The Journal of Systems & Software 144 (2018) 511–532

Contents lists available at ScienceDirect

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss

A systematic literature review on the semi-automatic configuration of extended product lines

Lina Ochoa[a,c], Oscar González-Rojas[*,a], Alves Pereira Juliana[b], Harold Castro[a], Gunter Saake[b]

[a] Universidad de los Andes, Systems and Computing Engineering Department, School of Engineering, Bogotá, Colombia
[b] University of Magdeburg, Department of Technical and Business Information Systems, School of Computer Science, Magdeburg, Germany
[c] Centrum Wiskunde & Informatica, Software Analysis & Transformation Group, Amsterdam, The Netherlands

ARTICLE INFO

Keywords:
Extended product line
Product configuration
Systematic literature review

ABSTRACT

Product line engineering has become essential in mass customisation given its ability to reduce production costs and time to market, and to improve product quality and customer satisfaction. In product line literature, mass customisation is known as product configuration. Currently, there are multiple heterogeneous contributions in the product line configuration domain. However, a secondary study that shows an overview of the progress, trends, and gaps faced by researchers in this domain is still missing. In this context, we provide a comprehensive systematic literature review to discover which approaches exist to support the configuration process of extended product lines and how these approaches perform in practice. Extend product lines consider non-functional properties in the product line modelling. We compare and classify a total of 66 primary studies from 2000 to 2016. Mainly, we give an in-depth view of techniques used by each work, how these techniques are evaluated and their main shortcomings. As main results, our review identified (i) the need to improve the quality of the evaluation of existing approaches, (ii) a lack of hybrid solutions to support multiple configuration constraints, and (iii) a need to improve scalability and performance conditions.

1. Introduction

Product line engineering (PLE) was born as a new paradigm that supports mass customisation (Pohl et al., 2005), given the need to respond to the demand of customised products. This paradigm offers a set of benefits, such as reduction of production costs and time-to-market, as well as the improvement of products quality and the increase of customers satisfaction by responding to individual stakeholders' requirements (Clements and Northrop, 2001; Pohl et al., 2005). As defined by Pohl et al. (2005), PLE is composed by two main processes, *domain engineering* and *application engineering*. In the *domain engineering* process, a complete *Product Line* (PL) or family of products is defined, including the *variability* of the family that captures all common and variable features in the PL. An extended PL contains feature attributes that represent non-functional properties of the product family, which improve domain expressiveness and scope stakeholders' optimization requirements. Nevertheless, these attributes also add computational complexity to configure user-specific products. In the *application engineering* process, a set of products are derived from the defined PL. One of its main sub-processes correspond to *product configuration*, where a set of features are selected from the PL variability space. However, the manual selection of features from a PL begins to be an error-prone and time-consuming task when the variability space is too large. This happens because decision makers should consider both variability constraints (*e.g.* if a feature *A* is selected then feature *B* should also be considered) and business needs (*e.g.* the cheapest product should be selected) (Asadi et al., 2014; White et al., 2008).

To overcome the manual configuration challenges, in the last decade, researchers in the PL field have proposed semi-automatic approaches to support the product configuration optimization in large-scale variability spaces. However, a secondary study is needed in order to present a complete overview of the current state-of-the-art to guide future research in this domain. Thus, we performed a *Systematic Literature Review* (SLR) according to the guidelines proposed by Kitchenham et al. (2009, 2015). This work is presented as an extension of the survey presented by Ochoa et al. (2017). We assessed 66 articles, in order to answer three main research questions (*cf.* Section 3.1):

RQ1. Which techniques are employed to support the semi-automatic configuration of extended PLs?

* Corresponding author.
E-mail addresses: lm.ochoa750@uniandes.edu.co (L. Ochoa), o-gonza1@uniandes.edu.co (O. González-Rojas), juliana.alves-pereira@ovgu.de (A.P. Juliana), hcastro@uniandes.edu.co (H. Castro), gunter.saake@ovgu.de (G. Saake).

https://doi.org/10.1016/j.jss.2018.07.054
Received 22 April 2017; Received in revised form 20 June 2018; Accepted 17 July 2018
Available online 19 July 2018

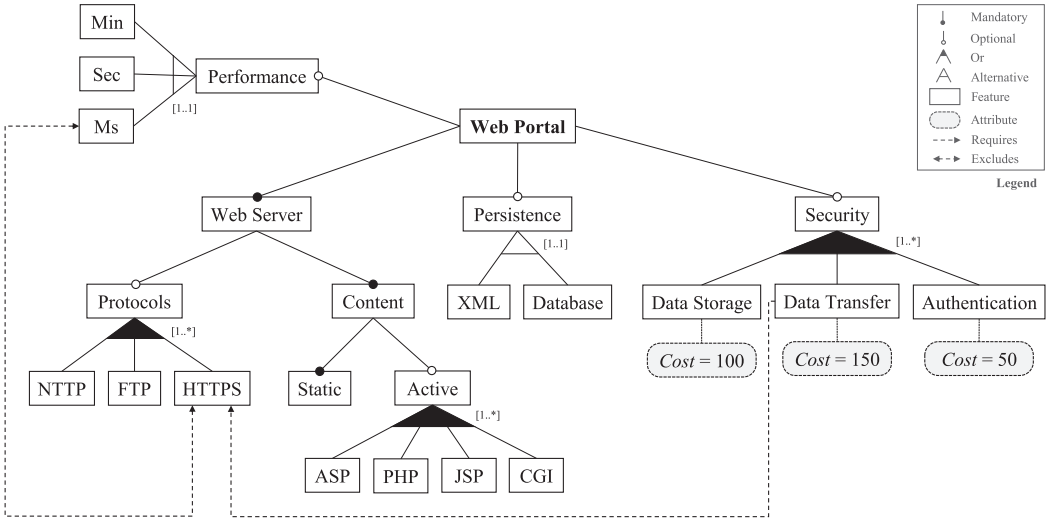


Fig. 1. *Web Portal* FM (Mendonça et al., 2008) extended with the *cost* feature attribute.

RQ2. How are the proposed approaches evaluated?
RQ3. What are the open challenges faced by the current approaches?

The remainder of this paper is structured as follows. Section 2 presents the main terms and concepts used along this article and a discussion of related work. Section 3 describes the research methodology used to conduct this SLR. Section 4, Section 5, and Section 6 address the research questions *RQ1, RQ2*, and *RQ3*, respectively. Finally, Section 7 discusses potential threats to validity, and Section 8 presents the conclusions of the SLR and directions for future work.

2. Background and motivation

2.1. Core concepts in the configuration of extended product lines

A *product line(PL)* represents a product family with a set of common and variable components that are represented within a variability model (*cf.* domain engineering process Pohl et al., 2005). An *extended Product Line* (extended PL) is a PL that contains attributes attached to its features. *Feature Models* (FMs) (Kang et al., 1990) have become the *de facto* variability modelling approach for representing PLs. A variability model is then used to derive products according to individual customer requirements (*cf.* application engineering process Pohl et al., 2005). To illustrate these concepts, Fig. 1 presents a modified and extended version of the *Web Portal* PL (Mendonça et al., 2008) taken from the SPLOT[1] model repository.

Variability modelling. A *Feature Diagram* (FD) is an acyclic undirected graph $FD = (r, F, TC)$, where r is the root of the FD (*i.e.* Web Portal in Fig. 1); F is the set of nodes representing *features*, which denote functional requirements relevant to a group of stakeholders (Czarnecki and Eisenecker, 2000) (*e.g.* Web Server, Data Storage in Fig. 1); and TC is the set of edges representing *tree constraints*, which denote relations among features (Czarnecki and Eisenecker, 2000; Kang et al., 1990). A *mandatory* TC specifies that the selection of a child feature is mandatory if the parent feature is selected. For example, the *Web Server* feature must be selected for all products derived from the FM. An *optional* TC specifies that the selection of a child feature is optional if the parent feature is selected. For instance, the feature *Active* can be selected or deselected when the feature *Content* is selected. An *or-group* TC states that if the parent is selected *at least* one of the child features should be selected. For example, if the feature *Protocols* is selected, at least one of the three child features (*NFTP, FTP, HTTPS*) must be included. An *alternative-group* TC states that if the parent is selected *at most* one of the child features should be selected. For example, if the feature *Persistence* is selected, just one of the two child features (*XML* and *Database*) must be included.

In addition, a FM can have *Cross-Tree Constraints* (CTC) that represent *requires* or *excludes* implications among unconnected features. For instance, if *Data Transfer* is selected in a product, then *HTTPS* must also be selected; and if *HTTPS* is selected, then *Ms* must be deselected, and vice versa. TC and CTC are known as the FM *integrity constraints*. Lastly, *Extended Feature Models* (EFM) represent non-functional properties (*i.e.* attributes) related to features (Benavides et al., 2010). For example, the *Data Storage, Data Transfer*, and *Authentication* features have the *cost* attribute represented by an integer value.

Product configuration. This process allows the configuration of the PL variability into a single product by selecting features from the variability model. Product configuration can be defined as $C = (S, D)$ where S is the set of selected features, D is the set of deselected features, $S, D \subseteq F$, and $S \cap D = \varnothing$ (Benavides et al., 2010). If $S \cup D = F$, then the configuration is *complete*, otherwise it is known as a *partial* configuration. In both cases, a *configuration* is valid if it satisfies all *integrity constraints*. Complete and valid configurations are known as *products* (Ochoa et al., 2015). For example, the *Web Portal* FM in Fig. 1 represents 15.360 different products. One of those products correspond to $C_1 = (S_1, D_1)$, where S_1={*Web Portal*, *Web Server*, *Protocols*, *HTTPS*, *Content*, *Static*, *Security*, *Data Transfer*, *Authentication*, *Performance*, *Min*} and D_1={*NTTP*, *FTP*, *Active*, *ASP*, *PHP*, *JSP*, *CGI*, *Persistence*, *XML*, *Database*, *Data Storage*, *Sec*, *Ms*}.

Usually, product configuration is manually performed; decision-makers select the desired features in order to fulfil a set of particular product requirements. However, when the variability model grows and allows the configuration of hundreds of possible products, this manual task becomes error-prone and time-consuming (Asadi et al., 2014; Henard et al., 2015; Mazo et al., 2012; Siegmund et al., 2008; White et al., 2008). In a *semi-automatic configuration*, a software system considers a set of functional and non-functional requirements and generates a set of complete configurations.

A semi-automatic product configuration in extended PLs requires features auto-completion based on *configuration constraints*; which are product restrictions or requirements (*e.g.* single or multiple optimization objective, hard limits) defined by one or multiple stakeholders

[1] Found at http://www.splot-research.org/.

Table 1
PL-related secondary studies.

SLR	Year	Articles
Chen et al. (2009)	2009	34
Alves et al. (2010)	2010	49
Benavides et al. (2010)	2010	53
Rabiser et al. (2010)	2010	118
Chen and Babar (2011)	2011	97
Engström and Runeson (2011)	2011	64
da Mota Silveira Neto et al. (2011)	2011	45
da Silva et al. (2013)	2013	20
Laguna and Crespo (2013)	2013	74
Galster et al. (2014)	2014	196
Soares et al. (2014)	2014	36
Afzal et al. (2016)	2016	19
Bashroush et al. (2017)	2017	37

during the PL configuration process (see Section 4.2). These constraints are also known in the literature as *solution constraints, user preferences*, and *product requirements*, among others. For instance, an optimization process may find which configurations derive the minor cost, the major performance, or the best cost-performance relationship for the PL.

In this paper, we aim at giving an in-depth view of optimization techniques used by current works (*e.g. constraint programming, evolutionary algorithms, fuzzy logic*) to assist the configuration of extended PLs (*cf.* Section 4). This study is motivated for the appearance of real-world variability intensive systems that are coping with extended information such as quality attributes represented as non-boolean variables.

2.2. Related work

We identified a set of secondary studies that address sub-problems related to the PLs field. However, none of them addresses the particular issue concerning the semi-automatic configuration of PLs. All considered studies with their corresponding publication year and number of analysed articles are listed in Table 1.

On the one hand, some studies address the problem of variability management in Software PL (SPL) artefacts like requirements, design, architecture, components, and tests. Chen et al. (2009) studied the evolution on variability management approaches, and issues that drove the referred evolution. These approaches are then evaluated in the study presented by Chen and Babar (2011). Evaluation methods (*e.g.* case study), environments (*e.g.* industrial), quality, and studied objects are the main factors of the analysis. Similarly, Galster et al. (2014) tackle the variability handling problem in software engineering. Variability dimensions in the studied articles are identified and defined. Likewise,Afzal et al. (2016) identify articles where artificial intelligence techniques are used to support different phases during the PL variability management (*e.g.* inconsistencies identification, debugging, product configuration). Lastly, Soares et al. (2014) investigate how different approaches analyse non-functional properties related to SPLs during runtime.

On the other hand, the studies presented by Engström and Runeson (2011), and da Mota Silveira Neto et al. (2011) analyse approaches that test SPLs. In the case of Engström and Runeson (2011), there is a particular interest on identifying if the analysed articles contribute to test framework (*i.e.* test organization and process), test management (*i.e.* test planning and assessment), testability (*i.e.* improving testing), system and acceptance testing, integration testing, unit testing, or automation tasks. da Mota Silveira Neto et al. (2011) are interested in exploring the approaches testing strategy (*e.g.* product by product testing, SPL incremental testing), the related static and dynamic analysis (*e.g.* formal verification), testing levels (*e.g.* unit and integration testing), among other aspects. Although, some of the studied approaches consider the selection of a set of products to improve testing efficiency and effectiveness, the selection of a product to satisfy customer needs is not considered as part of their research.

Similarly, Alves et al. (2010) focus on the field of Requirements Engineering (RE) applied to SPLs. They have a particular interest on identifying artefact formats (*e.g.* features), activities employed during the RE process (*e.g.* planning), SPL adoption strategies (*i.e.* proactive, extractive, reactive), and adoption evidence (*e.g.* industrial practice). Likewise, (Laguna and Crespo, 2013) address a particular problem of RE, the SPL evolution. The refactoring of existing SPLs and the re-engineering of legacy systems into SPLs are the two main topics explored by this secondary study. SPLs are considered from three perspectives, model evolution, code evolution, and tool support.

Two additional studies performed reviews related to product derivation. For instance, Bashroush et al. (2017); Rabiser et al. (2010) identified a set of requirements for tool-support during product derivation (*e.g.* automated and interactive variability resolution, variability visualisation); da Silva et al. (2013) reviewed articles related to dynamic SPLs. In this research they claim that features and reconfiguration plans are the main input during dynamic product derivation.

Finally, the SLR presented by Benavides et al. (2010) is more likely to address problems which are partially related to the PL configuration concern. They aim at exploring available variability model analysis (*e.g.* core features, void FM), employed techniques (*e.g.* BDD, SAT, CP), and a brief performance evaluation when executing the identified operations. Even though this paper does not directly tackle the product configuration problem in extended PLs, it motivates our work and is cited in our SLR.

In conclusion, none of the aforementioned surveys directly address the problem of extended PLs configuration. SPLs testing, variability management, requirements engineering, evolution, and variability model analysis are addressed as main concerns of the secondary studies, but the configuration problem is sometimes not even mentioned, nor studied in isolation from other variability management phases. Consequently, the current SLR provides further details on the problem of product configuration, such as information related to configuration constraints, techniques employed for solving the extended PL configuration problem, and performance and scalability results of existing approaches addressing this issue in extended variability models. This information is important to assist future researchers and industrial practitioners, when creating tools for supporting product configuration in large and extended PLs based on a set of requirements.

3. The review methodology

This paper presents a *Systematic literature review* (SLR) that follows the guidelines proposed by Kitchenham et al. (2009, 2015). The objective of this SLR is to analyse and classify relevant information related to approaches framed in the context of the semi-automatic configuration of extended PLs. In this section, we present the SLR methodology that includes the definition of the research questions, the search process, the SLR scope defined by inclusion and exclusion criteria, the way on which data was collected, and the corresponding search results. Main activities related to the SLR process and their location in the article are shown in Fig. 2.

3.1. Research questions

We defined three main research questions in order to clarify the state-of-the-art related to the semi-automatic configuration of extended PLs, namely:

RQ1. Which techniques are employed to support the semi-automatic configuration of extended PLs?
RQ2. How are the proposed approaches evaluated?
RQ3. What are the open challenges faced by the current approaches?

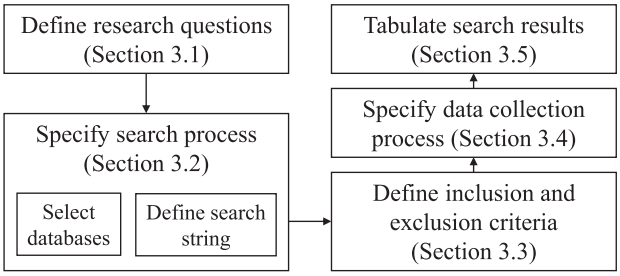


Fig. 2. SLR process.

These questions tackle three concerns orthogonal to all approaches: employed techniques, conducted evaluation process, and open challenges. We systematically identify the techniques employed by studied approaches to configure extended PLs (*cf.* RQ1), such as constraint programming and evolutionary algorithms (*cf.* Section 4). We analyse how the evaluation process was conducted by each study (*cf.* RQ2), such as industrial and academic case studies (*cf.* Section 5). Lastly, we identify main gaps to guide future research in this domain (*cf.* RQ3) by presenting the concerns that have not been addressed by the current studies (*cf.* Section 6).

3.2. Search process

To find and select relevant approaches for the SLR, we searched in five databases known as having relevant research in the field of computer science and engineering (Brereton et al., 2007), namely *ACM Digital Library, Science Direct, IEEE Xplore, Scopus*, and *Springer Link*. We define the following *search string* to obtain a set of articles in all the aforementioned databases.

(product line *OR* product family *OR* system family)
AND
(configuration *OR* product selection *OR* feature selection)
AND
(attribute OR non-functional *OR* quality *OR* preference *OR* requirement)

First, we specify different synonyms related to the term *product line*, mainly *product family* and *system family*. Then, we refer to the PL configuration process. The selected synonyms correspond to *configuration, product selection*, and *feature selection*. This is particularly used in approaches that employ feature modelling as variability modelling technique. Finally, we specify our interest on extended PLs by defining terms such as *attribute, non-functional* property or value, *quality, preference*, and *requirement*.

The search string is applied to obtain a first subset of pre-selected articles. This string is subtly modified in each database according to the offered search capabilities (*cf.* Appendix A). A considerable number of duplicate articles is expected during this phase, given the indexing of overlapping papers among databases. For instance, Scopus claims to consider articles from more than 5.000 publishers, which may include articles kept in other selected databases. These duplicate papers are automatically identified and discarded with *EndNote*[2]. Afterwards, the group is filtered according to the relevance of each paper. This task was performed by a junior researcher (*i.e.* Ochoa), and a senior researcher (*i.e.* Pereira) double checked. Then, Ochoa and Pereira filtered the articles by using inclusion and exclusion criteria. Once the selection process was finished, each researcher double checked the work made by the other researcher and the justification for including or discarding each paper. Finally, the articles data was tabulated and double checked by a third senior researcher (*i.e.* González-Rojas).

[2] Found at https://endnote.com/.

3.3. Inclusion and exclusion criteria

We set the following inclusion criteria:

(i) *Articles must be published between January 2000 and September 2016.* We selected January 2000 because in this year the *Software Product Line Conference* (SPLC) was founded. This fact represented a milestone in the PL field and the birth of a community around SPL. Even though two workshops related to PLs were held in 1996 (*i.e.* Workshop on Design and Evolution of Software Architecture of Product Families) and 1998 (*i.e.* International ESPRIT ARES Workshop), neither of them lasted for more than two periods. This lack of continuity shows an immaturity in both the workshops and the PL engineering field. Nevertheless, we reviewed their proceedings, and articles addressed PL issues related to design, architecture, and development process that are out of the scope in this research. Lastly, our search was done in September 2016 that is why papers published before that date are avoided.

(ii) *Articles must be published in journals, conferences, workshops, or symposiums.* These are four of the most known types of research publication in the field of computer science and software engineering.

(iii) *Articles must address extended PLs.* There are multiple studies that address the problem of the semi-automatic configuration of a PL. However, we have interest only in those PL approaches that model non-functional properties. This sets an additional computational complexity, which carries new challenges to be considered by the community.

We define the following exclusion criteria:

(i) *Articles must not be written in a language different to English.* Given that English is the common language in the academic environment, we just consider articles published in the corresponding language.

(ii) *Highly related articles should be discarded.*We only consider the last publish version of a group of articles when they tackled a same topic in a related manner, and some of the authors appeared on all of them.

(iii) *Articles with a missing configuration technique should be discarded.* We included only articles that deal with techniques that support the configuration process of extended PLs, as opposed to considering the product configuration management process in general. Therefore, configuration management sub-processes such as PL version management (*e.g.* how to manage different versions of a PL) or PL testing (*e.g.* verify the correctness of a given PL) are not included in our research. However, it is important to notice that the domain of the studied PL is not considered as part of the exclusion criteria. For instance, the configuration of PLs representing cloud computing services or test scenarios does not suppose an impediment to consider the underlying paper.

(iv) *Short papers must not be considered.* We exclude short papers with less than four pages, since they lack information to evaluate the approach.

3.4. Data collection

Data obtained during the SLR is tabulated in a matrix where each row represents an article, and each column an approach characteristic. First, we extract information about the article publication, mainly its title, authors, year, and type (*i.e.* journal, conference, workshop, or symposium). In addition, related articles of the same research are addressed as only one approach; i.e. when a same author publishes different articles related to a same research line, they are grouped and studied as one approach.

Second, to address *RQ1*, we collect evidence about the employed *configuration technique* by each study. We extract information about the approach algorithms; the PL modelling approach; number of employed models; supported integrity and configuration constraints; defined encodings (*i.e.* how the PL configuration process is translated into a particular technique); and number of supported stakeholders.

Third, to address *RQ2*, we identify the employed *evaluation process* by investigating if each study considers specific metrics during performance tests. Metrics are identified and listed without considering computational complexity when estimating their value. Besides, we extract information about the employed PLs identifying the PL nature (*i.e.* academic, industrial, randomly generated); PL domain (*e.g.* e-commerce, cloud, mobile devices); number of entities as features, CTCs, and non-functional properties; employed non-functional properties (*i.e.* from real case studies, toy examples, randomly generated); and non-functional property types (*e.g.* costs, risks, RAM).

Finally, to address *RQ3*, we consider *gaps and open challenges* related to each studied approach. To collect this qualitative information, we followed a postformed coding technique (Seaman, 1999), where Ochoa and Pereira wrote a set of gaps and open challenges per reviewed article. These two elements were usually captured from the future work, conclusions, or threats to validity sections of a studied article. Once all articles were analysed, both researchers grouped these notes into ten highly related and coupled groups according to the appearing keywords and its underlying meaning. In the end, seven gaps and open challenges were related to *configuration techniques*, whereas three were related to *evaluation processes*.

3.5. Search results

This section presents the number of articles obtained during the search process (*cf.* Section 3.5.1), and the typology of the selected articles with regards to publication year and publication type (*cf.* Section 3.5.2).

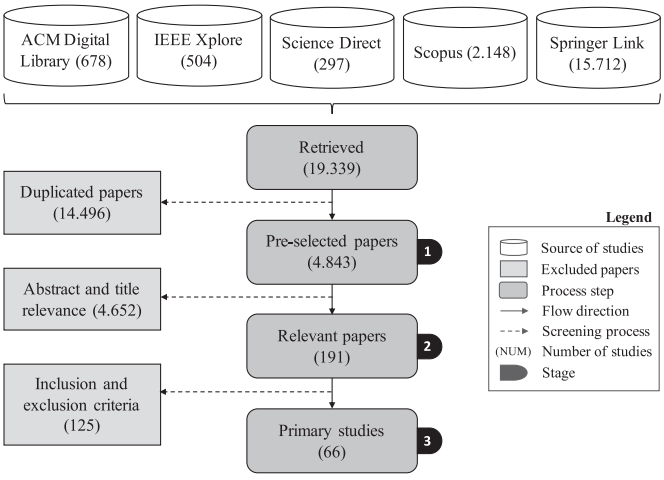


Fig. 3. Selection procedure of primary studies.

Table 2
SLR filtering phases.

Filtering	Articles
Search string	4.843
Abstract and title relevance	191
Inclusion and exclusion criteria	66

Table 3
Databases search results.

Database	Initial	Duplicate	Pre-selected
ACM Digital Library	678	447	231
Science Direct	297	116	181
IEEE Xplore	504	146	358
Scopus	2.148	951	1.197
Springer Link	15.712	12.836	2.876
Total	*19.339*	*14.496*	*4.843*

3.5.1. Articles selection

To select the SLR relevant articles and tabulate the search results, we discard duplicate papers and perform a three-step filtering (see Fig. 3). The number of selected articles per filtering phase are shown in Table 2.

In the first filtering phase, we consider the predefined search string when finding articles in the five selected databases (*cf.* Section 3.2).This phase was performed by Ochoa. Table 3 shows the obtained results of this first step, with a total of 4.843 pre-selected articles. Notice that depending on the order of analysis of duplicate papers, one database can own more articles. Therefore, results in Table 3 absolutely depend on the articles discard order.

Afterwards, we perform a second filtering phase based on the title and abstract of the pre-selected articles. In this step, if the article has a different scope from the one that we were considering, then it is discarded. However, if there is any doubt about the relevance of the paper, it remains in the candidate group. For this second filtering, we obtained 191 candidate papers from the previous 4.843. The big difference is related to the fact that some databases (*e.g.* Springer Link) returned non-relevant articles usually related to different fields such as chemistry and physics. This phase was performed by Ochoa and double checked by Pereira.

During the third filtering phase, Ochoa and Pereira performed a partial or complete reading of the candidate papers. In this step, we apply the inclusion and exclusion criteria for selecting a relevant subset of publications (*cf.* Section 3.3). If the article does not comply with the specified criteria it is discarded, otherwise it is included in the final selection. We obtained 66 articles that were included in the SLR as relevant *primary studies*, 46 of them are considered as *main primary studies* while the remaining ones are considered as short versions of this group. Selected papers are shown in Appendix B. To minimize researcher bias during this phase, Ochoa and Pereira cross-checked the inclusion or exclusion justification provided by the other researcher for each article.

3.5.2. Articles typology

Fig. 4 shows the number of articles per year and publication type. Most of the articles are published in conferences (*i.e.* 28 articles) or journals (*i.e.* 27 articles). The International *Software Product Line Conference* (SPLC) contributes with 10 articles to this SLR. The *Software Quality Journal* (SQJ), the *Journal of Systems and Software* (JSS), and the *International Journal of Production Research* (IJPR) contributes with 3 articles each one of them. In total, there are articles from 20 different

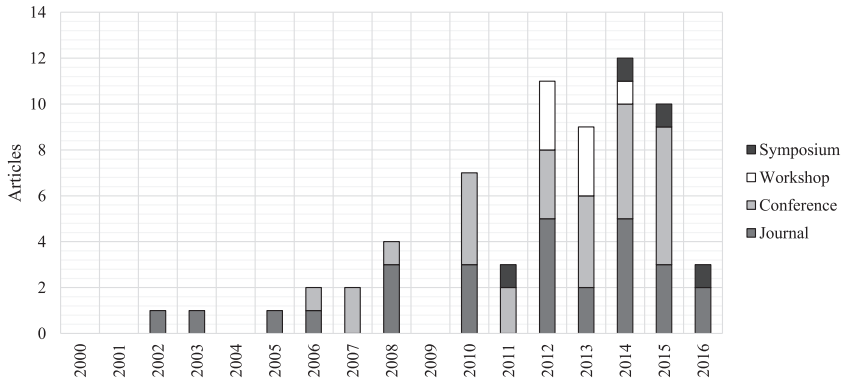


Fig. 4. SLR publications per year.

Table 4
Articles per grouped approach.

ID	Authors	Article(s)
A01	Hierons et al.	Hierons et al. (2016)
A02	Noorian et al.	Noorian et al. (2014, 2016)
A03	Zanardini et al., Villela et al.	Villela et al. (2012); Zanardini et al. (2016)
A04	Henard et al.	Henard et al. (2015)
A05	Dou et al.	Dou et al. (2016)
A06	Gamez et al.	Gamez et al. (2015)
A07	Halim et al.	Halim et al. (2015)
A08	Leite et al.	Leite et al. (2015)
A09	Myllärniemi et al.	Myllärniemi et al. (2015)
A10	Ochoa et al.	Ochoa et al. (2015)
A11	Pascual et al.	Pascual et al. (2015)
A12	Tan et al.	Tan et al. (2015)
A13	Lian and Zhang	Lian and Zhang (2014, 2015)
A14	Asadi et al., Soltani et al.	Asadi et al. (2014); Soltani et al. (2012)
A15	Bagheri	Bagheri and Ensan (2014a)
A16	Bagheri and Ensan	Bagheri and Ensan (2014b)
A17	Tan et al.	Tan et al. (2014)
A18	Lee et al.	Lee et al. (2014)
A19	Olaechea et al.	Olaechea et al. (2014)
A20	Sánchez et al.	Sánchez et al. (2014)
A21	White et al.	White et al. (2010, 2014, 2007, 2008)
A22	El Yamany et al.	El Yamany et al. (2014)
A23	Mussbacher et al., Liu et al.	Liu et al. (2014); Mussbacher et al. (2012)
A24	Sayyad et al.	Sayyad et al. (2013a,c)
A25	García-Galán et al.	García-Galán et al. (2013)
A26	Ghezzi and Molzam Sharifloo	Ghezzi and Molzam Sharifloo (2013)
A27	Chen et al.	Chen et al. (2013)
A28	Karimpour and Ruhe	Karimpour and Ruhe (2013)
A29	Olaechea et al., Murashkin et al.	Murashkin et al. (2013); Olaechea et al. (2012)
A30	Quinton et al.	Quinton et al. (2012, 2013)
A31	Zhang et al.	Zhang et al. (2011, 2014)
A32	Bagheri et al.	Bagheri et al. (2010b, 2012)
A33	Mazo et al.	Mazo et al. (2012)
A34	Ostrosi et al.	Ostrosi et al. (2012)
A35	Schroeter et al.	Schroeter et al. (2012)
A36	Siegmund et al.	Siegmund et al. (2012a,b)
A37	Li et al.	Li et al. (2012)
A38	Asadi et al.	Asadi et al. (2011)
A39	Salinesi et al.	Salinesi et al. (2010)
A40	Bagheri et al.	Bagheri et al. (2010a)
A41	Hong et al., Song et al.	Hong et al. (2008a, 2010a, 2008b, 2006, 2010b); Song et al. (2007)
A42	Shi et al.	Shi et al. (2010)
A43	Siegmund et al.	Siegmund et al. (2008)
A44	Ong et al.	Ong et al. (2006)
A45	Yeh and Wu	Yeh and Wu (2005)
A46	Du et al.	Du et al. (2002, 2003)

journals, 15 conferences, 6 workshops, and 3 symposiums (see Appendix C for a complete list).

Additionally, we observe an increasing tendency of research on the field of extended PL configuration. From 2012 onward, there is a constant concern in this process, and 2014 presents the highest amount of published articles in this field (*i.e.* 12 articles). Data from 2016 cannot be incorporated in the tendency analysis because we did not consider all months of the year. Moreover, many accepted papers addressing the scope of this SLR have not been published. Consequently, those papers were not considered in our analysis.

Finally, after reviewing each article we defined a set of 46 grouped approaches for answering the proposed research questions. Table 4 presents the complete list of approaches including their IDs (used as references along the paper), involved authors, and included articles.

4. Analysis of configuration techniques on extended PLs

This section describes the algorithms and techniques employed to support the semi-automatic configuration of extended PLs (*cf. RQ1*). Section 4.1 presents different techniques for encoding a PL and the implementation particularities of the analysed approaches. Section 4.2 highlights the configuration constraints employed in the studied approaches, and their frequency of use according to each encoding technique.

4.1. Product line encoding algorithms

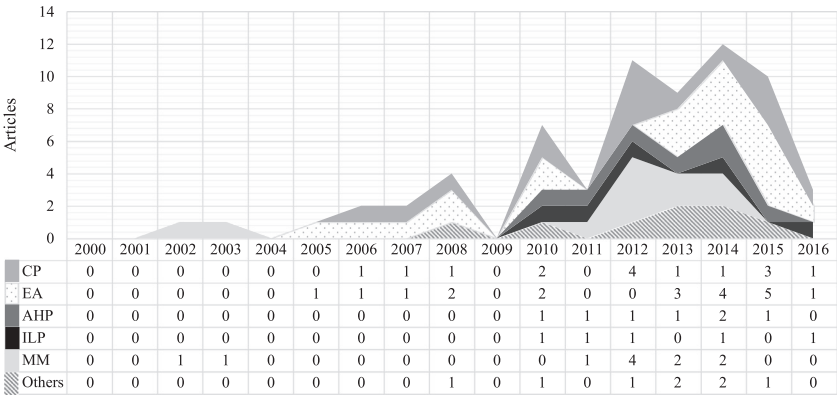
After performing the complete reading of all selected *main primary studies*, we identified six groups of techniques used to automate the configuration of extended PLs: (i) *Constraint Programming* (CP), (ii) *Evolutionary Algorithms* (EA), (iii) *Analytic Hierarchical Process* (AHP), (iv) *Integer Linear Programming* (ILP), (v) *Mapping and Models* (MM), and (vi) *Other* methods. The latter group includes techniques such as CP used exclusively for solving *knapsack problems* (Shi et al., 2010), *recommendation techniques* based on gambling queries (Bagheri and Ensan, 2014a), and *ad-hoc* algorithms (Siegmund et al., 2008), as well as hybrid solutions. Fig. 5 shows the number of articles per year and employed technique. As it can be seen, EA and CP are the most used techniques appearing in 20 and 15 publications respectively.

Hereafter, we present a brief description of each encoding technique and the implementation particularities of the *main primary studies*.

4.1.1. Encoding in constraint programming

Constraint programming (CP) is a paradigm that proposes an *exact* approach (Olaechea et al., 2014), where *Constraint Satisfaction Problems* (CSP) are specified as a set of variables with their corresponding domain, and a set of constraints that affect these domains (Apt, 2003; Rossi et al., 2006).

In order to configure a product in an extended PL, features are



	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016
CP	0	0	0	0	0	0	1	1	1	0	2	0	4	1	1	3	1
EA	0	0	0	0	0	1	1	1	2	0	2	0	0	3	4	5	1
AHP	0	0	0	0	0	0	0	0	0	0	1	1	1	1	2	1	0
ILP	0	0	0	0	0	0	0	0	0	0	1	1	1	0	1	0	1
MM	0	0	1	1	0	0	0	0	0	0	0	1	4	2	2	0	0
Others	0	0	0	0	0	0	0	0	1	0	1	0	1	2	2	1	0

Fig. 5. PL configuration articles per technique.

Table 5
FM encoding based on CP.

Integrity constraint	Encoding
Mandatory	$f_p = f_c$ Salinesi et al. (2010)
Optional	$f_p \geq f_c$ Salinesi et al. (2010)
Or	$\sum_{i=1} f_{ci} > 0$ Mannion (2002)
Alternative	$\sum_{i=1} f_{ci} = 1$ Salinesi et al. (2010)
Requires $f_1 \rightarrow f_2$	$f_1 \leq f_2$ Salinesi et al. (2010) or $f_1 = 1 \rightarrow f_2 = 1$ Ochoa et al. (2015)
Excludes $f_1 \rightarrow \neg f_2$	$f_1 + f_2 \leq 1$, $f_1 \times f_2 = 0$ Salinesi et al. (2010), or $f_1 = 1 \rightarrow f_2 = 0$ Ochoa et al. (2015)
Cardinality [*min*, max]	$(\sum_{i=1} f_{ci} \geq \min) \wedge (\sum_{i=1} f_{ci} \leq \max)$ Salinesi et al. (2010)
Children selection	$f_2 = 1 \rightarrow f_1 = 1$ Ochoa et al. (2015)
Parent deselection	$f_1 = 0 \rightarrow f_2 = 0$ Ochoa et al. (2015)

considered as binary variables in CSP. Then, for n features in the model, n variables are created, such that each variable $f_i \in \{0, 1\}$. Furthermore, *Non-Functional Properties* (NFP) are represented as integer variables whose domain correspond to the one defined in the FM. A NFP a related to the i^{th} feature is defined as $f_i \cdot a \in D_i$. The value of a NFP a corresponds to the aggregation sum of the NFP a value from each selected feature (White et al., 2007).

The solution of the modelled CSP is obtained by finding a suitable configuration that meets integrity and configuration constraints (Salinesi et al., 2010; Siegmund et al., 2012b). Integrity constraints encodings are shown in Table 5, where f_p stands for *parent feature*, f_c for *child feature*, $f_c i$ for the i^{th} child feature of an integrity constraint. The selection among encodings can affect the execution time during CSP solving.

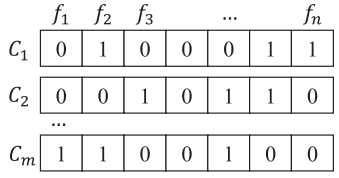
The encodings of configuration constraints differ among solutions. For instance, optimization objectives are represented as maximization or minimization functions in a CSP, whereas for hard limits over an attribute type, an inequality operator is defined over the aggregated value of the attribute and the given boundary value (*e.g.costs* $>$ *1.000*) (Ochoa et al., 2015). Nevertheless, there are some particularities among the studied approaches. Some solutions first prune the search space by representing a subset of features from the FM (García-Galán et al., 2013; White et al., 2007). Ong et al. (2006) propose an *invasion algorithm* that takes a binary CSP and traduces it to a directed graph representing a set of CSP solutions. Other approaches propose encodings for additional needs in the configuration problem. For instance, White et al. (2014) manage different CSPs at different time slots, then there are $n \times k$ variables related to the n features of the PL in k time slots. (Zanardini et al., 2016) enrich the CSP representation with decision trees that are known as *configuration trees* where each node represents a PL partial configuration, and each edge an increase of the set of features (*i.e.* an increment in functionality). Then nodes are not singleton sets with a single feature, but instead they are a complete configuration that may be invalid.

White et al. (2010) present a diagnostic approach, known as *Configuration Understanding and REmedy* (CURE). CURE is used when multiple stakeholders define a conflicting configuration and a repair is needed. In this context, a minimal set of selected and deselected features is proposed. Similarly, Ochoa et al. (2015) face these conflicts by considering a set of stakeholders' complete configurations to reduce the search space. In both approaches CSP techniques are applied to derive a product based on all stakeholders needs.

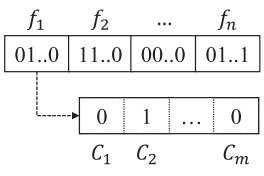
4.1.2. Encoding in evolutionary algorithms

Evolutionary Algorithms (EA) are stochastic or approximate algorithms (Olaechea et al., 2014) useful during optimization or learning tasks implementation (Yu and Gen, 2012). They emulate the natural species evolution, where given a first population of m individuals living in an environment with limited resources, there is a competition were only the fittest individuals survive (Eiben and Smith, 2003; Yu and Gen, 2012). An *individual* is related to one or more *chromosomes* that are composed by *n genes*. *Individuals* are randomly generated, or some heuristics are employed to obtain a higher fitness in this first sample. Then, each individual is evaluated according to the *fitness function* (*i.e.* optimization objectives and environmental constraints) to produce *multiple generations* derived from the initial population. Each individual is a representation of a *solution* in a particular domain.

Genetic algorithms (GA) and *Multi-Objective Evolutionary Algorithms* (MOEA) are commonly used in PL configuration. GAs are known to have a binary code that emulates the natural genetic encoding (Yu and Gen, 2012). MOEAs were designed for solving *multi-objective optimization problems*. The fitness function and operations behaviour change among different algorithms. In the PL field, in order to obtain solutions



(a) Binary string EA-based encoding.



(b) EA-based encoding with m products.

Fig. 6. FSG configurations and solution constraints.

Table 6
FM encoding based on ILP.

Integrity constraint	Encoding
Mandatory	$f_p = f_c$ Noorian et al. (2016) or $f_p \leq f_c$ Li et al. (2012)
Optional	$f_p \geq f_c$ Noorian et al. (2016)
Or	$\sum_{i=1} f_{ci} \geq f_p$ Noorian et al. (2016)
Alternative	$\sum_{i=1} f_{ci} = f_p$ Li et al. (2012); Noorian et al. (2016)
Requires $f_1 \rightarrow f_2$	$f_1 \leq f2$ Li et al. (2012); Noorian et al. (2016)
Excludes $f_1 \rightarrow \neg f_2$	$f_1 + f_2 \leq 1$ Li et al. (2012); Noorian et al. (2016)
Paternity	$f_c \leq f_p$ Li et al. (2012)

that conform to the integrity constraints, *correctness* is defined as one of the optimization objectives. Finally, a trade-off is performed among the multiple optimization objectives (Yu and Gen, 2012).

The general encoding of EA-based approaches in the PL configuration context considers a *binary string* representation as proposed with GAs. Then, for n features in the PL, a $1 \times n$ vector is generated in order to represent a configuration (*i.e.* chromosome), *i.e.* $C_j = [v(f_1), v(f_2), ..., v(f_n)]$ such that $f_i \in F$, $v(f_i) \in \{0, 1\}$, $i, j \in \mathbb{N}$, and $i < n$ (*cf.* Fig. 6a) (El Yamany et al., 2014; Lian and Zhang, 2015; Olaechea et al., 2014; Pascual et al., 2015).

This encoding can be reduced by removing some *prunable features* (*core* and *dead features*) from the binary string representation (Tan et al., 2015). An analysis that considers both sets is unnecessary, since core features are present in all products, and dead features are never included. Hierons et al. (2016) also remove selected parent features if one or more child features are selected. These features are added back when the remaining decisions are defined. Karimpour and Ruhe (2013) also employ a single binary encoding, however they have a particular need on representing multiple products in the same vector. Thus, in each of the n positions of the vector, they defined m bit values. Each of these bit values represents the selection state of a given feature related to one of the m considered products (*cf.* Fig. 6b).

Nevertheless, not all EA-based approaches use the binary string representation. For instance, Yeh and Wu (2005) and Dou et al. (2016) also represent chromosomes in a $1 \times n$ vector, however $v(f_i)$ will be equal to an integer value related to the selected characteristic of the given feature. In this context, features are not necessarily represented as boolean variables, but instead as integer variables with a finite domain representing different options for a decision. For example, suppose there is a feature *color* $\in [1, 3]$, where 1 represents *white*, 2 represents *purple*, and 3 represents *green*. Moreover, Hong et al. (2010a) employ *co-evolutionary programming*. In this approach AND-OR trees are used to represent both product design and manufacturing process. So, each node in the tree represents a product characteristic but also the manufacturing process that should be employed to derive that characteristic.

4.1.3. Encoding in analytic hierarchy process

According to (Saaty, 2008), *Analytic Hierarchy Process* (AHP) is a theory of measurement employed in scenarios where a decision considers multiple factors, and a trade-off among those factors is needed.

AHP has been employed to rank the features of a PL in order to select the best candidates for a final product. Thus, a $n \times n$ square matrix is constructed based on the n features present in the PL. Each cell in the matrix will represent the relative importance of the i^{th} feature with regards to the j^{th} feature. In this case, $n \times \frac{(n-1)}{2}$ comparisons among features are performed. The traditional values used for representing the degree of importance among two features are the integers 1, 3, 5, 7, and 9; where the highest values denote a higher importance. Values of the matrix are then normalized based on the *eigenvalues* estimation (Saaty, 1987). The v eigenvector is a non-null vector such that given a matrix A and an eigenvalue λ, we have the following relation $Av = \lambda v$. Finally, the most important features are included in the product (Bagheri et al., 2010a; Tan et al., 2014; Zhang et al., 2014).

Zhang et al. (2014) introduce the usage of positive and negative impacts over NFPs. An NFP positive impact measures the importance of including a feature in a final product configuration, while the negative impact measures the importance of discarding the feature in the final product configuration. Similarly, just the subset of features with positive or negative impact over a particular NFP are considered in the AHP matrix, reducing the number of comparisons to be performed by stakeholders.

On a similar context, Bagheri et al. (2010a) present the *Stratified AHP* (S-AHP), a method that does not compare PL features, instead it compares NFPs. A square matrix $M_{k \times k} = \{M_{i,j} = \alpha | 1 \leq i, j \leq k\}$ is created, such that α represents the relative importance of the i^{th} NFP with regards to the j^{th} NFP. A first comparison among high level and possibly fewer concerns is performed, reducing the amount of pair-wise comparisons among features. This approach was also adopted by Asadi et al. (2014) and Halim et al. (2015). In addition, Halim et al. (2015) propose the use of the *Triangular Fuzzy Number* (TFN) for specifying the relative importance among NFPs in the AHP matrix. TFN manages the fuzziness of stakeholders' decisions by defining a triple (a, b, c) where three consecutive numbers in a scale from 1 to 9 are defined, then $a, b, c \in [1, 9]$, $b = a + 1$, and $c = b + 1$. In a similar way, Tan et al. (2014) estimate the relative importance based on the ELO rating system, a technique used to score chess players (Elo, 1978).

4.1.4. Encoding in integer linear programming

Integer Linear Programming (ILP) is a method that considers a set of *decision variables* $V = \{x_1, x_2, ldots, x_n\}$ with integer numeric values and they should be optimally selected; and a set of *constraints* $C = \{C_1, C_2, ..., C_m\}$ that are equalities or inequalities related to a linear combination of a subset of variables in V. For both cases $n, m \in \mathbb{N}$. The selection is based on an objective linear function f that should be *minimized* or *maximized* (Vanderbei, 2014). A solution to an ILP problem is defined by the set $S = \{s_1, s_2, ..., s_i\}$. If all constraints in C are satisfied, we refer to S as a *feasible* solution (Vanderbei, 2014).

An ILP model represented in the PL configuration context is composed by a set of n variables f_i that represent features in the FM, such that $f_i \in \{0, 1\}$; a set of constraints that represent both integrity and configuration constraints; and a SOO function (Noorian et al., 2016). When all variables have a binary domain, the approach is known as *0–1 programming* (Li et al., 2012).

Similar to CP-based approaches (*cf.* Section 4.1.1), multiple encodings have been proposed to translate integrity constraints. Some examples are shown in Table 6, where f_p stands for *parent feature*, f_c for *child feature*, and $f_c i$ for the i^{th} child feature of an integrity constraint. Multiple encodings are similar or equal to the ones presented in the CP encodings, given their nature of expressing constraints over integer variables.

In order to specify additional configuration constraints, some of the ILP-based approaches have introduced extensions to the FM representation. For instance, Bagheri et al. (2010b) introduce the language $L(N)$, which allows the specification of integrity constraints of a FM, and a set of selected or required features (which are known as hard constraints in the approach). A fuzzy extension of the language is also presented to specify soft constraints, which are stakeholders' preferences where quality attributes are prioritized. Moreover, Lopez-Herrejon and Batory (2001) transform the FM into a *Graph Product Line* (GPL) (Lopez-Herrejon and Batory, 2001), and propose a technique that allows the evaluation of PL methodologies by using the GPL encoding as main evaluation problem.

4.1.5. Encoding in model and mapping approaches

Model and Mapping (MM) approaches employ a modelling technique and a set of mapping rules to derive PL configurations based on decision

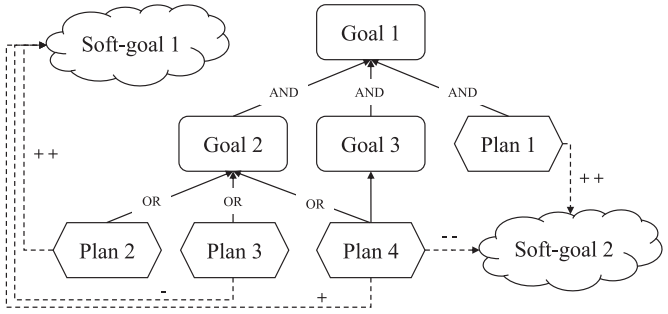


Fig. 7. Goal model example.

propagation from variability models. This approach does not necessarily mean that we are dealing with environments with multiple PLs. Sometimes, models are used to complement the information provided in another model. The three main techniques used among the studied approaches correspond to *goal models, ontologies*, and *grammars*.

Goal models are defined as AND/OR graphs that supports the specification of *goals* (*i.e.* objectives related to functional requirements), *soft-goals* (*i.e.* objectives related to non-functional requirements), and *plans* (*i.e.* task to operationalize goals) (*cf.* Fig. 7). Two links are admitted, mainly *decomposition* links that define a hierarchy among goals, generating a tree-like structure with *and/or* associations; and *contribution* links enriched with *help* (*i.e.* +), *hurt* (*i.e.* -), *make* (*i.e.* ++), and *break* (*i.e.* --) labels, which define how a goal model entity contributes to a particular soft-goal (Asadi et al., 2011; Yu et al., 2008). An *ontology* is an engineering artefact that represents a domain by a formal representation of shared concepts (Quinton et al., 2012; Sánchez et al., 2007; Shekhar and Xiong, 2008). A *grammar* is a set of structural rules that supports the specification of expressions in a given language (Seel, 2012).

In the context of *goal models*, Asadi et al. (2011) use a particular type of models known as *i* models*. These models are related to FMs by means of *presence conditions*, which are references to goals. They are evaluated based on the goal satisfaction degree of the feature: *Full Satisfaction* (FS), *Full Denial* (FD), *Partial Satisfaction* (PS), and *Partial Denial* (PD). A goal satisfaction degree is defined as a boolean formula, and its value is derived based on feature annotations made by stakeholders. Similarly, Lee et al. (2014) use goal models to complement the information presented in FMs. Then, in a matrix that interrelates goals and features, annotations are described in each cell to define the correlation among the selection of goals and features; i.e. if a goal is interrelated with a feature, its selection supposes the selection of the related feature. Moreover, they provide different FMs to separate the problem space from the solution space by its goal-driven FM approach. Lastly, Mussbacher et al. (2012) present *Aspect-oriented User Requirements Notation* (AoURN), a SPL framework for specifying stakeholders' goals needed during product configuration. Cross-cutting concerns are treated as aspects in the modelling approach. Thus, the framework considers goal models and FMs to represent stakeholders needs and PL features. Then, they are related in an impact model, where the impact of a feature on each goal is identified.

In the *ontologies* context, (Chen et al., 2013) describe fuzzy concepts with three main elements, mainly *property, value*, and *constraints*. In addition, (Quinton et al., 2013) use ontology engineering to connect and map concepts to features in the FM.

Finally, in the *grammar* context, (Ostrosi et al., 2012) propose the use of fuzzy solutions with a fuzzy configuration grammar. Similarly, (Du et al., 2003) use a graph grammar to manipulate a set of operations such as *attaching, removing*, and *swapping* features, or *scaling* NFPs in a PL.

4.1.6. Encoding in other methods

We found other approaches that could not be classified in the previous groups. In this section, we present a brief description of these approaches.

Olaechea et al. (2012) present ClaferMoo as an extension of Alloy, a relational modelling language. This work allows the specification of multi-objective functions in the Clafer language (Bak et al., 2011) to obtain PL configurations that respond to stakeholders' needs (*i.e.* multi-optimization objectives). Siegmund et al. (2008) adopt an ad-hoc algorithm for finding an optimal product in a PL. In this approach, a staged product derivation (Czarnecki et al., 2004) is considered to reduce the number of products and the complexity of the solution search. First, functional requirements are defined to reduce the search space. Then, limits over NFPs are defined. Finally, the optimization process is performed finding local optimum solutions until a time limit is reached.

Sánchez et al. (2014) also use an ad-hoc algorithm known as *Configuration Selection Algorithm* (CSA) to find a product by considering (i) a set of resource constraints; (ii) a partial configuration; and (iii) a weighted objective function. CSA performs its search over a *binary tree* where nodes represent valid states, including partial and complete configurations, and edges represent the selection or deselection of a given feature. The objective function is presented in Eq. 1, where w_i is the weight of the i^{th} NFP such that $\sum_{i=1}^{|NFP|} w_i = 1$, $NFP_i(C)$ is the configuration aggregated value of the i^{th} NFP, μ_i is the average value of $NFP_i(C)$, and σ_i is the standard deviation of the same function.

$$F(C) = \sum_{i=1}^{|} NFP|w_i \times \frac{NFP_i(C) - \mu_i}{\sigma_i} \quad (1)$$

Moreover, Shi et al. (2010) present a knapsack-based algorithm that modifies the *Filtered Cartesian Flattening* (FCF) algorithm (White et al., 2009). FCF uses the *Multiple-choice Multi-dimensional Knapsack Problem* (MMKP) algorithm to select features from a PL. In this approach, the MMKP algorithm is replaced with a simple knapsack. In summary, FCF takes a FM and creates a finite number of sets in monolayer. Then, K items are selected based on their value and cost, and with the knapsack algorithm an optimized selection is performed (the higher K, the higher is the optimization rate).

Bagheri and Ensan (2014a) also propose a staged PL configuration process. Decision models are generated dynamically from a FM. The complete process is described as a *utility elicitation*, where features are ranked and selected based on *gamble queries* (*i.e.* stakeholders decisions). Once a feature is selected or deselected *utility propagation* is performed (*i.e.* the utility of the selected or deselected feature and related features is positively and negatively updated).

Myllärniemi et al. (2015) highlight security concerns during PL configuration by considering different countermeasures when trade-offs among different NFPs are needed. A *countermeasure* is the specification of an action that prevents, detects, or reacts to threats, attacks, or vulnerabilities. The problem is traduced to an *Answer Set Programming* (ASP) problem by the employment of *Weight Constraint Rules* (WCR). Configurations are then generated as stable models.

4.2. Product line configuration constraints

We classify configuration constraints in the following 10 main groups.

Single-objective optimization (SOO) problem, also known as the *one-objective optimization* problem, is defined as the *minimization* or *maximization* of a function $f(x)$, where x is a vector of decision variables $x = (x_1, x_2, ..., x_n)$ where $n \in \mathbb{N}$. Elements in x could be continuous or discrete variables, and usually one or more global solutions are found. Moreover, a set of m *inequality* constraints $g_i(x) \leq 0$ or a set of p *equality* constraints $h_j(x) = 0$ can be defined, such that $i, j, m, p, n \in \mathbb{N}$, $i < m$, and $j < p$. Thus, $p < n$ in order to obtain a non-over-constrained problem (Coello et al., 2006).

Multi-objective optimization (MOO), also known as *multi-criteria optimization*, is known to be a NP-complete problem. In contrast to the

SOO, $f(x)$ is a vector of k objective functions where $F(x) = [f(x)_1, f(x)_2, ..., f(x)_k]$ and $k \in \mathbb{N}$. Each objective function should be minimized or maximized. As multiple solutions can be found in the Pareto front, a solution for a MOO problem is a trade-off among the considered objectives in function $F(x)$ (Coello et al., 2006).

Preferences and weights are usually employed to describe the importance of specific non-functional properties. Different approaches are used to specify in which degree a specific non-functional property is preferred over others. This is done in a competitive environment where trade-offs are required in order to better satisfy stakeholders' needs. One of the methods used for expressing *stakeholders' preferences* is the employment of *weighting factors*. For instance, Hong et al. (2010a) propose the employment of Eq. 2 in order to define *weighting factors*. In this case, f_i corresponds to the evaluation measure of the i^{th} aggregated value of a non-functional property related to a given product C; I_i is known as the *evaluation index* and it is responsible of converting evaluation measures in a normalized and comparable value, $I_i \in [0, 1]$; and all W_i are the *weighting factors* related to each evaluation index. They are defined by stakeholders or measured by a domain-specific formula. $F(C)$ is then used in a SOO problem (Hong et al., 2010a).

$$F(C) = \frac{\sum_{i=0}^{m-1} W_i I_i(f_i)}{\sum_{i=0}^{m-1} W_i} \quad (2)$$

The weight of a product is multiplied by its own value, which is based on the sum of value of each feature contained in the configuration (Karimpour and Ruhe, 2013). Similarly, Ghezzi and Molzam Sharifloo (2013) define a *utility function* related to a complete configuration as $F(C) = w_1p_1 + w_2p_2 + \cdots + w_lp_l$ where w_i is the weight assigned to the i^{th} non-functional property, p_i is the satisfaction degree of the set of l NFPs that should be satisfied, and $i, l \in \mathbb{N}$. Lastly, weights can also be defined as a pairwise comparison among different factors as proposed by AHP approaches (*cf.* Section 4.1.3).

Extra-functional constraints compute feature attributes as the aggregation of feature attributes of other features (Benavides et al., 2005). These constraints are expressed as arithmetic constraints. For example, suppose there is a non-leaf feature f_1 that has a *cost* attribute defined by the following EFC, $f_1.cost = 2 \cdot f_2.cost + 4 \cdot f_3.cost$, for $f_1, f_2, f_3 \in F$. This means that the value of the *cost* attribute of f_1 is equal to the sum of the *cost* value of f_2 times 2, plus the *cost* value of f_3 times 4.

Filter is an automated analysis operation where a reduced search space is explored in order to select a PL configuration (Benavides et al., 2010). First, stakeholders select a set of features (*i.e.* a valid partial configuration) and key requirements to reduce the PL variability space. Then, a subset of products S' are obtained $S' \subseteq S$ from a complete set of products or solutions S (Benavides et al., 2010; Czarnecki and Kim, 2005).

Seeding is a constraint used as a strategy in EA-based approaches. In these approaches, the initial population is generated or *seeded* with "good" individuals, based on a set of optimization objectives specified by stakeholders. Then, the algorithm starts from a mature population that already responds to a group of requirements and continues improving the quality of the derived off-springs. The implementation of *seeding* constraints is done by the use of heuristic or deterministic algorithms (Coello et al., 2006; Eiben and Smith, 2003).

Hard limits, also known as *resource constraints*, are constraints that are defined as *equalities* or *inequalities* over the aggregation of a specific NFP. These constraints complement the definition of optimization functions, for both SOO and MOO problems (*cf.* SOO and MOO). In this way, a set of feature attributes related to the same NFP (*e.g. cost*) are aggregated (*e.g. sum*) and an inequality (*i.e.* $\leq$, $<$, $>$, $\geq$) or equality (*i.e.* $=$) relation is defined between the corresponding aggregation and a constant. These constraints are useful for defining resource limits like budget boundaries (Ochoa et al., 2015). For instance, Bagheri and Ensan (2014b) employ hard limits in order to define *reliability* upper and lower boundaries when configuring a PL.

Finite domain Constraint used when feature attributes should be defined in a sub-domain of values. Thus, given a feature f_i with a feature attribute a_j defined in a domain D_{ij}, $a_j \in D_{ij}$, and $i, j \in \mathbb{N}$, a new domain D'_{ij} can be defined, such that $D'_{ij} \subseteq D_{ij}$ and now $a_j \in D'_{ij}$ (Ochoa and González-Rojas, 2017). Finite domain constraints are useful when, for certain scenarios, a subset of the complete NFP domain is required.

Cross-model constraints (CMC) are defined among features in different models. Similar to CTC, the most common representations are *requires* and *excludes* implications. As an example, suppose we have two FMs, FM_1 and FM_2 each one with a set of features F_1 and F_2, respectively. A *requires* CMC can be established between $f_{1i} \in F_1$ and $f_{2j} \in F_2$, such that $f_{1i} \rightarrow f_{2j}$, $i, j \in \mathbb{N}$, $i < |F_1|$, and $j < |F_j|$. Similarly, an *excludes* CMC can be represented as $f_{1i} \rightarrow \neg f_{2j}$ (Chavarriaga et al., 2014; Metzger et al., 2007). Notice that as with CTCs, complex relations among features of different variability models can also be defined as propositional logic predicates; features are represented as boolean variables and relations among features with operators such as $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$, and $\neg$ (Batory, 2005).

Fuzzy requirements Stakeholders' requirements surrounded by vagueness and uncertainty (Chen et al., 2013; Halim et al., 2015). They are represented with diverse fuzzy membership functions such as *Trapezoidal Fuzzy Number* (TFN) and *Triangular Fuzzy Number* (TFN). In the TFN context, three integer values are used to express the stakeholders' needs $< v_l, v_i, v_u >$, where v_l is the lower bound of the ideal value, v_i is the ideal value, and v_u is the upper bound of the ideal value (Bagheri et al., 2010b; Chen et al., 2013; Halim et al., 2015). However, fuzzy linguistic variables defined in an ordinal scale may also be specified from natural language and imprecise requirements, such as *high, medium*, and *low* (Bagheri et al., 2010a). Chen et al. (2013) define fuzzy requirements by means of a triple $<$ *property, value, constraints* $>$ where the *property* is related to an NFP (*e.g.* costs), the fuzzy requirement is represented as a particular *value* (*e.g.* low), and some additional *constraints* are specified over such property value (*e.g.* costs should be less than a given value).

Table 7 groups the studied approaches by the different used techniques and algorithms. Each approach is evaluated in terms of its capability to support configuration constraints: ✓completely supported, • partially supported, and empty space as not supported. The acronyms of each column title are defined as follows: *single-objective optimization* (SOO), *multi-objective optimization* (MOO), *hard limit* (HL), *preferences and weights* (P), *finite domain* (FD), *filter* (F), *cross-model constraint* (CMC), *extra functional constraint* (EFC), *seeding* (S), *fuzzy requirements* (FR), and *others* (*O). Hereafter, we highlight the main results related to each configuration technique.

CP-based approaches support SOO and hard limit constraints. Filter and EFC constraints are supported by 6 and 5 studies respectively. These configuration constraints are supported given the nature of the approach, which allows the definition of logical and arithmetical constraints, as well as optimization objectives. Other configuration constraints are also supported by these approaches including MOO, preferences and weights, finite domain, and other uncommon constraints such as CP *symbolic* (*e.g.* all different, at least, at most) and *reified constraints* (Mazo et al., 2012; Salinesi et al., 2010). The only two constraints that have not been addressed by CP approaches correspond to seeding which is an inherent option of EA-based approaches, and fuzzy requirements.

EA-based approaches support SOO and most of them support MOO (10 out of 13 studies). Some few approaches support hard limits, preferences and weights, and filter. Seeding, given its direct relation with EAs, is also supported by 3 studies. However, finite domain and fuzzy requirements are not employed by EA-based approaches.

AHP-based approaches support *preferences and weight* constraints. This is related to the pairwise comparison that is inherent to AHP solutions. Three approaches partially support fuzzy requirements (Asadi et al., 2014; Bagheri et al., 2010a; Halim et al., 2015), and just one approach supports multiple configuration constraints such as SOO, hard

Table 7
Configuration constraints supported by different approaches.

	Approach ID	SOO	MOO	HL	P	FD	F	CMC	EFC	FR	S	*O
CP-based	A03 - Zanardini et al.	✓		✓			✓		✓			
	A06 - Gamez et al.	✓		✓				✓	•			
	A08 - Leite et al.	✓	✓	✓	•			•				
	A10 - Ochoa et al.	✓		✓			•					
	A21 - White et al.	✓		✓	•							
	A25 - García et al.	✓		✓			✓		✓			
	A33 - Mazo et al.	✓		✓		•	✓		✓			✓
	A36 - Siegmund et al.	✓		✓	•		✓		✓			
	A39 - Salinesi et al.	✓	✓	✓		✓						✓
	A44 - Ong et al.	✓	•	✓	✓		✓	•				
EA-based	A01 - Hierons et al.	✓	✓									
	A04 - Henard et al.	✓	✓								✓	
	A05 - Dou et al.	✓		•	✓						✓	
	A11 - Pascual et al.	✓	✓				•				✓	
	A12 - Tan et al.	✓	✓									
	A13 - Lian et al.	✓	✓									
	A16 - Bagheri et al.	✓		✓			✓					
	A19 - Olaechea et al.	✓	✓									
	A22 - El-Yamany et al.	✓	✓				✓					
	A24 - Sayyad et al.	✓	✓						✓			
	A28 - Karimpour et al.	•	•		✓							
	A41 - Hong et al.	✓	•	✓	✓			•				
	A45 - Yeh et al.	✓					•					
AHP-based	A07 - Halim et al.				✓					•		
	A14 - Asadi et al.	✓		✓	✓		•		•	•		
	A17 - Tan et al.				✓							
	A31 - Zhang et al.				✓							
	A40 - Bagheri et al.				✓					•		
ILP	A02 - Noorian et al.	✓			✓			✓				
	A32 - Bagheri et al.	✓		✓	✓		•			✓		
	A37 - Li et al.	✓		✓	•							
MM-based	A18 - Lee et al.				•			✓				
	A23 - Mussbacher et al.				•			✓				
	A27 - Chen et al.							✓		✓		
	A30 - Quinton et al.			✓				✓				
	A34 - Ostrosi et al.							✓		✓		
	A35 - Schroeter et al.			•				✓				
	A38 - Asadi et al.				✓			✓				
	A46 - Du et al.			•			•	✓				✓
Other	A09 - Myllärniemi et al.							✓	✓			
	A15 - Bagheri et al.				✓					✓		
	A20 - Sánchez et al.	✓		✓	•		✓		✓			
	A26 - Ghezzi et al.	•		✓	✓				✓			
	A29 - Olaechea et al.	✓	✓				✓		✓			
	A42 - Shi et al.	✓		✓	•							
	A43 - Siegmund et al.	✓		✓			•		✓			

✓: completely supported, •: partially supported.

limit, preferences and weight, filter, EFC, and fuzzy requirements (Asadi et al., 2014). This last study is a hybrid solution that employs three different methods: AHP, FCM, and HTN.

ILP-based solutions support SOO and preference constraints. These preferences are usually specified as weights in a SOO function. These configuration constraints respond to the definition of ILP models. Additionally, two of these approaches consider hard limits (Bagheri et al., 2010b; Li et al., 2012), and one of them also support filter, CMCs, and fuzzy requirements. MOO, finite domain, EFC, seeding and other type of configuration constraints are not considered in the studied ILP approaches.

MM-based approaches support CMC constraints, given the need to propagate decisions among different models. Three of these approaches also support hard limits (Du et al., 2003; Quinton et al., 2013; Schroeter et al., 2012), other three support preferences and weights (Asadi et al., 2011; Lee et al., 2014; Mussbacher et al., 2012), and two support fuzzy requirements (Chen et al., 2013; Ostrosi et al., 2012). In the case of (Du et al., 2003), filter and *production rules*, which were previously described (*cf.* Section 4.1.5) are also supported. SOO, MOO, finite domain, EFC, and seeding are not considered by any of the selected studies.

In the case of *other approaches*, no pattern is identified given the diversity of employed methods. For instance, Myllärniemi et al. (2015), who employ an ASP solution, support CMCs and EFCs. (Olaechea et al., 2012) and (Siegmund et al., 2008) support SOO, hard limits, filter, and EFCs. In the first case preferences and weights are also considered. In the case of relational modelling (Olaechea et al., 2012) SOO, MOO, filter, and EFC are included in the solution. The knapsack-based approach (Shi et al., 2010) supports SOO, hard limits, and preferences and weights given the nature of the knapsack problem. Lastly, the utility elicitation approach considers preferences and weights, as well as fuzzy requirements. Finite domain, seeding, and other configuration constraints are not considered by any of the aforementioned studies.

As conclusion, we identified that CP and ILP-based approaches tend to support SOO and hard limit constraints. Other configuration constraints as filtering and EFC are also supported. EA-based approaches are suitable for solving SOO and MOO problems. Seeding is an additional configuration constraint which is inherent to these algorithms.

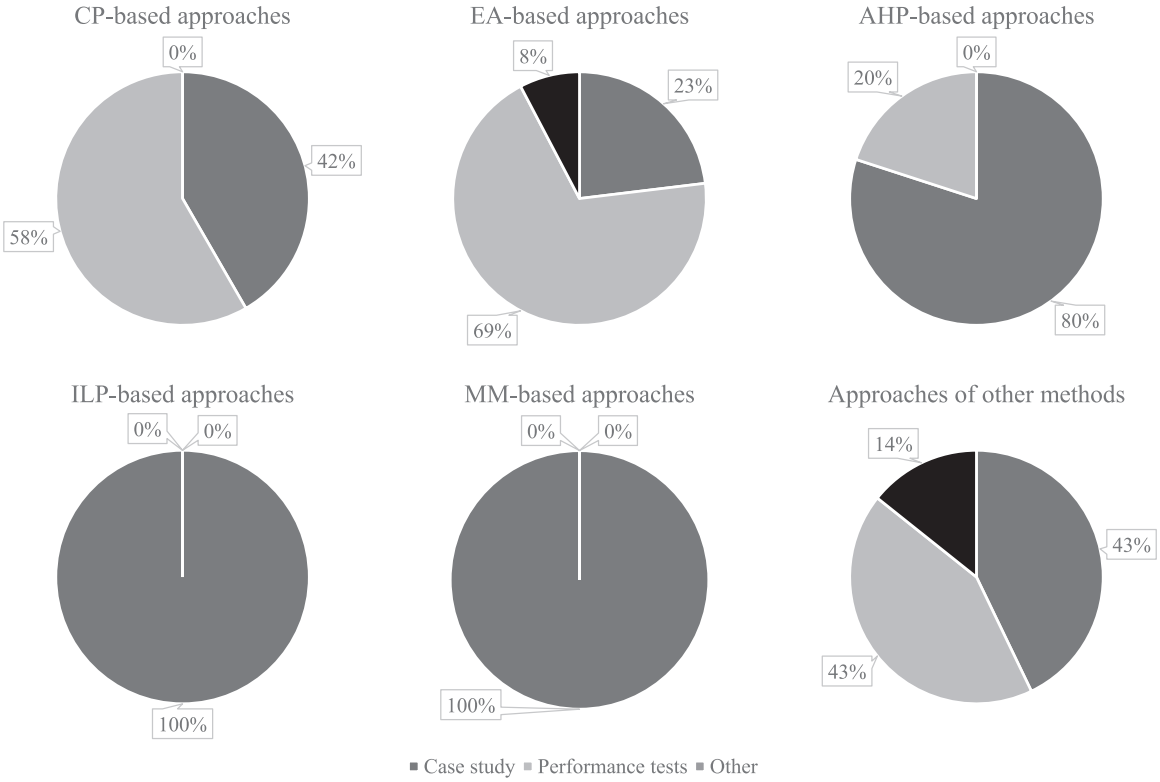


Fig. 8. Approaches evaluation techniques per method.

Table 8
Performance and scalability tests of existing approaches.

	Approach ID	Method	FM	#F	#CC	ET
CP-based	A06 - Gamez et al.		Booking	200	5	> 10 s
	A21 - White et al.		Generated	5.000	3	3 min
	A25 - García et al.		AWS EC2	35	5	< 1 h 30 min
	A33 - Mazo et al.		Generated	5.000	Undefined	1,8 s
EA-based	A01 - Hierons et al.	SIP	Generated	10.000	5 (50K Eval)	< 4 min
	A04 - Henard et al.	SATIBEA	Linux	6.888	5 (~ Eval)	15 min
	A05 - Dou et al.	IGA	Tablet PC	10	1 (50 Eval)	1 min 9 s
	A11 - Pascual et al.	NSGA-II	Generated	5.000	3 (5K Eval)	17,9 s
	A12 - Tan et al.	IBEA	E-Shop	290	5 (25K Eval.)	6,9 s
	A13 - Lian et al.	IVEA	E-Shop	290	5 (50K Eval)	2,95 min
	A16 - Bagheri et al.	GA	Generated	10.000	1 (500 Eval)	≤ 0,5 s
	A19 - Olaechea et al.	IBEA	E-Shop	290	4 (≤ 250K Eval)	< 1.000 h
	A24 - Sayyad et al.	IBEA	E-Shop	290	≤ 5 (2M Eval)	8 min
	A45 - Yeh et al.	GA	Weighing Scales	133	1 (5K Eval)	0,26 s
Other	A14 - Asadi et al.	AHP	Generated	100	6 NFPs	1,43 min
	A29 - Olaechea et al.	Relational Modelling	LinkedList	18	3	4,6 s
	A42 - Shi et al.	Knapsack	Generated	100	3	0,7 s
	A43 - Siegmund et al.	Ad-hoc Algorithm	Berkeley DB	36	2	< 50 min

FM: feature model, F: features, CC: configuration constraints, ET: execution time.

Nevertheless, other constraints like hard limits and filtering are barely managed. AHP approaches are useful specially for supporting stakeholders' preferences. MM-based approaches are good for supporting CMCs given the need to map elements among different models. Other approaches that could not be classified in any of the mentioned methods, support different configuration constraints and a generalization cannot be made.

5. Evaluation process

In this section we identify how selected configuration approaches are evaluated (*cf.* RQ2). Section 5.1 describes usual evaluation techniques conducted in the literature. Section 5.2 shows performance and scalability tests performed in different PLs. Section 5.3 presents performance metrics employed by some of the selected studies and highlights their relation to the used methods and algorithms.

5.1. Evaluation techniques

Most approaches use a *case study* or *performance and scalability tests* to evaluate their solution. From the 48 main selected primary studies, 26 studies use a *case study* evaluation, 20 studies use *performance and scalability tests*, and 2 studies use other techniques such as *prototyping* and *usability tests*. Nonetheless, most case study evaluations only present running examples used to explain the proposed techniques.

Fig. 8 shows the percentage of studies per method that employ any of the aforementioned evaluation techniques.

ILP and MM-based approaches only employ case studies in their

corresponding evaluations, and 80% of AHP-based studies also employ this technique. CP and EA-based approaches, as well as some of the studies that use other unclassified methods (*i.e.* relational modelling (Olaechea et al., 2012), knapsack-based approach (Shi et al., 2010), and a ad-hoc algorithm (Siegmund et al., 2008)), have a high use of performance and scalability tests (58%, 69%, and 43%, respectively). Lastly, other evaluation techniques such as mathematical proofs are used in approaches based on EA and other methods (8% and 14%, respectively). These results show an interest on achieving performance and scalability when configuring a PL.

5.2. Performance and scalability results

In this section, we consider the approaches that employ performance and scalability tests as evaluation techniques. We aim at presenting the execution time taken by each solution to configure a PL, considering the number of configuration constraints, when they are described, and the size of the FM. In cases where more than one PL was tested, we consider only the FM with the highest number of features. This analysis is shown for each of the considered methods, except for ILP and MM-based approaches given the lack of performance and scalability tests during their evaluations (*cf.* Fig. 8).

Table 8 presents the approaches that employ performance and scalability tests. The acronyms of each column title are defined as follows: *feature model* (FM), number of *features* (#F), number of *configuration constraints defined in a configuration scenario* (#CC), and *execution time* during PL configuration (ET). Depending on the approach, additional data will be added in order to identify the particularities of the evaluation (*e.g.* Gamez et al. (2015) consider features as tasks in a workflow and define a number of internal operations, not of configuration constraints).

The PL size of *CP-based approaches* oscillates between 35 and 5.000 features. However, the PLs with the biggest size are toy examples automatically generated by an external tool. The number of configuration constraints vary from 1 to 5, and some approaches like the ones presented by Gamez et al. (2015) and White et al. (2014) present different concerns considered during the evaluation of the approaches (*i.e.* internal operations (Gamez et al., 2015) and steps in the staged-configuration White et al., 2014). Depending on the considered constraints and the size of the PL, the execution time varies. For all cases, solutions are obtained in at least 1,8 s (Mazo et al., 2012) and at most 1 h 30 min (García-Galán et al., 2013). However, if the complexity and size of the FM increases or the number of considered configuration constraints is higher, these values can be drastically affected. Although toy examples are important to give a sense of the approach performance, the results cannot be generalized to real-world scenarios. In fact, for the unique real FM based on AWS EC2 service that has a total of 35 features and 5 configuration constraints, the execution time was around 1 h 30 min (García-Galán et al., 2013). This highlights the need to use large real-world PLs in future work to prove the feasibility of the existing approaches.

EA-based approaches are characterized by the employed algorithm (*e.g.* SIP, IBEA) and by the number of considered evaluations (Eval). The nature of performance and scalability tests change from one approach to another. On the one hand, a particular GA is compared against other tree-searching methods like *Breadth-First Search* (BFS), *Uniform-Cost Search* (UCS), *Depth-First Search* (DFS), and *Iterative Deepening A** (IDA*) (Yeh and Wu, 2005). On the other hand, the performance of the solution is evaluated and compared among different EAs, such as IBEA, NSGA-II, and MOCHC (Hierons et al., 2016; Lian and Zhang, 2015; Pascual et al., 2015; Sayyad et al., 2013c; Tan et al., 2015). One identified advantage related to EA-based approaches correspond to the employment of the same or similar FMs. To guide future work, in addition to large real-world PLs, we recommend the use of the state-of-the-art PLs, which allows the comparison of results presented by different approaches. For instance, the *E-Shop* FM is used by most of the studies.

FMs size of EA-based approaches varies between 10 and 10.000 features, and the considered configuration constraints (usually SOO or MOO) oscillates between 1 and 5 objectives. As in CP-based approaches, the two bigger PLs are toy examples automatically generated by an external tool. The number of features and configuration constraints, and specially the number of evaluations, directly affect the execution time of the solution. For most cases the execution time varies between 0,26s (Yeh and Wu, 2005) and 15 min (Henard et al., 2015), except for the (Olaechea et al., 2014) approach that takes more than 1.000 h to configure a PL where 4 optimization objectives and 250.000 evaluations are considered. Although performance is relevant for all approaches, in most cases the fostering of stakeholders' needs represented as optimization objectives is preferred. That is why some approaches suggest the use of IBEA-based solutions (Olaechea et al., 2014; Sayyad et al., 2013c; Tan et al., 2015), including SATIBEA (Henard et al., 2015) and IVEA (Lian and Zhang, 2015). These solutions are enhanced with other methods like the ones presented in SIP (Hierons et al., 2016) and the feedback-directed operations (Tan et al., 2015).

For AHP-based approaches, just (Asadi et al., 2014) approach present performance and scalability tests. In this study FMs with varying sizes were generated, considering 50, 100, 150, and 200 features. For FMs with 100 features the execution time took 3,41 s with 1 NFP, 20,1 s with 2 NFPs, 56,5 s with 4 NFPs, and 1,43 min with 6 NFPs. The considered execution time corresponds to the adoption of the HTN planner technique; the employed time during the AHP ranking is not considered in these results. As shown in the table, the performance during this phase is quite good and acceptable in contexts where low latencies are expected. However, the ranking phase which is manually performed could be a tedious and time-consuming task, becoming a bottleneck in the PL configuration process.

Finally, we consider the three remaining studies that employed other methods such as relational modelling (Olaechea et al., 2012), knapsack algorithm and FCF (Shi et al., 2010), and an ad-hoc algorithm (Siegmund et al., 2008). The ad-hoc algorithm proposed by (Siegmund et al., 2008) presents the highest execution time when configuring a PL with 36 features. This time can be decreased if hard limits are incorporated as configuration constraints. For FMs with 85 or more features, no answer is generated. The knapsack-based approach (Shi et al., 2010) has the best performance when considering a FM with 100 features; execution time corresponds to 0,7 s. Hard limits can be added to costs and values related to features, and a SOO is also defined to specify the knapsack problem. Lastly, the relational modelling approach presented by (Olaechea et al., 2012) configures a PL of 18 features in around 4,6s, while considering up to 3 optimization objectives. An interesting future work would be the evaluation of the listed existing approaches by using real-world PLs to create new benchmarks.

5.3. Performance and quality metrics in product line configuration

In this section we list and describe some of the most common performance and quality metrics employed by the selected studies. Section 5.3.1 describes the employed metrics with their corresponding definition. Section 5.3.2 briefly presents which metrics are employed by the selected approaches.

5.3.1. Performance and quality metrics

This section presents performance and quality metrics employed by the studied approaches.

Coverage is a measure defined as the set of points of the true Pareto front P included in the approximate Pareto front P' (*cf.* Eq. 3) (Olaechea et al., 2014).

$$Coverage(P') = \frac{|P \cap P'|}{|P|} \tag{3}$$

Table 9
Performance metrics in EA-based approaches.

Approach ID	HV	Spread	GD	C %	Epsilon	ET	Others
A01 - Hierons et al.	✓			✓		✓	✓
A04 - Henard et al.	✓	✓	✓		✓		✓
A05 - Dou et al.							✓
A11 - Pascual et al.	✓		✓			✓	
A12 - Tanet al.	✓			✓		✓	
A13 - Lian et al.	✓	✓	✓		✓	✓	
A16 - Bagheri et al.						✓	
A19 - Olaechea et al.	✓					✓	✓
A24 - Sayyad et al.	✓	✓		✓		✓	
A28 - Karimpour et al.							✓
A41 - Hong et al.							✓
A45 - Yeh et al.							✓
Total	*7*	*3*	*3*	*3*	*2*	*7*	*7*

HV: hypervolume, GD: generational distance, C%: percentage of correct solutions, ET: execution time.

Execution time is a metric that measures the time taken by an approach to configure a PL by considering a set of constraints such as integrity and configuration constraints.

Hypervolume (HV) is a measure that estimates how close a set of generated points in a *performance space* are to the true *Pareto front* (Sayyad et al., 2013c). It is the ratio among the hypervolume of the approximate Pareto front P' and the hypervolume of the true Pareto front P (Olaechea et al., 2014). Eq. 4 shows how to estimate HV, where h_i and h_j are hypercubes computed among the *reference point* for the i^{th} and j^{th} configurations in P' and P, respectively ($i, j \in \mathbb{N}$). Higher values of the indicator are desired (Pascual et al., 2015).

$$HV(P') = \frac{volume(\bigcup_{i=1}^{|P'|} h_i)}{volume(\bigcup_{j=1}^{|P|} h_j)} \tag{4}$$

Spread is a measure that estimates the scattering of a set of solutions (Deb et al., 2002; Sayyad et al., 2013c). Eq. 5 shows the spread calculation as proposed by Deb et al. (2002), where d_0 and d_N are the Euclidean distances from the extreme solutions (*i.e.* first and last position in the Pareto front) to the boundary solutions of the set (*i.e.* d_1 and d_{N-1} respectively), d_i represents the Euclidean distance among two consecutive solutions (*i.e.* the i^{th} solution and the $i+1$th solution), $\overline{d}$ is the average distance of distances d_i such that $i \in [1, N-2]$, and N is the number of obtained solutions including the extreme values. A higher value denotes a more diverse solution space, where solutions are distributed among the different optimization objectives (Henard et al., 2015).

$$Spread = \frac{d_0 + d_N + \sum_{i=1}^{N-2} d_i - \overline{d}}{d_0 + d_N + (N-2)\overline{d}} \tag{5}$$

Generational distance (GD) (Lian and Zhang, 2015; Pascual et al., 2015) is an indicator that measures the quality of the obtained results, by computing the distance between the set of derived solutions P' from the true Pareto front P. Eq. 6 calculates *GD*, where d_i is the Euclidean distance from the i^{th} configuration in P' to the nearest solution in P (Pascual et al., 2015). Lower values are desired, meaning that the distance from P' to P is shorter (*i.e.* if $GD(P') = 0$ then $P' = P$) (Pascual et al., 2015).

$$GD(P') = \frac{1}{|P'|}\sqrt{\sum_{i=1}^{|P'|} d_i^2} \tag{6}$$

Pareto front size (PFS) is a metric that measures the number of solutions found in the Pareto front; it corresponds to the cardinality of the Pareto front ($PFS = |P|$) (Henard et al., 2015). A higher value of PFS is preferred (Henard et al., 2015). However, if too many solutions are offered to a decision maker the selection of the most suitable one could turn into an overwhelming task. Then, other mechanisms should be implemented to truncate the number of solutions.

Percentage of correct solutions is a metric that represents the ratio of valid and complete configurations; i.e. solutions that respond to both integrity and configuration constraints. A higher value of this metric is desired (Sayyad et al., 2013c).

Epsilon (ϵ) indicates the distance between the approximate Pareto front P' and the true Pareto front P. In this case a lower value of ϵ is preferred, showing that P' is closer to P (Henard et al., 2015; Lian and Zhang, 2015).

Consistency ratio (CR) estimates comparison conflicts among factors in a matrix. A CR below 0.1 is desired (Tan et al., 2014). In Eq. 7, λ_{max} is the principal eigenvalue and $\overline{\lambda_{max}}$ is an average eigenvalue obtained from random matrices (Alonso and Lamata, 2006).

$$CR = \frac{\lambda_{max} - n}{\overline{\lambda_{max}} - n} \tag{7}$$

Optimization rate is a measure that estimates the ratio between the value obtained in the approximate answer C and the value obtained in the optimal answer C_O. The optimization rate is calculated as defined in Eq. 8, where $v(C)$ is a function that estimates the value of C, and $v(C_O)$ estimates the value of C_O (Shi et al., 2010).

$$OR(C) = \frac{v(C)}{v(C_O)} \tag{8}$$

5.3.2. Employed metrics

Based on the employed method some of the previously listed metrics are preferred over others. However, in almost all cases without considering the used method, *execution time* metric is considered in the evaluation results. Pareto front-based metrics are considerably important in approaches that perform MOO. Hereafter, we explain which metrics are preferred by each method. Notice that 20 of the 48 analysed approaches do not consider any performance or quality metrics. Mainly ILP and MM-based methods are excluded from this analysis because no performance and scalability tests are employed in their evaluations.

In the case of CP-based approaches, 100% of the selected studies consider the *execution time* metric as the main measure to evaluate their solution. Nevertheless, quality aspects of the obtained solutions during PL configuration are tested just by considering the aggregated values of the NFPs related to the selected features. Additional metrics should be considered to verify the validity and satisfaction degree of the configured products in future research. Furthermore, EA-based approaches consider a higher variety of performance metrics. Table 9 shows the relation among these studies and some of the most recurrent employed metrics.

Most approaches rely on *HV* and *execution time* metrics. Other used metrics correspond to *spread* (Henard et al., 2015; Lian and Zhang, 2015; Sayyad et al., 2013c), *GD* (Henard et al., 2015; Lian and Zhang, 2015; Pascual et al., 2015), and *percentage of correct solutions* (Hierons et al., 2016; Sayyad et al., 2013c; Tan et al., 2015). Some particular metrics like *epsilon* (Henard et al., 2015; Lian and Zhang, 2015) and *PFS* (Henard et al., 2015) are less used. The diversity of metrics used by the approaches of this group can be attributed to the inability to guarantee the satisfaction of both integrity and configuration constraints; different aspects of the derived solutions are evaluated to test their quality.

When considering AHP-based approaches three main metrics are used: *execution time* (Asadi et al., 2014), *number of comparisons* (Tan et al., 2014), and *consistency ratio* (Tan et al., 2014; Zhang et al., 2014). The last two metrics are inherent to the AHP nature where pairwise comparisons are performed and conflicts among comparisons can arise.

Finally, for other approaches the most used metric is *execution time*

(Olaechea et al., 2012; Shi et al., 2010; Siegmund et al., 2008). In the case of recommender techniques (Bagheri and Ensan, 2014a) the *rank order* of the solutions and *usability* metrics are considered. The relational modelling (Olaechea et al., 2012) approach considers both the *execution time* and the *PFS*. Additionally, the knapsack-based approach (Shi et al., 2010) considers the *execution time* and the *optimization rate* metrics. Lastly, the ad-hoc algorithm solution presented by Siegmund et al. (2008) only considers the *execution time* metric.

To conclude, different performance and quality metrics are used along the selected studies. However, EA-based approaches are the ones that present the largest variety of metrics (*e.g.* HV, spread, GD). This may be related to their inability of guaranteeing constraints satisfaction and procuring global optimal solutions. For the remaining methods, execution time is essential for measuring the scalability and performance of the approach. Other less used metrics such as rank order, optimization rate, and PFS are included in some of the selected approaches.

6. Open challenges

This section presents open challenges and gaps identified in the SLR selected studies (*cf.* RQ3). They will be listed in two separate groups, one addressing the configuration techniques gaps, and other addressing the challenges related to the evaluation process.

First, we identify the open challenges related to the employed techniques.

Development of hybrid solutions. Each configuration technique offers a set of advantages and disadvantages that affect the configuration process and result. Different configuration constraints are supported by each technique (*cf.* Section 5.2). Moreover, performance is favoured in some cases (*e.g.* EA-based approaches), while in others performance and scalability have critical issues (*e.g.* CP-based approaches). Furthermore, some approaches cannot guarantee the satisfaction of constraints (*e.g.* EA-based approaches), while others demand their compliance (*e.g.* CP-based approaches). These reasons demand the implementation of hybrid solutions that consider the strengths presented by each different technique. For example, a CP-based approach could be used to derive a small set of valid configurations, which can be used as the initial population of an EA-based approach. This action could enhance the quality of the obtained configurations without deeply affecting the performance of the solution.

Tuning configuration techniques and algorithms. In the case of large and complex models, the tuning of the configuration techniques and algorithms is required to improve performance when deriving solutions. As it has been acknowledged by some of the selected studies from our SLR (Henard et al., 2015; Sayyad et al., 2013b; 2013c), tuning algorithms parameters can improve the obtained results. Moreover, heuristics can be employed to outperformed previous results, as it is the case of (Mazo et al., 2014) where the performance of a PL configuration approach based on CP was improved by defining search heuristics. Given that the semi-automatic configuration of large PLs is still a challenge, there is a big opportunity to use such heuristics for improving performance results.

Exploring new techniques and algorithms. New techniques should be explored, such as the implementation of ad-hoc algorithms (Sánchez et al., 2014; Siegmund et al., 2008) and recommender algorithms (Bagheri and Ensan, 2014a). For this last alternative, new studies have emerged as the one proposed by (Pereira et al., 2016), where a recommender system is employed to guide the configuration of PLs by considering historical data related to previous configurations. Nevertheless, this approach does not support extended PLs. Thus, as future work, we suggest the use of context-aware recommendation techniques to incorporates NFPs and the exploration of various types of statistical tests to identify which of the contextual NFPs are truly significant in the sense that they indeed affect the recommendations. Also, we suggest evaluating the proposed approach under the use of other recommendation algorithms on self-adaptive systems. Finally, we also recommend analysing the impact of the proposed algorithms on configuration performance and the conduction of a user-controlled study to investigate user's satisfaction.

Considering new configuration constraints. Future work should consider configuration constraints that are not yet contemplated by current approaches, in order to allow the specification of diverse stakeholders' requirements. For instance, CP and ILP-based approaches can employ MOO and fuzzy requirements; EA-based studies can explore hard limits, finite domain, CMC, EFC, and fuzzy requirements; AHP-based approaches can also support other configuration constraints that are not only focused in preferences and weights; and MM-based solutions can implement SOO, MOO, finite domain, filtering, and EFC constraints (*cf.* Section 4.2); and recommendation techniques can use previous configurations to suggest feature selection. Therefore, the implementation of a new approach that combines several techniques is important to consider the diversity of stakeholder's requirements.

Supporting multi-stakeholder environments. Although staged configurations (Czarnecki et al., 2004; Siegmund et al., 2008) as well as different views (Lee et al., 2014; Ostrosi et al., 2012; Schroeter et al., 2012) are provided by some techniques to deal with multi-stakeholder environments, just 17% of the selected studies from our SLR implement those solutions. In these approaches, conflicts among stakeholders (Ochoa et al., 2015) and responsibilities may arise. Thus, future research is needed in this field to derive products with a higher satisfaction degree for a set of different decision makers. As an example, (Martinez et al., 2014) present a visualisation paradigm called FRoGs (*Feature Relations Graphs*). This solution supports different decision makers to obtain a better understanding of feature constraints during the product configuration process. However, FRoGs does not support the configuration process of extended PLs.

Management of qualitative NFPs. Quantitative NFPs are managed by most approaches. However, few solutions support qualitative NFPs. According to two systematic literature reviews (Galster et al., 2014; Mahdavi-Hezavehi et al., 2013) current methods focus on performance and availability NFPs. Moreover, in most cases NFPs are managed as nominal values. Although fuzzy logic and MM-based solutions (*cf.* Section 4) have been adopted to manage qualitative NFPs, further research is still required in this field. Current studies do not provide enough evidence for practitioners to apply the proposed approaches to handle variability. Therefore, we suggest industry studies to prove the feasibility of current approaches.

Reconfiguration in dynamic environments. When environments represented by PLs change, both variability models and configurations should embrace these changes to satisfy product requirements and stakeholders' needs. Thus, dynamic reconfiguration during runtime is needed in order to maintain validity and completeness of the derived products. Currently, most approaches are developed for static models and configuration constraints, where configurations are strictly tied to human actions and not to changing contexts. However, a new research problem related to dynamic PLs has born to face this issue and is a well-known challenge in the PL field (Capilla et al., 2014). In this scenario, we suggest the use of context-aware techniques to support the runtime configuration of adaptive systems.

Henceforth, we identify the open challenges related to the evaluation process used by the selected studies.

Evaluation technique. Most of the analysed approaches present a running example that explains how the proposed configuration technique works, without presenting a further evaluation (*i.e.* performance and scalability tests) (Chen and Babar, 2011; El Yamany et al., 2014;

Ghezzi and Molzam Sharifloo, 2013; Lee et al., 2014). In addition, most evaluations do not rely on real-world FMs (Asadi et al., 2014; Shi et al., 2010; White et al., 2010; 2014). This supposes a threat to validity in current researches. Thus, to support the studies validity, features and attributes should be modelled based on real data from business experience. Moreover, quality metrics that measure the obtained results, as the ones employed by EA-based solutions in Section 5.3.2, are required to prove the validity of the proposed technique and to allow the improvement of the results (*i.e.* satisfaction degree of stakeholders with regards to a given configuration). Finally, it is also important to further explore the adoption of user studies.

Publish evaluation inputs and results. Although some approaches performed a rigorous evaluation of their solution; the evaluated FMs and their characteristics, as well as the obtained results were not made publicly available. Consequently, the reproduction of the experiments in future work is no longer possible.

Tool support. Although there are available tools (Pereira et al., 2014; Bashroush et al., 2017) to partially assist decision makers during the semi-automatic configuration process of extended PLs (*e.g.* VariaMos (Mazo et al., 2012), jUCMNav (Mussbacher et al., 2012), Kumbang (Myllärniemi et al., 2015), MO-DAGAME (Pascual et al., 2015), SALOON (Quinton et al., 2013), PuMA (Schroeter et al., 2012), SPL Conqueror (Siegmund et al., 2012b), and Ascent Desigh Studio and FaMa (White et al., 2014)), those tools do not offer support to the whole set of the SLR identified configuration constraints and there is also a lack of visualisation mechanisms. For instance, user guidance is needed during the selection of one configuration from a set of possible results, all of them retrieved by a multi-optimization tool. Possibly, this could be achieved through the use of visualization mechanisms, recommendation techniques, and explicit data from stakeholders (*e.g.* historical configurations (Pereira et al., 2016)). Moreover, it was also found that a user study that compares those tools and prove their practical support on real-world scenarios is still missing.

As future work we aim to address some of the identified challenges related to the extended PL configuration process. First, we will further explore trending techniques such as CP and EA, tuning parameters, and introducing new configuration constraints and heuristics that can enhance the current results. Second, we aim at extending the approach presented by (Pereira et al., 2016) to support the semi-automatic configuration of extended PLs. Finally, we will continue developing tool support that offers most of the identified contributions, as well as visualisation mechanisms to ease the configuration process.

7. Threats to validity

This section discusses potential *threats to validity* that might have affected the results of the SLR. We faced similar threats to validity as any other SLR. The findings of this SLR may have been affected by bias in the selection of the primary studies, inaccuracy in the data extraction and in the classification of the primary studies, and incompleteness in defining the open challenges. Next, we summarize the main threats to validity of our work, and the strategies we have followed to minimize the likelihood of their realization and impact. We discussed the SLR validity with regards to its *internal* and *external* validity (Wohlin et al., 2000).

Internal validity. Internal validity threats are related with the reliability of the selection and data extraction processes. To further increase the validity of the review results, we conducted the inclusion and exclusion processes in parallel by involving three researchers and we cross-checked the outcome after each phase. In the case of disagreements, we discussed until a final decision was achieved. Arguments related to the exclusion of potentially relevant studies were documented. However, the outcome of the selection process (*i.e.* selected and excluded articles) could vary if it is replicated by other researchers.

For selected papers, a potential threat to validity is the reliability and accuracy of the data extraction process, since not all information was trivial to extract (*i.e.* many papers lacked details about the design and execution of the reported study). Consequently, some data had to be interpreted which involved subjective decisions. Therefore, to ensure the validity, multiple sources of data were analysed; i.e. papers, websites, technical reports, manuals, and executable artefacts. Moreover, whenever there was doubt regarding extracted data of a particular paper, we discussed the reported information from different perspectives in order to solve all discrepancies. However, we are aware that the data extraction process is a subjective activity and likely to yield different results when executed by different researchers. Thus, we took the following steps to address this threat. First, we created a spreadsheet to collect the data and a guideline document to define each term on the spreadsheet. Second, we split the research team into two groups for the extraction task. Third, the two teams performed the classification independently. Fourth, one team checked the extraction task from another team. Fifth, we held a meeting to discuss the non-agreements until a consensus was reached. Therefore, in general we believe that the overall validity of the selection process is high given the use of a systematic procedure and involvement of expert researchers in the field.

External validity. The restriction of considering articles published only after 2000 was a major external validity of this study. Although it can affect the completeness of our search results, according to (Chen and Babar, 2011) only a small number of papers have been published in the SPL domain before 2000. To further decrease the probability of missing relevant papers, the search for relevant studies relied on several relevant scientific databases, and it considered not only on journals but also on conferences and workshops. In addition, to have a complete set of studies, we also analysed the references of the primary studies to identify other relevant approaches. Nevertheless, the quality of the search engines could have influenced the completeness of the SLR primary studies set. This means that our search may have missed studies whose authors considered different terms to specify the PL configuration process (*i.e.* studies that did not use the SLR keywords). To minimize these limitations and to support the review replicability, this SLR followed a strict protocol described in Section 3. The protocol was discussed before the start of the review to increase the reliability of the selection and data extraction processes.

Reviewers reliability and definition of open challenges are also included as external validity threats. The four reviewers are researchers in the software engineering field, and none of the selected primary studies was written by all of the four authors. In the case of a paper authored by one or more of the authors of this paper, the decision of how to categorize the study was conducted by the remaining authors. Therefore, we are not aware of any bias introduced during the analysis. Additionally, it is possible that the definition of open challenges was affected by the authors personal interest and opinions. We are aware that the completeness of open challenges is another limitation that should be considered while interpreting the results of this review; additional open challenges not highlighted in this work should be included in future reviews.

Finally, although we focused only on FMs, which are considered as the de facto variability modelling, other approaches (*e.g.* OVM, decision models, and CVL) are also important in the literature of PLs. Still, analysing such studies remains as an important future work task.

8. Conclusions and future work

This paper presents a SLR that addresses the configuration of extended PLs, given the need to conduct a secondary source that provides an overview of the research made in this field. We selected 66 articles for a deep analysis, where 48 of them are considered as main studies. We identified an increasing trend in PL configuration research from 2010 up to 2016. Hereafter, we describe the three research questions addressed by this review and their corresponding results.

RQ1 focuses on the identification of techniques used by selected studies to support the semi-automatic configuration of extended PLs. EA and CP are the most popular techniques among the studied approaches. Other algorithms and techniques such as AHP, ILP, and MM are also employed by the analysed studies. Some studies could not be classified in the aforementioned techniques. This unclassified group considers other techniques such as ASP (Myllärniemi et al., 2015), knapsack-based solutions (Shi et al., 2010), ad-hoc (Sánchez et al., 2014; Siegmund et al., 2008) and recommender algorithms (Bagheri and Ensan, 2014b), and relational modelling (Olaechea et al., 2012), among others. Furthermore, we extracted a list of stakeholders' requirements supported by those approaches, such as SOO, MOO, hard limits, preferences and weights, finite domain, filter, CMC, EFC, seeding, and fuzzy requirements. Depending on the employed method some configuration constraints are better supported than others (*cf.* Section 4.2).

RQ2 considers the evaluation process of the selected primary studies. Most studies conduct a case study, or performance and/or scalability tests to prove the approach validity. CP and EA-based approaches are the ones with the highest interest on performance and scalability support; FMs with up to 10.000 features are tested under different configuration scenarios. However, better performance and scalability results are achieved by EA-based approaches. In addition, performance and quality metrics are considered by more than half of the studies. In the case of EA-based approaches quality metrics such as HV, spread, and GD are also computed given their impossibility to guarantee constraints satisfaction.

Finally, RQ3 supports the identification of open challenges and gaps in the field of extended PLs configuration. Challenges related to configuration techniques consider the exploration of hybrid solutions and new techniques and algorithms that can better support the PL configuration needs. Moreover, tuning current algorithms parameters, as well as introducing new heuristics offer an improvement in the existing configuration alternatives. Furthermore, supporting new configuration constraints, multi-stakeholder environments, qualitative NFPs, and PL reconfiguration dynamic contexts are additional gaps to be faced by researchers. Besides, when facing approaches evaluation, the application of the selected evaluation technique should consider existent and rigorous methodologies as well as metrics that increase the study validity. Real-world PLs should also be considered in order to decrease threats to validity. Finally, the evaluation data and developed tools must be publicly available to guide future research in this field.

As future work, we plan to conduct a study with experts in the PL configuration field to rank the importance of the open challenges described in Section 6. Once they are ranked, we aim at facing the most relevant ones. Lastly, we consider that a benchmark that compares the performance and scalability of existing techniques when configuring extended PLs is required.

Acknowledgement

Juliana Alves Pereira is supported by the Brazilian National Council for Scientific and Technological Development (CNPq) grant 202368/2014-9.

Appendix A. Databases Search Queries

In this appendix, we present the particular search process and filters applied to each one of the five database engines. In all cases, a query was generated per combination of keywords. For instance, in the case of *product line, configuration*, and *attribute*, the resulting search string corresponds to *product AND line AND configuration AND attribute*. We use these search strings to guarantee homogeneity in the search process among different databases, due to limitations related to boolean operators in some of the selected engines. However, there are some filtering decisions that varied among databases. In the case of the *ACM Digital Library*, items were selected from *the ACM full-text collection*. Keywords were searched in *any field* of the article metadata, and we defined that *all* words should be found in that content. In addition, the *publication year* was set to be greater than or equal to *2000*.

In the *Science Direct* and *Scopus* database engines, the search string is searched in the *abstract, title, and keywords* content. Moreover, the *publication year* was defined from *2000* to *present* date. In both cases, as additional filters, we defined that the search should be done in *journals*, and in the fields of *computer science, decision sciences, engineering*, and *mathematics*. For the case of *Scopus* we defined that articles should be written in *English*, and that the accepted document types correspond to *conference papers, articles*, and *articles in press*.

For the *IEEE Xplore*, the keywords were searched *only in metadata*. The desired content type was related to *conference & publications* and *journals & magazines*. As in the case of *Science Direct*, the *publication year* was defined between *2000* and *present* date. Finally, in the case of *Springer Link all words* in the combined keywords were required to appear in the paper. The *publication date* was defined from *2000* to *2017*; paper language was set to *English*; and *computer science, decision sciences, engineering*, and *mathematics* were defined as *disciplines* of interest.

Appendix B. Selected Articles

Table B.10 and Table B.11 present the 66 studies considered in the current SLR. *Title, year* of publication, and *publication* name are shown for each article.

Table B10
SLR selected articles I.

Title	Year	Publication
SIP: Optimal Product Selection from Feature Models using Many-objective Evolutionary Optimization (Hierons et al., 2016)	2016	ACM TOSEM
Quality-centric Feature Model Configuration Using Goal Models (Noorian et al., 2016)	2016	ACM SAC
Resource Usage Aware Configuration in Software Product Lines (Zanardini et al., 2016)	2016	JLAMP
Combining Multi-Objective Search and Constraint Solving for Configuring Large Software Product Lines (Henard et al., 2015)	2015	ICSE
Customer-oriented Product Collaborative Customization Based on Design Iteration for Tablet Personal Computer Configuration (Dou et al., 2016)	2015	CIE
SPL-TQSSS: A Software Product Line Approach for Stateful Service Selection (Gamez et al., 2015)	2015	ICWS
Multi Attribute Architecture Design Decision for Core Asset Derivation (Halim et al., 2015)	2015	JT
Automating Resource Selection and Configuration in Inter-clouds Through a Software Product Line Method (Leite et al., 2015)	2015	CLOUD
Representing and Configuring Security Variability in Software Product Lines (Myllärniemi et al., 2015)	2015	QoSA
Using Decision Rules for Solving Conflicts in Extended Feature Models (Ochoa et al., 2015)	2015	SLE
Applying Multiobjective Evolutionary Algorithms to Dynamic Software Product Lines for Reconfiguring Mobile Applications (Pascual et al., 2015)	2015	JSS
Optimizing Selection of Competing Features via Feedback-directed Evolutionary Algorithms (Tan et al., 2015)	2015	ISSTA
Optimized Feature Selection Towards Functional and Non-functional Requirements in Software Product Lines (Lian and Zhang, 2015)	2015	SANER
Toward Automated Feature Model Configuration with Optimizing Non-functional Requirements (Asadi et al., 2014)	2014	IST
Dynamic Decision Models for Staged Software Product Line Configuration (Bagheri and Ensan, 2014a)	2014	REJ
Reliability Estimation for Component-based Software Product Lines (Bagheri and Ensan, 2014b)	2014	CJECE
Quality Ranking of Features in Software Product Line Engineering (Tan et al., 2014)	2014	APSEC
A Holistic Approach to Feature Modeling for Product Line Requirements Engineering (Lee et al., 2014)	2014	REJ
An Evolutionary Methodology for Optimized Feature Selection in Software Product Lines (Lian and Zhang, 2014)	2014	SEKE
From Intentions to Decisions: Understanding Stakeholders' Objectives in Software Product Line Configuration (Noorian et al., 2014)	2014	SEKE
Comparison of Exact and Approximate Multi-objective Optimization for Software Product Lines (Olaechea et al., 2014)	2014	SPLC
An Approach for Managing Quality Attributes at Runtime Using Feature Models (Sánchez et al., 2014)	2014	SBCARS
Evolving Feature Model Configurations in Software Product Lines (White et al., 2014)	2014	JSS
OPTI-SELECT: An Interactive Tool for User-in-the-loop Feature Selection in Software Product Lines (El Yamany et al., 2014)	2014	SPLC
Combined Propagation-based Reasoning with Goal and Feature Models (Liu et al., 2014)	2014	MoDRE
Optimum Feature Selection in Software Product Lines: Let Your Model and Values Guide Your Search (Sayyad et al., 2013a)	2013	CMSBSE
Migrating to the Cloud: a Software Product Line based Analysis (García-Galán et al., 2013)	2013	CLOSER
Dealing with Non-Functional Requirements for Adaptive Systems via Dynamic Software Product-Lines (Ghezzi and Molzam Sharifloo, 2013)	2013	SESAS
Product Configuration Method based on Ontology Mapping (Chen et al., 2013)	2013	ICSESS
Bi-criteria Genetic Search for Adding New Features into an Existing Product Line (Karimpour and Ruhe, 2013)	2013	CMSBSE
Visualization and Exploration of Optimal Variants in Product Line Engineering (Murashkin et al., 2013)	2013	SPLC
Towards Multi-cloud Configurations Using Feature Models and Ontologies (Quinton et al., 2013)	2013	MultiCloud
On the Value of User Preferences in Search-based Software Engineering: A Case Study in Software Product Lines (Sayyad et al., 2013c)	2013	ICSE

Table B11
SLR selected articles II.

Title	Year	Publication
Quality Attribute Modeling and Quality Aware Product Configuration in Software Product Lines (Zhang et al., 2014)	2013	SQJ
Formalizing Interactive Staged Feature Model Configuration (Bagheri et al., 2012)	2012	JSEP
Using Feature Modelling and Automations to Select among Cloud Solutions (Quinton et al., 2012)	2012	PLEASE
Constraints: The Heart of Domain and Application Engineering in the Product Lines Engineering Strategy (Mazo et al., 2012)	2012	IJISMD
AoURN-based Modeling and Analysis of Software Product Lines (Mussbacher et al., 2012)	2012	SQJ
Modelling and Multi-objective Optimization of Quality Attributes in Variability-rich Software (Olaechea et al., 2012)	2012	NFPinDSML
A Fuzzy Configuration Multi-agent Approach for Product Family Modelling in Conceptual Design (Ostrosi et al., 2012)	2012	JIM
Dynamic Configuration Management of Cloud-based Applications (Schroeter et al., 2012)	2012	SPLC
SPL Conqueror: Toward Optimization of Non-functional Properties in Software Product Lines (Siegmund et al., 2012b)	2012	SQJ
Interoperability of Non-functional Requirements in Complex Systems (Siegmund et al., 2012a)	2012	SEES
Automated Planning for Feature Model Configuration Based on Functional and Non-functional Requirements (Soltani et al., 2012)	2012	SPLC
Towards Product Configuration Taking into Account Quality Concerns (Villela et al., 2012)	2012	SPLC
Using Knowledge-based Systems to Manage Quality Attributes in Software Product Lines (Zhang et al., 2011)	2011	SPLC
Formalizing Feature Selection Problem in Software Product Lines Using 0–1 Programming (Li et al., 2012)	2011	ISKE
Goal-driven Software Product Line Engineering (Asadi et al., 2011)	2011	ACM SAC
Using Integer Constraint Solving in Reuse Based Requirements Engineering (Salinesi et al., 2010)	2010	RE
Stratified Analytic Hierarchy Process: Prioritization and Selection of Software Features (Bagheri et al., 2010a)	2010	SPLC
Configuring Software Product Line Feature Models Based on Stakeholders' Soft and Hard Requirements (Bagheri et al., 2010b)	2010	SPLC
Rapid Identification of the Optimal Product Configuration and its Parameters Based on Customer-centric Product Modeling for One-of-a-kind Production (Hong et al., 2010b)	2010	CI
Optimal Concurrent Product Design and Process Planning based on the Requirements of Individual Customers in One-of-a-kind Production (Hong et al., 2010a)	2010	IJPR
Automated Diagnosis of Feature Model Configurations (White et al., 2010)	2010	JSS
A Preliminary Experimental Study on Optimal Feature Selection for Product Derivation using Knapsack Approximation (Shi et al., 2010)	2010	PIC
Identification of the Optimal Design and its Production Process for One-of-a-kind Production (Hong et al., 2008a)	2008	AMR
Identification of the Optimal Product Configuration and Parameters based on Individual Customer Requirements on Performance and Costs in One-of-a-kind Production (Hong et al., 2008b)	2008	IJPR
Automatically Composing Reusable Software Components for Mobile Devices (White et al., 2008)	2008	JBCS
Measuring Non-functional Properties in Software Product Lines for Product Derivation (Siegmund et al., 2008)	2008	APSEC
A Customer-driven Approach to One-of-a-kind Product Design (Song et al., 2007)	2007	IDETC/CIE
Automating Product-Line Variant Selection for Mobile Devices (White et al., 2007)	2007	SPLC
Design for Customer Satisfaction in One-of-a-kind Production Environment (Hong et al., 2006)	2006	IDETC/CIE
Web-based Configuration Design System for Product Customization (Ong et al., 2006)	2006	IJPR
Solutions for Product Configuration Management: An Empirical Study (Yeh and Wu, 2005)	2005	AI EDAM
Modelling Platform-based Product Configuration using Programmed Attributed Graph Grammars (Du et al., 2003)	2003	JED
Graph Grammar based Product Family Modeling (Du et al., 2002)	2002	CERA

Appendix C. Publication Sources

This appendix shows all the considered publication sources referring to the *journal* (*cf.* Table C.12), *conference* (*cf.* Table C.13), *workshop* (*cf.* Table C.14), and *symposium* (*cf.* Table C.15) publication types. For each publication type we present a table with the *publication name*, its name *abbreviation*, and the number of *articles* published in that source.

Table C12
SLR articles per journal source.

Publication name	Abbreviation	Articles
ACM Transactions on Software Engineering and Methodology	ACM TOSEM	1
Artificial Intelligence for Engineering Design, Analysis and Manufacturing	AI EDAM	1
Advanced Materials Research	AMR	1
Concurrent Engineering: Research and Applications	CERA	1
Computers in Industry	CI	1
Computers and Industrial Engineering	CIE	1
Canadian Journal of Electrical and Computer Engineering	CJECE	1
International Journal of Information System Modeling and Design	IJISMD	1
International Journal of Production Research	IJPR	3
Information and Software Technology	IST	1
Journal of the Brazilian Computer Society	JBCS	1
Journal of Engineering Design	JED	1
Journal of Intelligent Manufacturing	JIM	1
Journal of Logical and Algebraic Methods in Programming	JLAMP	1
Journal of software: Evolution and Process	JSEP	1
Journal of Systems and Software	JSS	3
Jurnal Teknologi	JT	1
Requirements Engineering Journal	REJ	2
Software Engineering for Self-Adaptive Systems	SESAS	1
Software Quality Journal	SQJ	3
Total		*27*

Table C13
SLR articles per conference source.

Publication name	Abbreviation	Articles
Asia-Pacific Software Engineering Conference	APSEC	2
International Conference on Cloud Computing and Services Science	CLOSER	1
International Conference on Cloud Computing	CLOUD	1
International Conference on Software Engineering	ICSE	2
International Conference on Software Engineering and Service Science	ICSESS	1
International Conference on Web Services	ICWS	1
International Design Engineering Technical Conferences and Computers and Information in Engineering Conference	IDETC/CIE	2
International Conference on Intelligent Systems and Knowledge Engineering	ISKE	1
International Conference on Progress in Informatics and Computing	PIC	1
International Conference on Quality of Software Architectures	QoSA	1
International Requirements Engineering Conference	RE	1
International Conference on Software Analysis, Evolution, and Reengineering	SANER	1
International Conference on Software Engineering and Knowledge Engineering	SEKE	2
International Conference on Software Language Engineering	SLE	1
International Software Product Line Conference	SPLC	10
Total		*28*

Table C14
SLR articles per workshop source.

Publication name	Abbreviation	Articles
International Workshop on Combining Modelling and Search-Based Software Engineering	CMSBSE	2
International Model-Driven Requirements Engineering Workshop	MoDRE	1
International Workshop on Multi-Cloud Applications and Federated Clouds	MultiCloud	1
International Workshop on Nonfunctional System Properties in Domain Specific Modeling Languages	NFPinDSML	1
International Workshop on Product Line Approaches in Software Engineering	PLEASE	1
International Workshop on Software Engineering for Embedded Systems	SEES	1
Total		*7*

Table C15
SLR articles per symposium source.

Publication name	Abbreviation	Articles
ACM Symposium on Applied Computing	ACM SAC	2
International Symposium on Software Testing and Analysis	ISSTA	1
Brazilian Symposium on Software Components, Architectures and Reuse	SBCAR	1
Total		*4*

References

Afzal, U., Mahmood, T., Shaikh, Z., 2016. Intelligent software product line configurations. Comput. Stand. Interfaces 48, 30–48. https://doi.org/10.1016/j.csi.2016.03.003.

Alonso, J.A., Lamata, M.T., 2006. Consistency in the analytic hierarchy process: a new approach. Int. J. Uncertain. Fuzziness Knowl. Syst. 14 (04), 445–459. https://doi.org/10.1142/S0218488506004114.

Alves, V., Niu, N., Alves, C., Valença, G., 2010. Requirements engineering for software product lines: a systematic literature review. Inf. Softw. Technol. 52 (8), 806–820. https://doi.org/10.1016/j.infsof.2010.03.014.

Apt, K., 2003. Principles of Constraint Programming. Cambridge University Press, New York.

Asadi, M., Bagheri, E., Gašević, D., Hatala, M., Mohabbati, B., 2011. Goal-driven software product line engineering. Proceedings of the ACM Symposium on Applied Computing. ACM, New York, pp. 691–698. https://doi.org/10.1145/1982185.1982336.

Asadi, M., Soltani, S., Gasevic, D., Hatala, M., Bagheri, E., 2014. Toward automated feature model configuration with optimizing non-functional requirements. Inf. Softw. Technol. 56 (9), 1144–1165. https://doi.org/10.1016/j.infsof.2014.03.005.

Bagheri, E., Asadi, M., Gasevic, D., Soltani, S., 2010a. Stratified analytic hierarchy process: prioritization and selection of software features. Proceedings of the 14th International Conference on Software Product Lines. Springer, Berlin, Heidelberg, pp. 300–315. https://doi.org/10.1007/978-3-642-15579-6_21.

Bagheri, E., Di Noia, T., Ragone, A., Gasevic, D., 2010b. Configuring software product line feature models based on stakeholders' soft and hard requirements. Proceedings of the 14th International Conference on Software Product Lines. Springer, Berlin, Heidelberg, pp. 16–31. https://doi.org/10.1007/978-3-642-15579-6_2.

Bagheri, E., Ensan, F., 2014a. Dynamic decision models for staged software product line configuration. Requir. Eng. 19 (2), 187–212. https://doi.org/10.1007/s00766-013-0165-8.

Bagheri, E., Ensan, F., 2014b. Reliability estimation for component-based software product lines. Can. J. Electr. Comput. Eng. 37 (2), 94–112. https://doi.org/10.1109/CJECE.2014.2323958.

Bagheri, E., Noia, T.D., Gasevic, D., Ragone, A., 2012. Formalizing interactive staged feature model configuration. J. Softw.: Evol. Process 24 (4), 375–400. https://doi.org/10.1002/smr.534.

Bak, K., Czarnecki, K., Wasowski, A., 2011. Feature and meta-models in clafer: mixed, specialized, and coupled. Proceedings of the 3rd International Conference on Software Language Engineering. Springer, Berlin, Heidelberg, pp. 102–122. https://doi.org/10.1007/978-3-642-19440-5_7.

Bashroush, R., Garba, M., Rabiser, R., Groher, I., Botterweck, G., 2017. CASE tool support for variability management in software product lines. ACM Comput. Surv. 50 (1), 14:1–14:45. https://doi.org/10.1145/3034827.

Batory, D., 2005. Feature models, grammars, and propositional formulas. Proceedings of the 9th International Conference on Software Product Lines. Springer, Berlin, Heidelberg, pp. 7–20. https://doi.org/10.1007/11554844_3.

Benavides, D., Segura, S., Ruiz-Cortés, A., 2010. Automated analysis of feature models 20 years later: a literature review. Inf. Syst. 35 (6), 615–636. https://doi.org/10.1016/j.is.2010.01.001.

Benavides, D., Trinidad, P., Ruiz-Cortés, A., 2005. Automated reasoning on feature models. Proceedings of the 17th International Conference on Advanced Information Systems Engineering. Springer, Berlin, Heidelberg, pp. 491–503. https://doi.org/10.1007/11431855_34.

Brereton, P., Kitchenham, B.A., Budgen, D., Turner, M., Khalil, M., 2007. Lessons from applying the systematic literature review process within the software engineering domain. J. Syst. Softw. 80 (4), 571–583. https://doi.org/10.1016/j.jss.2006.07.009.

Capilla, R., Bosch, J., Trinidad, P., Ruiz-Cortés, A., Hinchey, M., 2014. An overview of dynamic software product line architectures and techniques: observations from research and industry. J. Syst. Softw. 91, 3–23. https://doi.org/10.1016/j.jss.2013.12.038.

Chavarriaga, J., Noguera, C., Casallas, R., Jonckers, V., 2014. Propagating decisions to detect and explain conflicts in a multi-step configuration process. In: Dingel, J., Schulte, W., Ramos, I., Abraho, S., Insfran, E. (Eds.), Proceedings of the MODELS. Springer, Cham, pp. 337–352. https://doi.org/10.1007/978-3-319-11653-2_21.

Chen, J., Cheng, Y., Nie, D., 2013. Product configuration method based on ontology mapping. Proceedings of the 4th International. Conference on Software Engineering and Service Science. IEEE, pp. 97–101. https://doi.org/10.1109/ICSESS.2013.6615264.

Chen, L., Ali Babar, M., Ali, N., 2009. Variability management in software product lines: a systematic review. Proceedings of the 13th Int. Software Product Line Conference. Carnegie Mellon University, Pittsburgh, pp. 81–90.

Chen, L., Babar, M.A., 2011. A systematic review of evaluation of variability management approaches in software product lines. Inf. Softw. Technol. 53 (4), 344–362. https://doi.org/10.1016/j.infsof.2010.12.006.

Clements, P., Northrop, L., 2001. Software Product Lines: Practices and Patterns. Addison-Wesley, Boston.

Coello, C.A.C., Lamont, G.B., Veldhuizen, D.A.V., 2006. Evolutionary algorithms for solving multi-objective problems (genetic and evolutionary computation). Springer, Secaucus.

Czarnecki, K., Eisenecker, U.W., 2000. Generative Programming: Methods, Tools, and Applications. Addison-Wesley, New York.

Czarnecki, K., Helsen, S., Eisenecker, U., 2004. Staged configuration using feature models. In: Nord, R.L. (Ed.), Proceedings of the SPLC. Springer, Berlin, Heidelberg, pp. 266–283. https://doi.org/10.1007/978-3-540-28630-1_17.

Czarnecki, K., Kim, C.H.P., 2005. Cardinality-based feature modeling and constraints: a progress report. Proceedings of the International Workshop on Software Factories. ACM, pp. 9.

Deb, K., Pratap, A., Agarwal, S., Meyarivan, T., 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. 6 (2), 182–197. https://doi.org/10.1109/4235.996017.

Dou, R., Zhang, Y., Nan, G., 2016. Customer-oriented product collaborative customization based on design iteration for tablet personal computer configuration. Comput. Ind. Eng. 99 (C), 474–486. https://doi.org/10.1016/j.cie.2015.11.007.

Du, X., Jiao, J., Tseng, M., 2002. Graph grammar based product family modeling. Concurr. Eng. 10 (2), 113–128. https://doi.org/10.1177/1063293X02010002635.

Du, X., Jiao, J., Tseng, M., 2003. Modelling platform-based product configuration using programmed attributed graph grammars. J. Eng. Decis. 14 (2), 145–167. https://doi.org/10.1080/0954482031000091482.

Eiben, A.E., Smith, J.E., 2003. Introduction to Evolutionary Computing. Springer, Berlin, Heidelberg.

El Yamany, A.E., Shaheen, M., Sayyad, A.S., 2014. OPTI-SELECT: an interactive tool for user-in-the-loop feature selection in software product lines. Proceedings of the 18th International Software Product Line Conference. ACM, New York, pp. 126–129. https://doi.org/10.1145/2647908.2655977.

Elo, A.E., 1978. The Rating of Chessplayers, Past and Present. Arco Pub., New York.

Engström, E., Runeson, P., 2011. Software product line testing - A systematic mapping study. Inf. Softw. Technol. 53 (1), 2–13. https://doi.org/10.1016/j.infsof.2010.05.011.

Galster, M., Weyns, D., Tofan, D., Michalik, B., Avgeriou, P., 2014. Variability in software systems - A systematic literature review. IEEE Trans. Softw. Eng. 40 (3), 282–306. https://doi.org/10.1109/TSE.2013.56.

Gamez, N., Haddad, J.E., Fuentes, L., 2015. SPL-TQSSS: a software product line approach for stateful service selection. Proceedings of the IEEE International Conference on Web Services. IEEE, Piscataway, pp. 73–80. https://doi.org/10.1109/ICWS.2015.20.

García-Galán, J., Rana, O.F., Trinidad, P., Cortés, A.R., 2013. Migrating to the cloud - A software product line based analysis. Proceedings of the 3rd International Conference on Cloud Computing and Services Science. SciTePress, pp. 416–426. https://doi.org/10.5220/0004357104160426.

Ghezzi, C., Molzam Sharifloo, A., 2013. Dealing with non-functional requirements for adaptive systems via dynamic software product-line. In: de Lemos, R., Giese, H., Müller, H.A., Shaw, M. (Eds.), Software Engineering for Self-Adaptive Systems II. LNCS 7475. Springer, Berlin, Heidelberg, pp. 191–213. https://doi.org/10.1007/978-3-642-35813-5_8.

Halim, S.A., Jawawi, D.N.A., Ibrahim, N., Zaki, M.Z.M., Deris, S., 2015. Multi attribute architecture design decision for core asset derivation. Jurnal Teknologi 77 (9), 75–87. https://doi.org/10.11113/jt.v77.6187.

Henard, C., Papadakis, M., Harman, M., Le Traon, Y., 2015. Combining multi-objective search and constraint solving for configuring large software product lines. Proceedings of the 37th International Conference on Software Engineering. IEEE, Piscataway, pp. 517–528. https://doi.org/10.1109/ICSE.2015.69.

Hierons, R.M., Li, M., Liu, X., Segura, S., Zheng, W., 2016. SIP: optimal product selection from feature models using many-objective evolutionary optimization. ACM Trans. Softw. Eng. Method. 25 (2), 17:1–17:39. https://doi.org/10.1145/2897760.

Hong, G., Dean, P., Yang, W., Tu, Y., Xue, D., 2008a. Identification of the optimal design and its production process for one-of-a-Kind production. Mater. Product Technol. 44, 607–617. doi:10.4028/www.scientific.net/AMR.44-46.607.

Hong, G., Dean, P., Yang, W., Tu, Y., Xue, D., 2010a. Optimal concurrent product design and process planning based on the requirements of individual customers in one-of-a-kind production. Int. J. Prod. Res. 48 (21), 6341–6366. https://doi.org/10.1080/

00207540903252282.
Hong, G., Hu, L., Xue, D., Tu, Y., Xiong, Y., 2006. Design for customer satisfaction in one-of-a-kind production environment. Proceedings of the 26th Computers and Information in Engineering Conference. ASME, pp. 175–184. https://doi.org/10.1115/DETC2006-99325.
Hong, G., Hu, L., Xue, D., Tu, Y.L., Xiong, Y.L., 2008b. Identification of the optimal product configuration and parameters based on individual customer requirements on performance and costs in one-of-a-kind production. Int. J. Prod. Res. 46 (12), 3297–3326. https://doi.org/10.1080/00207540601099274.
Hong, G., Xue, D., Tu, Y., 2010b. Rapid identification of the optimal product configuration and its parameters based on customer-centric product modeling for one-of-a-kind production. Comput. Ind. 61 (3), 270–279. https://doi.org/10.1016/j.compind.2009.09.006.
Kang, K.C., Cohen, S.G., Hess, J.A., Novak, W.E., Peterson, A.S., 1990. Feature-oriented domain analysis (FODA) feasibility study. Technical Report. Carnegie Mellon University, Pittsburgh.
Karimpour, R., Ruhe, G., 2013. Bi-criteria genetic search for adding new features into an existing product line. Proceedings of the 1st International Workshop on Combining Modelling and Search-Based Software Engineering. IEEE, Piscataway, pp. 34–38. https://doi.org/10.1109/CMSBSE.2013.6604434.
Kitchenham, B., Pearl Brereton, O., Budgen, D., Turner, M., Bailey, J., Linkman, S., 2009. Systematic literature reviews in software engineering - A systematic literature review. Inf. Softw. Technol. 51 (1), 7–15. https://doi.org/10.1016/j.infsof.2008.09.009.
Kitchenham, B.A., Budgen, D., Brereton, P., 2015. Evidence-Based Software Engineering and Systematic Reviews. Chapman & Hall/CRC.
Laguna, M.A., Crespo, Y., 2013. A systematic mapping study on software product line evolution: from legacy system reengineering to product line refactoring. Sci. Comput. Program 78 (8), 1010–1034. https://doi.org/10.1016/j.scico.2012.05.003.
Lee, J., Kang, K.C., Sawyer, P., Lee, H., 2014. A holistic approach to feature modeling for product line requirements engineering. Requir. Eng. 19 (4), 377–395. https://doi.org/10.1007/s00766-013-0183-6.
Leite, A.F., Alves, V., Rodrigues, G.N., Tadonki, C., Eisenbeis, C., Melo, A.C.M.A.d., 2015. Automating resource selection and configuration in inter-clouds through a software product line method. Proceedings of the 8th International Conference on Cloud Computing. IEEE, Washington, pp. 726–733. https://doi.org/10.1109/CLOUD.2015.101.
Li, J., Liu, X., Wang, Y., Guo, J., 2012. Formalizing feature selection problem in software product lines using 0–1 programming. In: Wang, Y., Li, T. (Eds.), Practical Applications of Intelligent Systems. Advances in Intelligent and Soft Computing 124. Springer, Berlin, Heidelberg, pp. 459–465. https://doi.org/10.1007/978-3-642-25658-5_55.
Lian, X., Zhang, L., 2014. An evolutionary methodology for optimized feature selection in software product lines. Proceedings of the 26th International Conference on Software Engineering and Knowledge Engineering. Knowledge Systems Institute Graduate School, pp. 63–66.
Lian, X., Zhang, L., 2015. Optimized feature selection towards functional and non-functional requirements in software product lines. Proceedings of the 22nd International Conference on Software Analysis, Evolution, and Reengineering. IEEE, Piscataway, pp. 191–200. https://doi.org/10.1109/SANER.2015.7081829.
Liu, Y., Su, Y., Yin, X., Mussbacher, G., 2014. Combined Propagation-based reasoning with goal and feature models. Proceedings of the 4th International Model-Driven Requirements Engineering Workshop. IEEE, Piscataway, pp. 27–36. https://doi.org/10.1109/MoDRE.2014.6890823.
Lopez-Herrejon, R.E., Batory, D.S., 2001. A standard problem for evaluating product-line methodologies. Proceedings of the 3rd International Conference on Generative and Component-based Software Engineering. Springer, London, pp. 10–24. https://doi.org/10.1007/3-540-44800-4_2.
Mahdavi-Hezavehi, S., Galster, M., Avgeriou, P., 2013. Variability in quality attributes of service-based software systems: a Systematic literature review. Inf. Softw. Technol. 55 (2), 320–343. https://doi.org/10.1016/j.infsof.2012.08.010.
Mannion, M., 2002. Using first-order logic for product line model validation. Proceedings of the 2nd International Conference on Software Product Lines. Springer, London, pp. 176–187. https://doi.org/10.1007/3-540-45652-X_11.
Martinez, J., Ziadi, T., Mazo, R., Bissyandé, T.F., Klein, J., Traon, Y.L., 2014. Feature relations graphs: a visualisation paradigm for feature constraints in software product lines. Proceedings of the 2nd IEEE Working Conference on Software Visualization. IEEE, Piscataway, pp. 50–59. https://doi.org/10.1109/VISSOFT.2014.18.
Mazo, R., Dumitrescu, C., Salinesi, C., Diaz, D., 2014. Recommendation heuristics for improving product line configuration processes. In: Robillard, M.P., Maalej, W., Walker, R.J., Zimmermann, T. (Eds.), Recommendation Systems in Software Engineering. Springer, Berlin, Heidelberg, pp. 511–537. https://doi.org/10.1007/978-3-642-45135-5_19.
Mazo, R., Salinesi, C., Diaz, D., Djebbi, O., Lora-Michiels, A., 2012. Constraints: the heart of domain and application engineering in the product lines engineering strategy. Int. J. Inf. Syst. Model. Design 3 (2), 33–68. https://doi.org/10.4018/jismd.2012040102.
Mendonça, M., Bartolomei, T.T., Cowan, D., 2008. Decision-making coordination in collaborative product configuration. Proceedings of the ACM Symposium on Applied Computing. ACM, New York, pp. 108–113. https://doi.org/10.1145/1363686.1363715.
Metzger, A., Pohl, K., Heymans, P., Schobbens, P.Y., Saval, G., 2007. Disambiguating the documentation of variability in software product lines: a separation of concerns, formalization and automated analysis. Proceedings of the 15th IEEE International Requirements Engineering Conference. IEEE, Piscataway, pp. 243–253. https://doi.org/10.1109/RE.2007.61.
da Mota Silveira Neto, P.A., Carmo Machado, I.d., McGregor, J.D., de Almeida, E.S., de Lemos Meira, S.R., 2011. A systematic mapping study of software product lines testing. Inf. Softw. Technol. 53 (5), 407–423. https://doi.org/10.1016/j.infsof.2010.12.003.
Murashkin, A., Antkiewicz, M., Rayside, D., Czarnecki, K., 2013. Visualization and exploration of optimal variants in product line engineering. Proceedings of the 17th International Software Product Line Conference. ACM, New York, pp. 111–115. https://doi.org/10.1145/2491627.2491647.
Mussbacher, G., Araújo, J.a., Moreira, A., Amyot, D., 2012. AoURN-based modeling and analysis of software product lines. Softw. Qual. J. 20 (3–4), 645–687. https://doi.org/10.1007/s11219-011-9153-8.
Myllärniemi, V., Raatikainen, M., Männistö, T., 2015. Representing and configuring security variability in software product lines. Proceedings of the 11th International Conference on Quality of Software Architectures. ACM, New York, pp. 1–10. https://doi.org/10.1145/2737182.2737183.
Noorian, M., Bagheri, E., Du, W., 2014. From intentions to decisions: understanding stakeholders' objectives in software product line configuration. Proceedings of the 26th International Conference on Software Engineering and Knowledge Engineering. Knowledge Systems Institute Graduate School, Skokie, pp. 671–677.
Noorian, M., Bagheri, E., Du, W., 2016. Quality-centric feature model configuration using goal models. Proceedings of the 31st Symposium on Applied Computing. ACM, New York, pp. 1296–1299. https://doi.org/10.1145/2851613.2851959.
Ochoa, L., González-Rojas, O., 2017. Program synthesis for configuring collaborative solutions in feature models. In: Ciuciu, I., Debruyne, C., Panetto, H., Weichhart, G., Bollen, P., Fensel, A., Vidal, M.-E. (Eds.), Proceedings of the OTM 2016 Workshops. LNCS 10034. Springer, Cham, pp. 98–108. https://doi.org/10.1007/978-3-319-55961-2_10.
Ochoa, L., González-Rojas, O., Thüm, T., 2015. Using decision rules for solving conflicts in extended feature models. Proceedings of the International Conference on Software Language Engineering. ACM, New York, pp. 149–160. https://doi.org/10.1145/2814251.2814263.
Ochoa, L., Pereira, J.A., González-Rojas, O., Castro, H., Saake, G., 2017. A survey on scalability and performance concerns in extended product lines configuration. Proceedings of the 11th International Workshop on Variability Modelling of Software-intensive Systems. ACM, New York, pp. 5–12. https://doi.org/10.1145/3023956.3023959.
Olaechea, R., Rayside, D., Guo, J., Czarnecki, K., 2014. Comparison of exact and approximate multi-objective optimization for software product lines. 18th Int. Software Product Line Conference. ACM, New York, pp. 92–101. https://doi.org/10.1145/2648511.2648521.
Olaechea, R., Stewart, S., Czarnecki, K., Rayside, D., 2012. Modelling and multi-objective optimization of quality attributes in variability-rich software. Proceedings of the 4th International Workshop on Nonfunctional System Properties in Domain Specific Modeling Languages. ACM, New York, pp. 2:1–2:6. https://doi.org/10.1145/2420942.2420944.
Ong, S.K., Lin, Q., Nee, A.Y.C., 2006. Web-based configuration design system for product customization. Int. J. Prod. Res. 44 (2), 351–382. https://doi.org/10.1080/00207540500244153.
Ostrosi, E., Fougères, A.J., Ferney, M., Klein, D., 2012. A fuzzy configuration multi-agent approach for product family modelling in conceptual design. J. Intell. Manuf. 23 (6), 2565–2586. https://doi.org/10.1007/s10845-011-0541-5.
Pascual, G.G., Lopez-Herrejon, R.E., Pinto, M., Fuentes, L., Egyed, A., 2015. Applying multiobjective evolutionary algorithms to dynamic software product lines for reconfiguring mobile applications. J. Syst. Softw. 103 (C), 392–411. https://doi.org/10.1016/j.jss.2014.12.041.
Pereira, J.A., Constantino, K., Figueiredo, E., 2014. A systematic literature review of software product line management tools. Software Reuse for Dynamic Systems in the Cloud and Beyond. Springer, pp. 73–89. https://doi.org/10.1007/978-3-319-14130-5_6.
Pereira, J.A., Matuszyk, P., Krieter, S., Spiliopoulou, M., Saake, G., 2016. A feature-based personalized recommender system for product-line configuration. Proceedings of the International Conference on Generative Programming: Concepts and Experiences. ACM, New York, pp. 120–131. https://doi.org/10.1145/2993236.2993249.
Pohl, K., Böckle, G., Linden, F.J.v.d., 2005. Software Product Line Engineering: Foundations, Principles and Techniques. Springer, Secaucus.
Quinton, C., Duchien, L., Heymans, P., Mouton, S., Charlier, E., 2012. Using feature modelling and automations to select among cloud solutions. Proceedings of the 3rd International Workshop on Product LinE Approaches in Software Engineering. IEEE, Piscataway, pp. 17–20. https://doi.org/10.1109/PLEASE.2012.6229762.
Quinton, C., Haderer, N., Rouvoy, R., Duchien, L., 2013. Towards multi-cloud configurations using feature models and ontologies. Proceedings of the International Workshop on Multi-cloud Applications and Federated Clouds. ACM, New York, pp. 21–26. https://doi.org/10.1145/2462326.2462332.
Rabiser, R., Grünbacher, P., Dhungana, D., 2010. Requirements for product derivation support: results from a systematic literature review and an expert survey. Inf. Softw. Technol. 52 (3), 324–346. https://doi.org/10.1016/j.infsof.2009.11.001.
Rossi, F., Beek, P.v., Walsh, T., 2006. Handbook of Constraint Programming (Foundations of Artificial Intelligence). Elsevier, New York.
Saaty, R., 1987. The analytic hierarchy process - What it is and how it is used. Math. Model. 9 (3), 161–176. https://doi.org/10.1016/0270-0255(87)90473-8.
Saaty, T.L., 2008. Decision making with the analytic hierarchy process. Int. J. Serv. Sci. 1 (1), 83–98. https://doi.org/10.1504/IJSSci.2008.01759.
Salinesi, C., Mazo, R., Diaz, D., Djebbi, O., 2010. Using integer constraint solving in reuse based requirements engineering. Proceedings of the 18th IEEE International Requirements Engineering Conference. IEEE, Piscataway, pp. 243–251. https://doi.org/10.1109/RE.2010.36.
Sánchez, D.M., Cavero, J.M., Martínez, E.M., 2007. The road toward ontologies. In: Sharman, R., Kishore, R., Ramesh, R. (Eds.), Ontologies: A Handbook of Principles,

Concepts and Applications in Information Systems. Integrated Series in Information Systems 14. Springer, Boston, pp. 3–20. https://doi.org/10.1007/978-0-387-37022-4_1.
Sánchez, L.E., Diaz-Pace, J.A., Zunino, A., Moisan, S., Rigault, J.-P., 2014. An approach for managing quality attributes at runtime using feature models. Proceedings of the 8th Brazilian Symposium on Software Components, Architectures and Reuse. IEEE, Piscataway, pp. 11–20. https://doi.org/10.1109/SBCARS.2014.13.
Sayyad, A.S., Ingram, J., Menzies, T., Ammar, H., 2013a. Optimum feature selection in software product Lines: let your model and values guide your search. Proceedings of the 1st International Workshop on Combining Modelling and Search-Based Software Engineering. IEEE, Piscataway, pp. 22–27. https://doi.org/10.1109/CMSBSE.2013.6604432.
Sayyad, A.S., Ingram, J., Menzies, T., Ammar, H., 2013b. Scalable product line configuration: a straw to break the camel's back. Proceedings of the 28th International Conference on Automated Software Engineering. IEEE, Piscataway, pp. 465–474. https://doi.org/10.1109/ASE.2013.6693104.
Sayyad, A.S., Menzies, T., Ammar, H., 2013c. On the value of user preferences in search-based software engineering: a case study in software product lines. Proceedings of the International Conference on Software Engineering. IEEE, Piscataway, pp. 492–501. https://doi.org/10.1109/ICSE.2013.6606595.
Schroeter, J., Mucha, P., Muth, M., Jugel, K., Lochau, M., 2012. Dynamic configuration management of cloud-based applications. Proceedings of the 16th International Software Product Line Conference. ACM, New York, pp. 171–178. https://doi.org/10.1145/2364412.2364441.
Seaman, C.B., 1999. Qualitative methods in empirical studies of software engineering. IEEE Trans. Softw. Eng. 25 (4), 557–572. https://doi.org/10.1109/32.799955.
Seel, N.M., 2012. Grammar induction. Encyclopedia of the Sciences of Learning. Springer, Boston. https://doi.org/10.1007/978-1-4419-1428-6_4188. 1383-1383.
Shekhar, S., Xiong, H., 2008. Ontology. Encyclopedia of GIS. Springer, Boston. https://doi.org/10.1007/978-0-387-35973-1_915. 812-812.
Shi, R., Guo, J., Wang, Y., 2010. A preliminary experimental study on optimal feature selection for product derivation using knapsack approximation. Proceedings of the International Conference on Progress in Informatics and Computing. IEEE, pp. 665–669. https://doi.org/10.1109/PIC.2010.5687874.
Siegmund, N., Mory, M., Feigenspan, J., Saake, G., Nykolaychuk, M., Schumann, M., 2012a. Interoperability of non-functional requirements in complex systems. Proceedings of the 2nd International Workshop on Software Engineering for Embedded Systems. IEEE, Piscataway, pp. 2–8. https://doi.org/10.1109/SEES.2012.6225487.
Siegmund, N., Rosenmüller, M., Kuhlemann, M., Kästner, C., Apel, S., Saake, G., 2012b. SPL conqueror: toward optimization of non-functional properties in software product lines. Softw. Qual. J. 20 (3–4), 487–517. https://doi.org/10.1007/s11219-011-9152-9.
Siegmund, N., Rosenmõller, M., Kuhlemann, M., Kästner, C., Saake, G., 2008. Measuring non-functional properties in software product line for product derivation. Proceedings of the 15th Asia-Pacific Software Engineering Conference. IEEE, Piscataway, pp. 187–194. https://doi.org/10.1109/APSEC.2008.45.
da Silva, J.R.F., da Silva, F.A.P., do Nascimento, L.M., Martins, D.A.O., Garcia, V.C., 2013. The dynamic aspects of product derivation in DSPL: a systematic literature review. Proceedings of the 14th International Conference on Information Reuse Integration. IEEE, Piscataway, pp. 466–473. https://doi.org/10.1109/IRI.2013.6642507.
Soares, L.R., Potena, P., d. C. Machado, I., Crnkovic, I., d. Almeida, E.S., 2014. Analysis of non-functional properties in software product lines: a systematic review. Proceedings of the 40th EUROMICRO Conference on Software Engineering and Advanced Applications. IEEE, Piscataway, pp. 328–335. https://doi.org/10.1109/SEAA.2014.48.
Soltani, S., Asadi, M., Gašević, D., Hatala, M., Bagheri, E., 2012. Automated planning for feature model configuration based on functional and non-functional requirements. Proceedings of the 16th International Software Product Line Conference. ACM, New York, pp. 56–65. https://doi.org/10.1145/2362536.2362548.
Song, H., Xue, D., Tu, Y., 2007. A customer-driven approach to one-of-a-kind product design. Proceedings of the International Design Engineering Technical Conferences and Computers and Information in Engineering Conference. ASME, pp. 959–969. https://doi.org/10.1115/DETC2007-34878.
Tan, L., Lin, Y., Liu, L., 2014. Quality ranking of features in software product line engineering. Proceedings of the 21st Asia-Pacific Software Engineering Conference. IEEE, Piscataway, pp. 57–62. https://doi.org/10.1109/APSEC.2014.94.
Tan, T.H., Xue, Y., Chen, M., Sun, J., Liu, Y., Dong, J.S., 2015. Optimizing selection of competing features via feedback-directed evolutionary algorithms. Proceedings of the International Symposium on Software Testing and Analysis. ACM, New York, pp. 246–256. https://doi.org/10.1145/2771783.2771808.
Vanderbei, R.J., 2014. Linear Programming Foundations and Extensions, 4 edition. Springer, New York.
Villela, K., Arif, T., Zanardini, D., 2012. Towards product configuration taking into account quality concerns. Proceedings of the 16th International Software Product Line Conference. ACM, New York, pp. 82–90. https://doi.org/10.1145/2364412.2364426.
White, J., Benavides, D., Schmidt, D., Trinidad, P., Dougherty, B., Ruiz-Cortes, A., 2010. Automated diagnosis of feature model configurations. J. Syst. Softw. 83 (7), 1094–1107. https://doi.org/10.1016/j.jss.2010.02.017.
White, J., Dougherty, B., Schmidt, D.C., 2009. Selecting highly optimal architectural feature sets with filtered cartesian flattening. J. Syst. Softw. 82 (8), 1268–1284. https://doi.org/10.1016/j.jss.2009.02.011.
White, J., Galindo, J.A., Saxena, T., Dougherty, B., Benavides, D., Schmidt, D.C., 2014. Evolving feature model configurations in software product lines. J. Syst. Softw. 87, 119–136. https://doi.org/10.1016/j.jss.2013.10.010.
White, J., Schmidt, D.C., Wuchner, E., Nechypurenko, A., 2007. Automating product-line variant selection for mobile devices. Proceedings of the 11th International Software Product Line Conference. IEEE, Washington, pp. 129–140. https://doi.org/10.1109/SPLC.2007.12.
White, J., Schmidt, D.C., Wuchner, E., Nechypurenko, A., 2008. Automatically composing reusable software components for mobile devices. J. Braz. Comput. Soc. 14 (1), 25–44. https://doi.org/10.1007/BF03192550.
Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A., 2000. Experimentation in Software Engineering: An Introduction. Kluwer Academic Publishers, Norwell.
Yeh, J.-Y., Wu, T.-H., 2005. Solutions for product configuration management: an empirical study. Artif. Intell. Eng. Design Anal. Manuf. 19 (1), 39–47. https://doi.org/10.1017/S0890060405050043.
Yu, X., Gen, M., 2012. Introduction to Evolutionary Algorithms. Springer, London.
Yu, Y., do Prado Leite, J.C.S., Lapouchnian, A., Mylopoulos, J., 2008. Configuring features with stakeholder goals. Proceedings of the Symposium on Applied Computing. ACM, New York, pp. 645–649. https://doi.org/10.1145/1363686.1363840.
Zanardini, D., Albert, E., Villela, K., 2016. Resource-usage-aware configuration in software product lines. J. Log. Algebraic Methods Progr. 85 (1), 173–199. https://doi.org/10.1016/j.jlamp.2015.08.003.
Zhang, G., Ye, H., Lin, Y., 2011. Using knowledge-based systems to manage quality attributes in software product lines. Proceedings of the 15th International Software Product Line Conference. ACM, New York, pp. 32:1–32:7. https://doi.org/10.1145/2019136.2019172.
Zhang, G., Ye, H., Lin, Y., 2014. Quality attribute modeling and quality aware product configuration in software product lines. Softw. Qual. J. 22 (3), 365–401. https://doi.org/10.1007/s11219-013-9197-z.

Lina Ochoa is a Ph.D. candidate in Software Engineering at Centrum Wiskunde & Informatica. Currently, she is a member of the Software and Analysis Transformation (SWAT) research group. She did her Bachelor in Systems and Computing Engineering, and her Master in Software Engineering at Universidad de los Andes. During this period her research was focused on product lines configuration and modeling. This paper was a result of her thesis research.

Oscar González-Rojas is Associate Professor at the Systems and Computing Engineering Department at the Universidad de los Andes in Colombia. He received a Ph.D. degree in Computer Science at the Vrije Universiteit Brussel (VUB), Belgium and at the Universidad de los Andes (Uniandes), Colombia in 2010. He is lecturer of Business Process Management (BPM) Governance and IT Governance courses in the Master in Business and Information Technology at Uniandes. His main actuation areas in research and consulting are BPM, IT Governance, and Enterprise Architecture.

Juliana is a Ph.D. student at the University of Magdeburg (Germany). She has a master's degree in Computer Science at the Federal University of Minas Gerais, Brazil (2014) and a bachelor at the Federal University of Lavras, Brazil (2012). She is tutor in the Distributed Data Management lecture at the University of Magdeburg (Germany). She has experience in the field of Computer Science, with emphasis on Software Engineering, working on the following topics: software reuse, software product lines, search-based software engineering, empirical software engineering, and recommender systems. In recent years, she has published and revised research papers in premier software engineering conferences, symposium, and journals. She is currently a member of the Association for Computing Machinery (ACM) and of the workgroup DBSE (Databases & Software Engineering) at the Faculty of Computer Science at the University of Magdeburg (Germany).

Harold Castro graduated as Computing and System Engineer at Universidad de los Andes in Bogota, Colombia. He got a D.E.A (MSc) from the Institut National Polytechnique de Grenoble (INPG), in Grenoble, France and since 1995 he holds a Ph.D. in computer science from INPG also. Associate professor at the Computing and Systems Department at Universidad de los Andes. He is the director of the COMIT (Communications and Information Technology) research group which main research focus are distributed systems and High Performance Computing (HPC) Systems. Dr. Castro personally leads institutional and national cloud and HPC initiatives, and he was national coordinator for the establishment of a grid HPC platform between Europe and Latina America. His interest areas are: HPC, distributed systems and cloud computing.

Gunter Saake received the diploma and the Ph. D. degree in computer science from the Technical University of Braunschweig, F. R. G., in 1985 and 1988, respectively. From 1988 to 1989 he was a visiting scientist at the IBM Heidelberg Scientific Center where he joined the Advanced Information Management project and worked on language features and algorithms for sorting and duplicate elimination in nested relational database structures. In January 1993 he received the Habilitation degree (venia legendi) for computer science from the Technical University of Braunschweig. Since May 1994, Gunter Saake is full professor for the area Databases and Information Systems at the Otto-von-Guericke-University Magdeburg. From April 1996 to March 1998, he was dean of the faculty for computer science at the Otto-von-Guericke-University Magdeburg and was again elected as Dean in 2012.