International Journal of Software Engineering and Computer Systems (IJSECS)
ISSN: 2289-8522, Volume 4 Issue 2, pp. 19-37, August 2018

DOI: https://doi.org/10.15282/ijsecs.4.2.2018.2.0046

REPRESENTING VARIABILITY IN SOFTWARE ARCHITECTURE: A SYSTEMATIC LITERATURE REVIEW

U. Haider[1], E. Woods[2], R. Bashroush[1]
[1]University of East London, London, UK
[2]Endava, London, UK
Email: uhaider@clemson.edu

ABSTRACT

Variability in software-intensive systems is the ability of a software artefact (e.g., a system, subsystem, or component) to be extended, customised or configured for deployment in a specific context. Software Architecture is a high-level description of a software-intensive system that abstracts the system implementation details allowing the architect to view the system as a whole. Although variability in software architecture is recognised as a challenge in multiple domains, there has been no formal consensus on how variability should be captured or represented. The objective of this research was to provide a snapshot of the state-of-the-art on representing variability in software architecture while assessing the nature of the different approaches. To achieve this objective, a Systematic Literature Review (SLR) was conducted covering literature produced from January 1991 until June 2016. Then, grounded theory was used to conduct the analysis and draw conclusions from data, minimising threats to validity. In this paper, we report on the findings from the study.

Keywords: Variability, Software Architecture, Systematic Literature Review

1 INTRODUCTION

Over the past two decades, the field of Software Architecture has come a very long way, with increased interest in its application in various application domains. Systems are becoming more complex and larger in scale reinforcing the need for robust system architectures. The software architecture of a system is usually designed at the early stages of the system development lifecycle once initial requirements are understood. However, in some cases, as in the case of some legacy systems, the software architecture is defined implicitly and emerges during system development, rather than being developed explicitly.

The increase in variability in hardware, software, and operating environments, along with the strong business case for delaying design decisions as long as it is economically feasible, has required architectures to capture and cater for more complex variability in order to produce highly configurable and adaptable systems. It is common today to have systems encompassing thousands of interdependent variability points, making the process of modelling and maintaining these variability points a cumbersome process.

Variability covers several areas, from the software design and engineering process itself, to the output of such activities, including requirements, architecture, source code, test cases, binary code, configuration files, etc. (Svahnberg, van Gurp, & Bosch, 2005).

In this review, we focus solely on representing variability in software architecture design. Galster and Avgeriou (Galster & Avgeriou, 2011a) discussed that variability in software architecture can be a difficult activity to manage and comprehend. Thus, to address this challenge, one would need to understand the variety of challenges faced by practicing architects when dealing with variability management in practice (Galster & Avgeriou, 2011a).

Over the last 15 years, a lot of work has been reported that addresses the representation of variability in software architecture in different domains. Some approaches have defined variability in software architecture as a way of representing and reasoning about alternative system implementations (Bachmann & Bass, 2001; Galster & Avgeriou, 2011b). Similarly, a number of different mechanisms have been used to represent variability at the architecture level (e.g. Software Product Lines (SPL), Service-oriented architecture (SOA), etc.). Although it is generally agreed that variability representation is a key step of the development process, which can affect the success or failure of a system or a product line (Bashroush, 2010), there seems to be little consensus on how the representation is best conducted.

Very few Systematic Literature Reviews (SLR) considered variability beyond Software Product Lines. Galster et al. (Galster, Weyns, Tofan, Michalik, & Avgeriou, 2014) presented an SLR on variability in software systems in which they investigated variability handling in the various software engineering development phases (from requirements and design, to testing and maintenance). They found that most of the studies dealing with variability focused at the architecture and design phases; However, they did not analyse the modelling approaches or techniques covered within the studies.

In this paper, we present a systematically conducted literature review that capture and summarizes the state-of-the-art in representing variability in software architecture. The presentation of the work makes it accessible to practitioners working in the area who are looking to choose the best variability approach that fits their design needs, as well as researchers trying to identify areas that require further investigation

The rest of the paper is organized as follows. Section 2 describes the research methodology and lists the study research questions. Section 3 presents the data and meta-analysis of the results. Section 4 discusses and answers the study research questions. And, section 5 reports the study limitations and threats to validity. Finally, section 6 concludes the paper with final remarks.

2 RESEARCH METHODOLOGY

This section describes the research methodology adopted, namely a Systematic Literature Review (SLR), referred to as systematic review or review hereafter. "*A systematic review is a well-defined and methodical way to identify, evaluate, and synthesize the available evidence concerning a particular technology to understand the current direction and status of research or to provide background in order to identify research challenges*" (Kitchenham & Charters, 2007). This method was chosen because of the requirement to have a credible, repeatable and fair evaluation of the available studies on representing variability in software architectures.

The review starts by defining the research questions, followed by a definition of the search strategy process to be followed (sections 2.1 and 2.2). Then, inclusion and exclusion criteria are developed to provide a systematic way of selecting among identified primary studies (section 2.3). Finally, the data has been extracted from the primary studies to help answer the research questions (section 2.4). Once the data is

extracted, grounded theory is used to help analyse and draw conclusions to minimize threats to validity (discussed in section 5).

2.1 Research Questions

This research addresses concerns that relate to practitioners as well as researchers. As such, our review covers the following research questions:

RQ1: What approaches have been proposed to represent variability in software architecture?

RQ2: What is the context and areas of research of the studies employing variability in software architecture?

RQ3: What are the limitations of the existing approaches to represent variability in software architecture?

RQ1 is motivated by the need to describe the state-of –the art of how existing approaches represent variability. RQ2 helps better understand the applicability of each of the identified approaches, and to analyse any recurring patterns in different domain, while helping practitioners navigate through the reviewed approaches. We pose RQ3 to provide an overview of existing challenges in order to provide the directions for further research.

2.2 Search Strategy

The search string we have adopted to identify primary studies was designed following the criteria below:

- Extract terms from the topic being researched as well as research questions;
- Then, alternative terms (and synonyms) are included. This also covers terms that are spelt in different ways (e.g. British English vs American English);
- Based on the papers known to the researchers, related keywords are used to perform initial search in relevant repositories;
- Include other relevant terms where there is a possibility of identifying further material related to the topic.
- The identified words and keywords are then combined together in one string using instructs such as “OR” and “AND”;
- Finally, run the search strings and check relevance of returned results, then amend accordingly.

Following this strategy, the search string below was adopted:

<< (Variability OR Variabilities) AND (reference architecture OR software architecture OR architectural) >>

Seven digital repositories (1. IEEExplore; 2. ACM Digital library; 3. Citeseer; 4. SpringerLink; 5. Google Scholar; 6. ScienceDirect and 7. SCOPUS) were queried for primary studies. Validity was checked by looking for known papers the authors were already aware of. All papers checked were found in the identified primary studies.

Papers that we were not able to access online were acquired by contacting the relevant authors via email.
As an additional measure to ensure the comprehensiveness of the review, a manual check was conducted of the proceedings of the major conferences (such as ICSE, WICSA, ECSA and SPLC) and workshops (such as QoSA and VaMoS) that the researchers were aware of that published relevant papers.

The publication lists of known researchers publishing in the area were also checked manually. Finally, for the primary studies identified, forward and backward reference checking was conducted. For backward reference checking, we examined the reference list of the papers searching for any potential primary studies that had been missed. Similarly, for forward reference checking, we used search engines to identify citations to the primary studies that could be relevant to the review. This process helped to identify a number of additional potential primary studies. In terms of timeline, we searched for primary studies published between January 1991 and June 2016. The start date was set to be as early as possible (the earliest relevant primary studies identified were published in 2002). The search stage of this SLR was concluded in June 2016 (hence the end date), after that, the data extraction stage commenced.

2.3 Study Selection

The outcome from the different initial searches on digital libraries, manual searches, and known author searches, produced 1053 primary studies. After initial screening by the authors of this SLR based on title, abstract and keywords and excluding papers that were irrelevant or duplicates, 139 primary studies were selected. These remaining primary studies were subject to a more detailed review (of the full papers) where each paper was checked by three researchers. This process resulted in 30 papers being excluded. Of the remaining 109 primary studies, forward references (papers citing the primary study) and backward references (papers cited in the primary study) were followed which helped to identify a further 13 studies. The resulting 122 papers were then reviewed by applying the following inclusion and exclusion criteria:

- Inclusion criteria:
  - IC1: The primary study proposes or uses an approach to represent variability in software architecture;
  - IC2: In cases where the same content was published multiple times by the authors, the most recent and complete version was included as the primary study.

- Exclusion criteria:
  - EC1: The primary study addresses variability but not in software architecture domain.
  - EC2: The primary study is in the domain of software architecture, but does not consider variability. A paper that does not address variability along with software architecture has no value in answering our research questions.
  - EC3: Lack of sufficient details about representing variability in software architecture to make any useful contribution towards addressing research questions.
  - EC4: The primary study is a short (less than 3000 words) or symposium paper, opinion, abstract, tutorial summary, keynote, panel discussion, presentation slides, technical report, proceedings overview (for instance, from a conference,

workshop or special issue) or a book chapter. Only books and book chapters that were published as proceedings of peer-reviewed conferences were included (e.g. Springer LNCS or LNBIP series). These also had to be available through the corresponding digital libraries.

This led to the exclusion of 62 papers leaving us with 60 primary studies.

2.4 Data Extraction and Synthesis

On completion of the search and selection steps, data extraction was then conducted on the selected 60 primary studies to help answer the research questions defined in Section 2.1.

During data extraction, we captured information related to the paper synopsis, variability approach and the limitations. We made every effort to capture as much information as possible, but at the same time, kept the data as succinct as possible in order to avoid any potential influence of a taxonomic or classification framework on our results.

GoogleDocs was used to collect the extracted data from the different researchers and the aggregated results were made available in Excel spreadsheets for analysis. Finally, two researchers independently performed sanity checks on the results and the differences were reconciled collaboratively.

3 DATA AND ANALYSIS

Once the data extraction phase has been completed, data synthesis and analysis was conducted on the collected information. In this section, we provide an analysis of the primary studies in relation to their publication type, venues and trends.

Although we set our search period to start from January 1991 but unfortunately no studies were found in the 90s decade, the earliest primary studies identified were published in 2002. This could be due to the timing of the first major paper on the topic of Software Architecture by Shaw et al. (Shaw et al., 1995) in mid 90's. It is also worth mentioning here that no primary studies were identified in 2016. This is because 2016 is partially covered, when the search and selection process of this study was completed, and within that duration no such major conferences/workshops were conducted. Excluding VaMoS 2016 and WICSA 2016, which has been manually scanned for this review with no relevant studies.

Fig. 1 shows the number of primary studies identified, along with the breakdown of numbers of papers published via each publication outlet type (Conference, Journal or Workshop). The data presented shows papers bundled in 5 year brackets to smooth the effect of conference frequency (e.g. some conferences happen every 18 months, while others every 12 months) and public funding call trends (e.g. EU funded research projects addressing a specific challenge tend to start and end during the same time frame leading to increased paper publications in the area around the end of the funding period). Looking at the chart, it can be seen that there is an uptrend in research publications relating to variability in software architecture.

Additionally, it has been observed that the majority of the primary studies were published in the proceedings of conferences (72%, 43 papers), followed by Workshops (15%, 9 papers), and then Journals (13%, 8 papers).

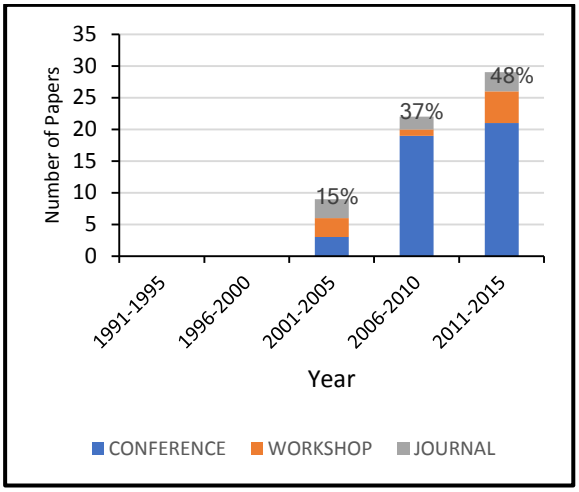

Figure 1: Publications per year

4 RESULTS AND DISCUSSION OF RESEARCH QUESTIONS

This section attempts to answer the study research questions by synthesizing and analysing the data extracted from the primary studies.

4.1 RQ1: What approaches have been proposed to represent variability in software architecture?

Two major approaches for representing variability in software architecture were identified in the primary studies: (1) defining variability using Unified Modelling Language (UML) or one of its extension in the form of another method, domain-ontology etc., and (2) using an ADL with explicit variability representation mechanisms. A detailed classification can be found in Table 1.

From the 60 selected primary studies, 48% (29 papers) of the primary studies presented various variability through UML, in which 23% (14 papers) used a form of meta-model based on UML class diagram. While 17% (10 papers) represented variability using other UML diagrams such as component diagram (e.g. (Bastarrica, Rivas, & Rossel, 2007) (Razavian & Khosravi, 2008) (Sánchez, Loughran, Fuentes, & Garcia, 2009) (Brito, Rubira, & de Lemos, 2009) (Losavio, Ordaz, Levy, & Baiotto, 2013)), activity diagram (e.g. (Abu-Matar & Gomaa, 2011), (Losavio et al., 2013)) and sequence diagram (e.g. (Laser, Rodrigues, Domingues, Oliveira, & Zorzo, 2015)). Finally, 8% (5 papers) extended the UML notation into UML PLUS (Product Line UML based Software Engineering) method ((Gomaa, 2013) (Albassam & Gomaa, 2013)); Kumbang ((Asikainen, Männistö, & Soininen, 2007), (Myllärniemi et al., 2012)), a modelling language and an ontology for modelling variability in software product line architectures from feature and component points of view; and KumbangSec ((Myllärniemi, Raatikainen, & Männistö, 2015)).

Table 1: Variability Representation Approaches

NOTATION	TOTAL PAPERS	PERCENTAGE	SOURCE
UML			
Class Diagram	14	23%	(Thiel & Hein, 2002b) (Moon, Chae, & Yeom, 2006) (Mikyeong, Heung Seok, Taewoo, & Keunhyuk, 2007) (Dobrica & Niemelä, 2008) (López, Casallas, & Hoek, 2009) (Helleboogh et al., 2009) (Pérez, Díaz, Costa-Soria, & Garbajosa, 2009) (Dai, 2009) (de Moraes et al., 2010) (Abu-Matar & Gomaa, 2011) (Tekinerdogan & Sözer, 2012) (Diaz, Perez, Fernandez-Sanchez, & Garbajosa, 2013) (Angelopoulos, Souza, & Mylopoulos, 2015) (Duran-Limon, Garcia-Rios, Castillo-Barrera, & Capilla, 2015)
Other (Component, Activity etc.)	10	17%	(Bastarrica et al., 2007) (Razavian & Khosravi, 2008) (Sánchez et al., 2009) (Helleboogh et al., 2009) (Brito et al., 2009) (de Moraes et al., 2010) (Kim, Lee, & Jang, 2011) (Losavio et al., 2013) (Laser et al., 2015) (Duran-Limon et al., 2015)
Extension (PLUS, Kumbang etc.)	5	8%	(Asikainen et al., 2007) (Myllärniemi et al., 2012) (Gomaa, 2013) (Albassam & Gomaa, 2013) (Myllärniemi et al., 2015)
	29	48%	
ADL	14	23%	(Hoek, 2004) (Zhang, Xiang, & Wang, 2005) (Bashroush, Brown, Spence, & Kilpatrick, 2005) (Satyananda, Danhyung, Sungwon, & Hashmi, 2007) (Bashroush et al., 2008) (Yu, Lapouchnian, Liaskos, Mylopoulos, & Leite, 2008) (Peng, Shen, & Zhao, 2009) (Coelho & Batista, 2011) (Haber, Rendel, Rumpe, & Schaefer, 2011) (Barbosa, Batista, Garcia, & Silva, 2011) (Haber, Rendel, Rumpe, Schaefer, & van der Linden, 2011) (Haber, Kutz, Rendel, Rumpe, & Schaefer, 2011) (Carvalho, Murta, & Loques, 2012) (Silva, Medeiros, Cavalcante, & Batista, 2013)
OVM	4	7%	(Razavian & Khosravi, 2008) (Helleboogh et al., 2009) (Groher & Weinreich, 2013) (Hwi, Sungwon, & Jihyun, 2013)
XML	2	3%	(de Moraes et al., 2010) (Myllärniemi et al., 2012)

Other (CVL, LISA etc.)	20	33%	(Thiel & Hein, 2002a) (Savolainen, Oliver, Mannion, & Hailang, 2005) (Ortiz, Pastor, Alonso, Losilla, & de Jódar, 2005) (Eklund, Askerdal, Granholm, Alminger, & Axelsson, 2005) (Andersson & Bosch, 2005) (Sinnema, Ven, & Deelstra, 2006) (Satyananda, Danhyung, & Sungwon, 2007) (Dhungana, Neumayer, Grünbacher, & Rabiser, 2008) (Kakarontzas, Stamelos, & Katsaros, 2008) (López et al., 2009) (Mann & Rock, 2009) (Brito et al., 2009) (Zhu et al., 2011) (Ahn & Kang, 2011) (Galster, Avgeriou, & Tofan, 2013) (Haber et al., 2013) (Pascual, Pinto, & Fuentes, 2013) (Groher & Weinreich, 2013) (Lytra et al., 2014) (Smiley, Mahate, & Wood, 2014)

23% (14 papers) of the selected primary studies described how to represent variability using an ADL, with a number of different ADLs adopted. The ADLs used for addressing variability were:

- ***xADL 2.0:*** (Hoek, 2004) uses xADL 2.0 together with several tools to express variability in xADL (MÉNAGE) and "*to select a particular system instance out of product line architecture (SELECTORDRIVER).*" (Peng et al., 2009) uses xADL 2.0 describing operators and process for merging reference architecture and application architecture. The result "*embodies all the application differences by new variation points, which makes it possible to synchronize application and component architectures.*"

- ***vADL:*** (Zhang et al., 2005) is an ADL that extends the framework of traditional ADL, and provides variability mechanisms, such as: Customized Interface, Variable Instance, Guard Condition, Variant Mapping, etc. vADL is able to describe the assembly of variability in product line architecture.

- ***ADLARS:*** (Bashroush et al., 2005) presents the ADL "ADLARS", a 3-view description of software architecture. This is an ADL with first class support for embedded systems product lines. It captures the relationship with explicit support for variability between the system's feature model and the architectural structures (using keywords like "supported", "unsupported" and "otherwise" in the description).

- ***ACME:*** (Satyananda, Danhyung, Sungwon, & Hashmi, 2007) describes two modelling notations, Forfamel for feature models and ACME (Garlan, Monroe, & Wile, 1997) for the architecture model. They are evaluated using the Formal Concept Analysis (FCA) technique, using a tool that generates a concept lattice graph that defines a mapping relationship between feature and architecture components.

- ***ALI:*** (Bashroush et al., 2008) presents an ADL called "ALI" (a descendent of "ADLARS" (Bashroush et al., 2005)) that aims to support product line engineering (and therefore also variability) as well as non-variant and individual system architectures.

- ***Darwin:*** (Yu et al., 2008) presents a framework with the Darwin ADL (with elements borrowed from one of its extensions, Koala (Ommering, van der Linden, Kramer, & Magee, 2000)). The paper proposes "*a decision-making process to generate a generic software design that can accommodate the full space of design alternatives from a goal model with high variability in configurations.*"

- ***MontiArc:*** an ADL designed to model architectures for asynchronously communicating logically distributed systems. Two studies present extension to MontiArc: (1) delta-modelling to represent variability - Δ-MontiArc in (Haber, Rendel, Rumpe, Schaefer, & van der Linden, 2011) and (Haber, Kutz, Rendel, Rumpe, & Schaefer, 2011), and (2) using hierarchical variability modelling - MontiArc^{HV} in (Haber et al., 2011). The given examples were difficult to extend if one is not using MontiArc, but the proposed variability modelling techniques were not new.

- ***PL-AspectualACME:*** (Barbosa et al., 2011) presents PL-AspectualACME (an extension to AspectualACME (Garcia et al., 2006)) with a graphical representation of the architectural model. The associated tool interprets the annotations, adding or removing the correct variant elements in the specification. (Coelho & Batista, 2011) presents the ADL PL-Aspectual ACME specifying the architecture for software product lines. The description is related to a goal model described in a formal visual notation PL-AOV Graph.

- ***CBabel:*** (Carvalho et al., 2012) presents the CBabel language, with features to support software architecture and contract description with a meta-model defined for architectural contracts.

- ***LightPL-ACME:*** (Silva et al., 2013) presents an ADL (an extension to ACME (Garlan et al., 1997)) with the aim of having "*a simple, lightweight language for SPL architecture description. It enables the association between the architectural specification and the artefacts involved in the SPL development process, including the relationship with the feature model by categorically defining the variability and the representation of both domain and application engineering elements.*"

Most of the work reported on the use of UML and ADLs for capturing variability at the architectural level was conducted by their original authors. A small proportion of these papers (e.g. (Sánchez et al., 2009) (Carvalho et al., 2012) (Albassam & Gomaa, 2013)) reported on work conducted in an industrial setting, but the rest used prototype implementations based in academia. We discuss the context of the research in more detail under RQ2 analysis.

OVM (Orthogonal Variability Model) and XML (EXtensible Markup Language) approaches represent variability in 7% (4 papers) and 3% (2 papers) of the selected primary studies respectively. Other ways that were identified to capture variability in the software architecture are: CVL (Common Variability Language) in (Pascual et al., 2013); LISA (Language for Integrated Software Architecture) in (Groher & Weinreich, 2013); formal modelling languages/framework (e.g. (Sinnema et al., 2006) (Satyananda, Danhyung, & Sungwon, 2007) (Hwi et al., 2013)) and modelling tools (e.g. (Dhungana et al., 2008) (Mann & Rock, 2009) (Lytra et al., 2014) (Smiley et al., 2014)), and; formal/informal textual and visual descriptions such as spreadsheets and process diagrams (e.g. (Thiel & Hein, 2002a) (Andersson & Bosch, 2005) (Kakarontzas et al., 2008; Zhu et al., 2011) (Galster et al., 2013) (Groher & Weinreich, 2013)).

It is important to state that the number of studies cross-cut multiple variability approaches, and accordingly, appear under more than one category in Table 1 (hence the total of 69 rather than 60). For instance, (Razavian & Khosravi, 2008) and (Helleboogh et al., 2009) covers UML and OVM; (Iris Groher & Rainer Weinreich, 2013) covers OVM and LISA; (de Moraes et al., 2010) and (Myllärniemi et al., 2012) covers UML and xml and variability mechanisms simultaneously. Also, (Helleboogh et al., 2009), (de Moraes et al., 2010) and (Duran-Limon et al., 2015) represent variability in both UML class and component diagrams.

Overall, UML and ADLs seemed to be the most commonly used approaches for capturing variability at an architectural level, making up 72% (43 papers) of the selected primary studies. UML was used in almost half of the studies, where it was extended through various mechanisms to support variability. While ADLs were mostly used in the product line domain.

4.2 RQ2: What is the context and areas of research of the studies employing variability in software architecture?

4.2.1 Research Context (Academia vs. Industry)

The research context of each primary study was classified as either: Academia (if the research was conducted in academia and by academics with no reference to industrial usage); Industry (if the research was conducted by industry based researchers or had direct industrial relevance); or both (when the research was a joint undertaking with both academic and industrial relevance). From the selected primary studies, we identified that only a small proportion of research (18%, 11 papers) was conducted in industry. 73% (44 papers) of the research surveyed was academic while 9% (5 papers) was classified as joint context (both industry and academia).

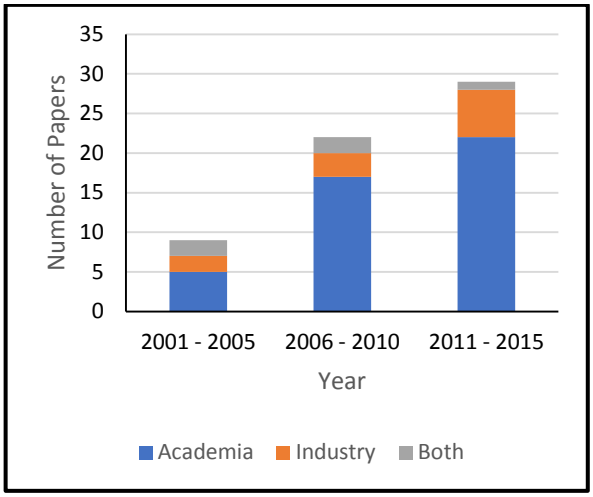

Figure 2: Research context

Fig. 2 shows that the majority of studies belong to the academia sector (73%), with 18% in industry and 9% joint. However, it was noticeable that the industry initiated papers doubled between 2011-2015 compared with 2006-2010, while academic papers only gone up by 9%. Yet, joint papers between industry and academia is going down with only 1 primary study published between 2011-2015.

4.2.2 Research Context (Theoretical vs. Practical)

Another way the research context of the primary studies was analysed was by checking whether the reported research had a practical or theoretical focus, or both. The results are reported in Fig. 3 which shows the majority of the work conducted is theoretical work with no direct application to practical problems.

Overall, 67% (40 papers) of the primary studies were focused purely on theoretical work with only 13% (8 papers) addressing practical issues and another 20% (12 papers) that can be classified as both.

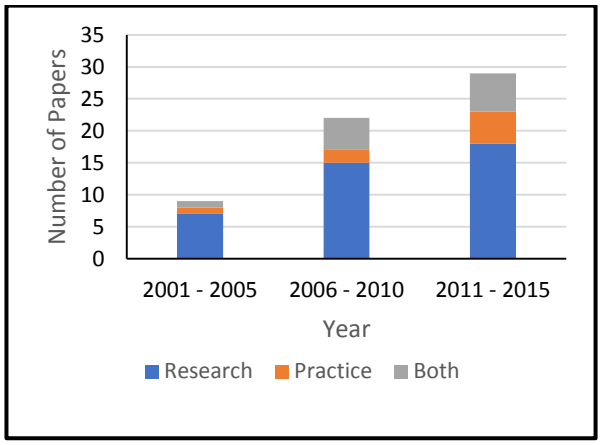

Figure 3: Research relevance

That said, Fig. 3 also shows that the trend is changing with higher percentage of papers with practical relevance appearing in the past 5 years compared to 2006-2010.

4.2.3 Research Areas

During the analysis, it became clear that the primary studies can be categorised under four main research areas:

1. Service Oriented Architecture (SOA)
2. Reference Architecture
3. Software Product lines (including Product Line Architectures -PLA and Dynamic SPL -DSPL)
4. Other (general Software Architecture)

The breakdown of primary studies per research area is shown in Table 2.

Fig. 4 shows a graphical distribution of the primary studies over the different areas identified. Noticeably, the work on variability in software architecture is dominated by work in the area of Software Product Lines.

Table 2: Breakdown of primary studies over research areas

RESEARCH AREA	TOTAL PAPERS	SOURCE
SOA	2	(Abu-Matar & Gomaa, 2011) (Galster et al., 2013)
Reference Architecture	4	(Ortiz et al., 2005) (Eklund et al., 2005) (Dobrica & Niemelä, 2008) (Galster et al., 2013)
Software Architecture (General)	11	(Sinnema et al., 2006) (Razavian & Khosravi, 2008) (Yu et al., 2008) (Pérez et al., 2009) (Dai, 2009) (Myllärniemi et al., 2012) (Tekinerdogan & Sözer, 2012) (Haber et al., 2013) (Gomaa, 2013) (Groher & Weinreich, 2013) (Angelopoulos et al., 2015)
SPL/PLA/DSPL	49	(Thiel & Hein, 2002b) (Thiel & Hein, 2002a) (Hoek, 2004) (Zhang et al., 2005) (Bashroush et al., 2005) (Savolainen et al., 2005) (Andersson & Bosch, 2005) (Moon et al., 2006) (Sinnema et al., 2006) (Bastarrica et al., 2007) (Asikainen et al., 2007) (Satyananda, Danhyung, Sungwon, et al., 2007) (Mikyeong et al., 2007) (Satyananda, Danhyung, & Sungwon, 2007) (Razavian & Khosravi, 2008) (Bashroush et al., 2008) (Dhungana et al., 2008) (Kakarontzas et al., 2008) (Sánchez et al., 2009) (Peng et al., 2009) (López et al., 2009) (Helleboogh et al., 2009) (Pérez et al., 2009) (Mann & Rock, 2009) (Brito et al., 2009) (de Moraes et al., 2010) (Kim et al., 2011) (Zhu et al., 2011) (Coelho & Batista, 2011) (Ahn & Kang, 2011) (Haber, Rendel, et al., 2011) (Barbosa et al., 2011) (Abu-Matar & Gomaa, 2011) (A. Haber et al., 2011) (Haber, Kutz, et al., 2011) (Carvalho et al., 2012) (Groher & Weinreich, 2013) (Pascual et al., 2013) (Gomaa, 2013) (Diaz et al., 2013) (Albassam & Gomaa, 2013) (Losavio et al., 2013) (Hwi et al., 2013) (Silva et al., 2013) (Lytra et al., 2014) (Smiley et al., 2014) (Myllärniemi et al., 2015) (Laser et al., 2015) (Duran-Limon et al., 2015)
	66[§]	

§ A number of studies cross-cut multiple research areas, and accordingly, appear under more than one research area (hence the total of 66 rather than 60)

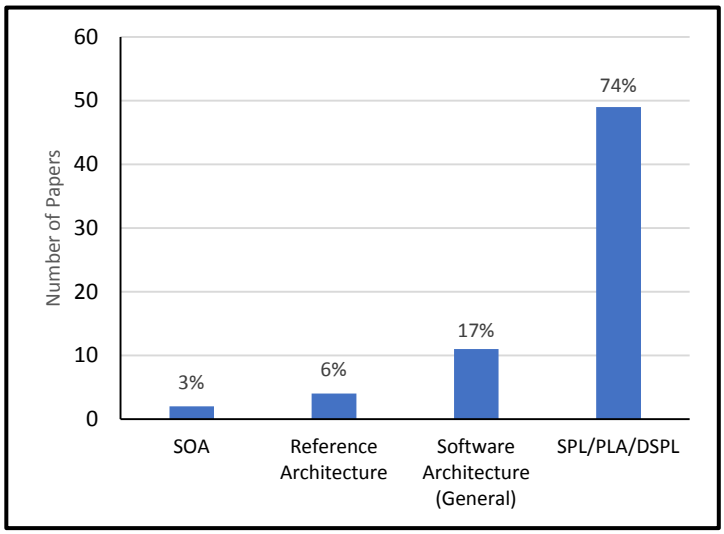

Figure 2: Breakdown of primary studies over research areas

4.3 RQ3: What are the limitation of the existing approaches to represent variability in software architecture?

Understanding the limitations of a particular piece of research is an important step towards understanding its applicability and utility. Unfortunately, in the literature reviewed for this study, 75% of the papers surveyed (45 of 60) did not make any attempt to report limitations of the research performed and 2% (1 study, (Eklund et al., 2005)) did not report the limitations of the research explicitly.

This left 14 studies (23%) that fully or partially identified limitations of their work, so helping to understand its maturity and the areas of its likely applicability. The limitations reported can be categorised under the following headers:

- -Technical limitations with the research methodology adopted: For example, some papers only used one case study ((Zhu et al., 2011), (Diaz et al., 2013)), while others used small unrepresentative study groups ((Andersson & Bosch, 2005)).
- -Technical limitations with the approach presented: For example, only addressing variability at either design time ((Ortiz et al., 2005), (Abu-Matar & Gomaa, 2011), (Angelopoulos et al., 2015) and (Duran-Limon et al., 2015)) or runtime ((Hoek, 2004), (Pérez et al., 2009)).
- -Both of the above (such as (Satyananda, Danhyung, & Sungwon, 2007), (López et al., 2009), (Brito et al., 2009), (Myllärniemi et al., 2012) and (Groher & Weinreich, 2013))

In reality, almost any piece of research is likely to embody some limitations, so it is surprising not to find all studies reporting limitations of either type.

5 THREATS TO VALIDITY AND STUDY LIMITATIONS

This section discusses the limitations and threats to validity of our study. As with most research methods, there are some inherent limitations to the SLR methodology. The first limitation is the possibility that the search and selection process may not have identified all of the relevant primary studies. This can be due to various reasons such as the use of

different terminology in primary studies to the one we adopted in the search term (particularly given that the work covered by this SLR cuts across multiple domains and research communities). To address this limitation, we have extended the search protocol and introduced a number of mitigating measures. First, we ran our automated searches on web sites of prominent publishers (e.g. IEEEXplore) as well as against general indexing search engines (e.g. Google Scholar) which helps to ensure comprehensiveness of results as different search engines use different ranking algorithms. Then, we conducted manual searches on proceedings of known publication outlets and publication lists of known authors in the domain and cross-examined the findings with the results produced from the automated search. Finally, we conducted forward and backward reference checks on the identified primary studies to further ensure that all of the relevant literature was identified.

Another limitation of SLRs is the exclusion of grey literature, such as thesis documents, white papers and technical reports. This could be a problem in some areas such as those where the work is led by industry, as practitioners tend to publish less in peer-reviewed outlets. However, looking at the analysis of RQ2, and to some extent at the initial results we had from the automated searches (conducted on general indexing websites such as Google Scholar), we notice that this study area is largely dominated by academic researchers with minimal potential for grey literature. Last but not least, there is the limitation of the language barrier where only primary studies published in English were searched and analysed. This could potentially mean that relevant primary studies published in other languages might have been missed. We do not have a strong mitigation to this threat other than noting that the majority of research in these areas appears to be published in English and so we do not believe that there is a high likelihood of significant research in this field remaining unpublished in English for long.

Given the end date of the search process of June 2016, few papers have been published since then. However, conducting a basic search just before publication, we managed to only identify one new primary study that could have been included in the SLR (Ali & Hong, 2017). Yet, this would not have impacted the findings and conclusions.

Beyond the inherent SLR methodology limitations, threats to validity can be classified under four main headers: construct, internal, external and conclusion (Matt & Cook, *1994*).

Some of the threats to *construct* and *internal* validity have already been discussed above. These threats arise from weaknesses in the execution of the research method adopted. A popular construct validity problem in SLRs is author bias and we have addressed this by having multiple authors review each primary study and had the overall process reviewed by an independent researcher (who was not one of the authors). This was discussed in the section on research methodology (Section 2).

On the other hand, the threat to *external* validity relates to the applicability of the results of the study beyond the context where it was conducted. Given that this study was not limited to one area, but studied multiple areas where variability in software architecture is used, *inductive generalization* is considerably strengthened. Moreover, we have made all of the raw data used for the study available for readers to better help them understand the reasoning and analysis conducted.

Finally, *conclusion* validity threats relate to the robustness of conclusions made based on the data available. A typical threat is when researchers gear conclusions to agree with their initial hypotheses. In our case, the research did not set any initial

hypotheses but rather addressed the research questions with an open view. Additionally, we based all our conclusions on grounded theory (Martin & Turner, 1986) and other analysis methods where multiple researchers were involved and independently agreed on the conclusions made.

6 CONCLUSIONS

This work aimed at capturing and cataloguing the state-of-the-art in representing variability in software architecture, making it more accessible to practitioners and researchers alike.

Overall, it can be said that this research area is witnessing an uptrend, especially since 2006 (see Fig. 1), and that work in this domain is starting to mature. In summary, we found that:

- UML (including various extensions) and Architecture Description Languages (ADL) were the most commonly used notations to represent variability in software architecture.
- The work on variability representation at the software architecture level can be largely mapped to three main research areas: Software Product Lines (SPL); Reference Architecture; and Service Oriented Architecture (SOA).
- Most of the work surveyed focused on proposing some form of new or improved design process or traceability technique relating to the development of systems that include variability.
- The majority of the work conducted (73%) was academically led, much of it with a fairly theoretical focus (67%).
- Overall, the research in this domain was found to have clear rationale and objectives, but generally lacking proper validation.

Future work should consider the creation of benchmarks to enable the comparison of the various techniques and their applicability to certain domains or problems.

REFERENCES

Abu-Matar, M., & Gomaa, H. (2011). *Variability Modeling for Service Oriented Product Line Architectures.* Paper presented at the Proceedings of the 15th International Software Product Line Conference (SPLC).

Ahn, H., & Kang, S. (2011). *Analysis of Software Product Line Architecture Representation Mechanisms.* Paper presented at the Proceedings of the Ninth International Conference on Software Engineering Research, Management and Applications, Baltimore, MD.

Albassam, E., & Gomaa, H. (2013). *Applying software product lines to multiplatform video games.* Paper presented at the Proceedings of the 3rd International Workshop on Games and Software Engineering (GAS).

Ali, N., & Hong, Jang-Eui (2017). *Creating adaptive software architecture dynamically for recurring new requirements.* Paper presented at the Proceedings of the International Conference on Open Source Systems and Technologies (ICOSST) doi:10.1109/ICOSST.2017.8279007

Andersson, J., & Bosch, J. (2005). Development and use of dynamic product-line architectures. *IEE Proceedings -Software, 152*(1), 15-28. doi:10.1049/ip-sen:20041007

Angelopoulos, K., Souza, V. E. S., & Mylopoulos, J. (2015, 2015//). *Capturing Variability in Adaptation Spaces: A Three-Peaks Approach.* Paper presented at the Proceedings of the 34th International Conference on Conceptual Modeling (ER 2015), Stockholm, Sweden.

Asikainen, T., Männistö, T., & Soininen, T. (2007). Kumbang: A domain ontology for modelling variability in software product families. *Adv. Eng. Inform., 21*(1), 23-40. doi:10.1016/j.aei.2006.11.007

Bachmann, F., & Bass, L. (2001). Managing variability in software architectures. *SIGSOFT Softw. Eng. Notes, 26*(3), 126-132. doi:10.1145/379377.375274

Barbosa, E. A., Batista, T., Garcia, A., & Silva, E. (2011). *PL-AspectualACME: an aspect-oriented architectural description language for software product lines.* Paper presented at the Proceedings of the 5th European conference on Software architecture (ECSA), Essen, Germany.

Bashroush, R. (2010). A NUI Based Multiple Perspective Variability Modeling CASE Tool. In M. A. Babar & I. Gorton (Eds.), *Software Architecture* (Vol. 6285, pp. 523-526): Springer Berlin Heidelberg.

Bashroush, R., Brown, T. J., Spence, I., & Kilpatrick, P. (2005). *ADLARS: An Architecture Description Language for Software Product Lines.* Paper presented at the Proceedings of the 29th Annual IEEE/NASA Software Engineering Workshop.

Bashroush, R., Spence, I., Kilpatrick, P., Brown, T. J., Gilani, W., & Fritzsche, M. (2008). *ALI: An Extensible Architecture Description Language for Industrial Applications.* Paper presented at the Proceedings of the 15th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems, ECBS.

Bastarrica, M. C., Rivas, S., & Rossel, P. O. (2007). *From a Single Product Architecture to a Product Line Architecture.* Paper presented at the Proceedings of the XXVI International Conference of the Chilean Society of Computer Science (SCCC).

Brito, P. H. S., Rubira, C. M. F., & de Lemos, R. (2009). *Verifying architectural variabilities in software fault tolerance techniques.* Paper presented at the Proceedings of the Joint Working IEEE/IFIP Conference on Software Architecture, 2009 & European Conference on Software Architecture. WICSA/ECSA 2009.

Carvalho, S. T., Murta, L., & Loques, O. (2012). *Variabilities as first-class elements in product line architectures of homecare systems.* Paper presented at the Proceedings of the 4th International Workshop on Software Engineering in Health Care (SEHC).

Coelho, K., & Batista, T. (2011). *From Requirements to Architecture for Software Product Lines.* Paper presented at the Proceedings of the 9th Working IEEE/IFIP Conference on Software Architecture (WICSA).

Dai, L. (2009). *Security Variability Design and Analysis in an Aspect Oriented Software Architecture.* Paper presented at the Proceedings of the Third IEEE International Conference on Secure Software Integration and Reliability Improvement.

de Moraes, A. L. S., de C Brito, R., Contieri, A. C., Ramos, M. C., Colanzi, T. E., de S Gimenes, I. M., & Masiero, P. C. (2010). *Using Aspects and the Spring Framework to Implement Variabilities in a Software Product Line.* Paper presented at the Proceedings of the XXIX International Conference of the Chilean Computer Science Society (SCCC).

Dhungana, D., Neumayer, T., Grünbacher, P., & Rabiser, R. (2008). *Supporting the Evolution of Product Line Architectures with Variability Model Fragments.* Paper presented at the Proceedings of the 7th Working IEEE/IFIP Conference on Software Architecture (WICSA).

Diaz, J., Perez, J., Fernandez-Sanchez, C., & Garbajosa, J. (2013). *Model-to-Code Transformation from Product-Line Architecture Models to AspectJ.* Paper presented at the Proceedings of the 39th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA).

Dobrica, L., & Niemelä, E. (2008). *An Approach to Reference Architecture Design for Different Domains of Embedded Systems.* Paper presented at the Proc. Software Engineering Research and Practice.

Duran-Limon, H. A., Garcia-Rios, C. A., Castillo-Barrera, F. E., & Capilla, R. (2015). An Ontology-Based Product Architecture Derivation Approach. *IEEE Transactions on Software Engineering, 41*(12), 1153-1168. doi:10.1109/TSE.2015.2449854

Eklund, U., Askerdal, Ö., Granholm, J., Alminger, A., & Axelsson, J. (2005). *Experience of introducing reference architectures in the development of automotive electronic systems.* Paper presented at the Proceedings of the second international workshop on Software engineering for automotive systems, St. Louis, Missouri.

Galster, M., & Avgeriou, P. (2011a). *Handling Variability in Software Architecture: Problems and Implications.* Paper presented at the Proceedings of the Ninth Working IEEE/IFIP Conference on Software Architecture.

Galster, M., & Avgeriou, P. (2011b). *The notion of variability in software architecture: results from a preliminary exploratory study.* Paper presented at the Proceedings of the 5th Workshop on Variability Modeling of Software-Intensive Systems, Namur, Belgium.

Galster, M., Avgeriou, P., & Tofan, D. (2013). Constraints for the design of variability-intensive service-oriented reference architectures – An industrial case study. *Information and Software Technology, 55*(2), 428-441.

Galster, M., Weyns, D., Tofan, D., Michalik, B., & Avgeriou, P. (2014). Variability in Software Systems - A Systematic Literature Review. *IEEE Transaction Software Engineering, 40*(3), 282-306.

Garcia, A., Chavez, C., Batista, T., Sant'anna, C., Kulesza, U., Rashid, A., & Lucena, C. (2006). *On the Modular Representation of Architectural Aspects.* Paper presented at the Proceedings of the Third European Workshop on Software Architecture, EWSA.

Garlan, D., Monroe, R., & Wile, D. (1997). *Acme: an architecture description interchange language.* Paper presented at the Proceedings of the Centre for Advanced Studies on Collaborative research, CASCON'.

Gomaa, H. (2013). *Evolving software requirements and architectures using software product line concepts.* Paper presented at the Proceedings of the 2nd International Workshop on the Twin Peaks of Requirements and Architecture (TwinPeaks).

Groher, I., & Weinreich, R. (2013). *Strategies for Aligning Variability Model and Architecture.* Paper presented at the Proceedings of the 20th Asia-Pacific Software Engineering Conference (APSEC).

Groher, I., & Weinreich, R. (2013). *Supporting Variability Management in Architecture Design and Implementation.* Paper presented at the Proceedings of the 46th Hawaii International Conference on System Sciences (HICSS).

Haber, A., Hölldobler, K., Kolassa, C., Look, M., Müller, K., Rumpe, B., & Schaefer, I. (2013). *Engineering Delta Modelling Languages*. Paper presented at the Proceedings of the 17th International Software Product Line Conference (SPLC), Tokyo, Japan.

Haber, A., Kutz, T., Rendel, H., Rumpe, B., & Schaefer, I. (2011). *Delta-oriented architectural variability using MontiCore.* Paper presented at the Proceedings of the 5th European Conference on Software Architecture (ECSA), Essen, Germany.

Haber, A., Rendel, H., Rumpe, B., & Schaefer, I. (2011). *Delta Modeling for Software Architectures.* Paper presented at the Proceedings of the Dagstuhl Workshop on Model-Based Development of Embedded Systems (MBEES), Germany.

Haber, A., Rendel, H., Rumpe, B., Schaefer, I., & van der Linden, F. (2011). *Hierarchical Variability Modeling for Software Architectures.* Paper presented at the Proceedings of the 15th International Software Product Line Conference (SPLC).

Helleboogh, A., Weyns, D., Schmid, K., Holvoet, T., Schelfthout, K., & Van Betsbrugge, W. (2009). *Adding variants on-the-fly: Modeling meta-variability in dynamic software product lines.* Paper presented at the Proceedings of the Third International Workshop on Dynamic Software Product Lines (DSPL {@} SPLC 2009), Pittsburgh, PA, USA.

Hoek, A. v. d. (2004). Design-time product line architectures for any-time variability. *Sci. Comput. Program, 53*(3), 285-304. doi:10.1016/j.scico.2003.04.003

Hwi, A., Sungwon, K., & Jihyun, L. (2013). *A Case Study Comparison of Variability Representation Mechanisms with the HeRA Product Line.* Paper presented at the Proceedings of the IEEE 16th International Conference on Computational Science and Engineering (CSE).

Kakarontzas, G., Stamelos, I., & Katsaros, P. (2008). *Product Line Variability with Elastic Components and Test-Driven Development.* Paper presented at the Proceedings of the International Conference on Computational Intelligence for Modelling Control & Automation.

Kim, Y.-G., Lee, S. K., & Jang, S.-B. (2011). Variability Management for Software Product-line Architecture Development. *International Journal of Software Engineering and Knowledge Engineering, 21*(07), 931-956. doi:doi:10.1142/S0218194011005542

Kitchenham, B., & Charters, S. (2007). Guidelines for performing systematic literature reviews in software engineering (version 2.3). In. Technical report, Keele University and University of Durham,.

Laser, M. S., Rodrigues, E. M., Domingues, A., Oliveira, F., & Zorzo, A. F. (2015). *Architectural Evolution of a Software Product Line: an experience report.* Paper presented at the Proceedings of the 27th International Conference on Software Engineering and Knowledge Engineering (SEKE), Pittsburgh, USA.

Losavio, F., Ordaz, O., Levy, N., & Baiotto, A. (2013). *Graph modelling of a refactoring process for Product Line Architecture design.* Paper presented at the Proceedings of the XXXIX Latin American Computing Conference (CLEI).

Lytra, I., Eichelberger, H., Tran, H., Leyh, G., Schmid, K., & Zdun, U. (2014). *On the Interdependence and Integration of Variability and Architectural Decisions.* Paper presented at the Proceedings of the Eighth International Workshop on Variability Modelling of Software-Intensive Systems (VaMoS), Sophia Antipolis, France.

López, N., Casallas, R., & Hoek, A. v. d. (2009). *Issues in mapping change-based product line architectures to configuration management systems*. Paper presented at the Proceedings of the 13th International Software Product Line Conference, San Francisco, California.

Mann, S., & Rock, G. (2009). *Dealing with Variability in ArchitectureDescriptions to Support Automotive Product Lines: Specification and Analysis Methods.* Paper presented at the Proceedings of the Embedded World Conference 2009, Nurnberg, Deutschland.

Martin, P. Y., & Turner, B. A. (1986). Grounded Theory and Organizational Research. *The Journal of Applied Behavioral Science, 22*(2), 141-157. doi:10.1177/002188638602200207

Matt, G. E., & D.Cook, T. (*1994*). Threats to the validity of research synthesis. In H. Cooper & L. V. Hedges (Eds.), *In the Handbook of Research Synthesis* (pp. 503-520). New York: Russell Sage Foundation.

Mikyeong, M., Heung Seok, C., Taewoo, N., & Keunhyuk, Y. (2007). *A Metamodeling Approach to Tracing Variability between Requirements and Architecture in Software Product Lines.* Paper presented at the Proceedings of the 7th IEEE International Conference on Computer and Information Technology (CIT).

Moon, M., Chae, H. S., & Yeom, K. (2006). *A metamodel approach to architecture variability in a product line.* Paper presented at the Proceedings of the 9th international conference on Reuse of Off-the-Shelf Components, Turin, Italy.

Myllärniemi, V., Raatikainen, M., & Männistö, T. (2015). *Representing and Configuring Security Variability in Software Product Lines.* Paper presented at the Proceedings of the 11th International ACM SIGSOFT Conference on Quality of Software Architectures (QoSA), Montréal, QC, Canada.

Myllärniemi, V., Ylikangas, M., Raatikainen, M., Pääkkö, J., Männistö, T., & Aaltonen, T. (2012). *Configurator-as-a-service: tool support for deriving software architectures at runtime.* Paper presented at the Proceedings of the WICSA/ECSA 2012 Companion Volume, Helsinki, Finland.

Ommering, R. v., van der Linden, F., Kramer, J., & Magee, J. (2000). The Koala Component Model for Consumer Electronics Software. *IEEE Computer, 33*, 78-85. doi:10.1109/2.825699

Ortiz, F. J., Pastor, J. A., Alonso, D., Losilla, F., & de Jódar, E. (2005). *A reference architecture for managing variability among teleoperated service robots.* Paper presented at the Proceedings of the 2nd International Conference on Informatics in Control Automation and Robotics (ICINCO).

Pascual, G. G., Pinto, M., & Fuentes, L. (2013). *Run-Time support to manage architectural variability specified with CVL.* Paper presented at the Proceedings of the 7th European conference on Software Architecture (ECSA), Montpellier, France.

Peng, X., Shen, L., & Zhao, W. (2009). *An Architecture-based Evolution Management Method for Software Product Line.* Paper presented at the Proceedings of the International Conference on Software Engineering and Knowledge Engineering (SEKE).

Pérez, J., Díaz, J., Costa-Soria, C., & Garbajosa, J. (2009). *Plastic Partial Components: A solution to support variability in architectural components.* Paper presented at the Proceedings of the Joint Working IEEE/IFIP Conference on Software Architecture & European Conference on Software Architecture. WICSA/ECSA 2009.

Razavian, M., & Khosravi, R. (2008). *Modeling variability in the component and connector view of architecture using UML.* Paper presented at the Proceedings of the 2008 IEEE/ACS International Conference on Computer Systems and Applications.

Satyananda, T. K., Danhyung, L., & Sungwon, K. (2007). *Formal Verification of Consistency between Feature Model and Software Architecture in Software Product Line.* Paper presented at the Proceedings of the International Conference on Software Engineering Advances (ICSEA).

Satyananda, T. K., Danhyung, L., Sungwon, K., & Hashmi, S. I. (2007). *Identifying Traceability between Feature Model and Software Architecture in Software Product Line using Formal Concept Analysis.* Paper presented at the Proceedings of the International Conference on Computational Science and its Applications (ICCSA).

Savolainen, J., Oliver, I., Mannion, M., & Hailang, Z. (2005). *Transitioning from product line requirements to product line architecture.* Paper presented at the Proceedings of the 29th Annual International Computer Software and Applications Conference (COMPSAC).

Shaw, M., DeLine, R., Klein, D. V., Ross, T. L., Young, D. M., & Zelesnik, G. (1995). Abstractions for software architecture and tools to support them. *IEEE Transactions on Software Engineering,, 21*(4), 314-335. doi:10.1109/32.385970

Silva, E., Medeiros, A. L., Cavalcante, E., & Batista, T. V. (2013). *A Lightweight Language for Software Product Lines Architecture Description.* Paper presented at the Proceedings of the 7th European Conference on Software Architecture, ECSA, Montpellier, France.

Sinnema, M., Ven, J. S. v. d., & Deelstra, S. (2006). Using variability modeling principles to capture architectural knowledge. *SIGSOFT Softw. Eng. Notes, 31*(5), 5. doi:10.1145/1163514.1178645

Smiley, K., Mahate, S., & Wood, P. (2014). *A Dynamic Software Product Line Architecture for Prepackaged Expert Analytics: Enabling Efficient Capture, Reuse and Adaptation of Operational Knowledge.* Paper presented at the Proceedings of the Working IEEE/IFIP Conference on Software Architecture (WICSA).

Svahnberg, M., van Gurp, J., & Bosch, J. (2005). A taxonomy of variability realization techniques: Research Articles. *Software—Practice & Experience, 35*(8), 705-754.

Sánchez, P., Loughran, N., Fuentes, L., & Garcia, A. (2009). *Engineering Languages for Specifying Product-Derivation Processes in Software Product Lines.* Paper presented at the Proceedings of the International Conference on Software Language Engineering.

Tekinerdogan, B., & Sözer, H. (2012). *Variability viewpoint for introducing variability in software architecture viewpoints.* Paper presented at the Proceedings of the WICSA/ECSA 2012 Companion Volume, Helsinki, Finland.

Thiel, S., & Hein, A. (2002a). Modeling and Using Product Line Variability in Automotive Systems. *IEEE Softw., 19*(4), 66-72. doi:10.1109/ms.2002.1020289

Thiel, S., & Hein, A. (2002b). *Systematic Integration of Variability into Product Line Architecture Design.* Paper presented at the Proceedings of the Second International Conference on Software Product Lines.

Yu, Y., Lapouchnian, A., Liaskos, S., Mylopoulos, J., & Leite, J. C. S. P. (2008). *From goals to high-variability software design.* Paper presented at the Proceedings of the 17th international conference on Foundations of intelligent systems, Toronto, Canada.

Zhang, T., Xiang, D., & Wang, H. (2005). *vADL: A Variability-Supported Architecture Description Language for Specifying Product Line Architectures.* Paper presented at the Proceedings of the Second International Software Product Lines Young Researchers Workshop (SPLYR) in conjunction with the 9th International Software Product Line conference (SPLC), Rennes, France.

Zhu, J., Peng, X., Jarzabek, S., Xing, Z., Xue, Y., & Zhao, W. (2011). *Improving product line architecture design and customization by raising the level of variability modeling.* Paper presented at the Proceedings of the 12th international conference on Top productivity through software reuse, Pohang, South Korea.