovery and quality evaluation to assist in reference architectures specification," 2007.

[27] A. Vasconcelos and C. Werner, *Architecture Recovery and Evaluation Aiming at Program Understanding and Reuse*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2007, vol. 4880, ch. 5, pp. 72–89.

[28] P. Frenzel, R. Koschke, A. Breu, and K. Angstmann, "Extending the Reflexion Method for Consolidating Software Variants into Product Lines," in *14th Working Conference on Reverse Engineering*, Oct 2007, pp. 160–169.

[29] A. Vasconcelos and C. Werner, "Evaluating reuse and program understanding in archmine architecture recovery approach," *Information Sciences*, vol. 181, no. 13, pp. 2761–2786, 2010.

[30] Y.-G. Kim, S. K. Lee, and S.-B. Jang, "Variability management for software product-line architecture development," *International Journal of Software Engineering and Knowledge Engineering*, vol. 21, no. 07, pp. 931–956, 2011.

[31] E. Y. Nakagawa, P. O. Antonino, and M. Becker, "Exploring the use of reference architectures in the development of product line artifacts," in *Proceedings of the 15th International Software Product Line Conference, Volume 2*. New York, NY, USA: ACM, 2011, pp. 28:1–28:8.

[32] M. Acher, A. Cleve, P. Collet, P. Merle, L. Duchien, and P. Lahire, "Reverse Engineering Architectural Feature Models," in *Proceedings of the 5th European Conference on Software Architecture*. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 220–235.

[33] H. Sajnani, "Automatic software architecture recovery: A machine learning approach," pp. 265–268, 2012.

[34] C. Nunes, A. Garcia, C. Lucena, and J. Lee, "History-sensitive heuristics for recovery of features in code of evolving program families," 2012.

[35] E. Y. Nakagawa, M. Becker, and J. C. Maldonado, "Towards a Process to Design Product Line Architectures Based on Reference Architectures," in *Proceedings of the 17th International Software Product Line Conference*. New York, NY, USA: ACM, 2013, pp. 157–161.

[36] J. Garcia, I. Ivkovic, and N. Medvidovic, "A comparative analysis of software architecture recovery techniques," pp. 486–496, 2013.

[37] H. S. de Andrade, E. Almeida, and I. Crnkovic, "Architectural Bad Smells in Software Product Lines: An Exploratory Study," in *Proceedings of the 11th Working IEEE / IFIP Conference on Software Architecture*. New York, NY, USA: ACM, 2014, pp. 12:1–12:6.

[38] V. Zapalowski, I. Nunes, and D. J. Nunes, "Archviz: a tool to support architecture recovery research," 2014.

[39] T. Lutellier, D. Chollak, J. Garcia, L. Tan, D. Rayside, N. Medvidovic, and R. Kroeger, "Comparing Software Architecture Recovery Techniques Using Accurate Dependencies," in *Proceedings of the IEEE/ACM International Conference on Software Engineering (ICSE)*, June 2015.

[40] W. Eixelsberger, M. Kalan, M. Ogris, H. Beckman, B. Bellay, and H. Gall, "Recovery of architectural structure: A case study," in *Development and Evolution of Software Architectures for Product Families*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 1998, vol. 1429, pp. 89–96.

[41] S. Zhang, S. Zhang, X. Chen, and X. Huo, "Cloud Computing Research and Development Trend," in *Second International Conference on Future Networks*, Jan 2010, pp. 93–97.

[42] M. Mollah, K. Islam, and S. Islam, "Next generation of computing through cloud computing technology," in *2012 25th IEEE Canadian Conference onElectrical Computer Engineering*, April 2012, pp. 1–6.

[43] A. Gajbhiye and K. Shrivastva, "Cloud computing: Need, enabling technology, architecture, advantages and challenges," in *2014 5th International Conference - Confluence The Next Generation Information Technology Summit (Confluence)*, Sept 2014, pp. 1–7.

[44] S. Mancoridis, B. Mitchell, Y. Chen, and E. Gansner, "Bunch: a clustering tool for the recovery and maintenance of software system structures," in *IEEE International Conference on Software Maintenance Proceedings*, 1999, pp. 50–59.

[45] B. S. Mitchell, "A heuristic search approach to solving the software clustering problem," Ph.D. dissertation, Drexel University, 2002.