Automated Analysis of Feature Models: A Detailed Literature Review

Version 1.0

David Benavides, Sergio Segura and Antonio Ruiz-Cortés
{benavides,sergiosegura,aruiz}@us.es

Applied Software Engineering Research Group
University of Seville, Spain
December 2009

Technical Report ISA-09-TR-04

This report was prepared by the

Applied Software Engineering Research Group (ISA)
Department of computer languages and systems
Av/ Reina Mercedes S/N, 41012 Seville, Spain
http://www.isa.us.es/

Support: This work has been partially supported by the European Commission (FEDER) and Spanish Government under CICYT projects SETI (TIN2009-07366) and WebFactories(TIN2006-00472) and by the Andalusian Government under ISABEL project (TIC-2533).

Contents

List of Tables

Chapter 1

20 years of analysis of feature models

Automated Analysis of Feature Models after 20 years: A Literature Review☆,☆☆

David Benavides, Sergio Segura and Antonio Ruiz-Cortés

Dpto. de Lenguajes y Sistemas Informáticos, University of Seville
Av. Reina Mercedes s/n, 41012, Seville - Spain

Abstract

Software product line engineering is about producing a set of related products that share more commonalities than variabilities. Feature models are widely used for variability and commonality management in software product lines. Feature models are information models where a set of products are represented as a set of features in a single model. The automated analysis of feature models cope with the computer–aided extraction of information from feature models. The literature on this topic has contributed with a set of operations, techniques, tools and empirical results which have not been surveyed until now. This paper provides a compehensive literature review on the automated analysis of feature models after 20 years of their invention. We contribute in this paper by bringing together previously-disparate streams of work to help shed light on this thriving area. We also present a conceptual framework to understand the different proposals as well as categorise future contributions. We finally discuss the different studies and propose some challenges to be faced in the future.

Key words: Feature models, automated analyses, software product lines, literature review

1. Introduction

Mass production is defined as the production of a large amount of standardized products using standardized processes that produce a large volume of the same product in a reduced time to market. Generally, the customers' requirements are the same and no customization is performed (imagine Japanese watches of the nineteens). After the industrial revolution, large companies started to organise –and are still organising– their production in a mass production environment.

However, mass production is already not enough in a highly competitive and segmented market and *mass customization* is devised to be a must for market succeed. According

☆This work has been partially supported by the European Commission (FEDER) and Spanish Government under CICYT projects SETI (TIN2009-07366) and WebFactories(TIN2006-00472) and by the Andalusian Government under ISABEL project (TIC-2533)

☆☆A very preliminay version of this paper was published in Jornadas de Ingeniería del Software y Bases de Datos (JISBD'06)

Email address: {benavides,sergiosegura,aruiz}@us.es (David Benavides, Sergio Segura and Antonio Ruiz-Cortés)

to Tseng and Jiao [87], mass customization is about "*producing goods and services to meet individual customer's needs with near mass production efficiency*". There are two key parts in this definition. Firstly, mass customization tries to meet as much individual customer's needs as possible (imagine current mobile phones). Secondly, this has to be done trying to meet as much as possible the mass production efficiency. To achieve this efficiency, practitioners propose to build products from existing assets sharing more commonalities than singularities.

Information systems market is a peculiar branch of industry in relation to more traditional branches. Making the parallelism with the history of traditional industries, the industrialization of information systems started with artisanal methods, evolved to mass production and is now pursuing mass customization to succeed in the market. In the software engineering literature, the mass customization of software products is known as *software product lines* [24] or *software product families* [66]. In order to achieve customer's personalization, *software product line engineering* promote the production of a family of software products from common features instead of producing them one by one from scratch. This is the key change: software product line engineering is about producing families of similar systems rather than the production of individual systems.

Software product lines have found a broad adoption in several branches of software production such as embedded systems for mobile devices, car embedded software and avionics [89]. However, other types of software and systems applications such as desktop or web applications are also pursuing the adoption.

An organisation decides to set up a software product line and faces the following questions, How is a particular product specified?, and How is the software product line itself specified? When this question was first posed, there was an ample evidence for a solution: in other industries product lines are specified in terms of *features*. Products in a software product line are differentiated by their features, where a feature is an increment in program functionality [6]. Individual products are specified using features, software product lines are specified using *feature models*.

Feature model languages are a common family of visual languages to represent software product lines [73]. The first formulation of a feature model language is due by Kang *et al.* in 1990 [48]. A feature model captures software product line information about common and variant features of the software product line at different levels of abstraction. A feature model is represented as a hierarchically arranged set of features with different relationships among those features. It models all possible products of a software product line in a given context. In contrast to classical information models, feature models not only represent a single product but a family of them in the same model.

The automated analysis of feature models is about extracting information from feature models using automated mechanisms [6]. Analysing feature models is an error–prone tedious task, manually infeasible with large–scale feature models. It is an active area of research and is gaining importance in both practitioners and researchers in the software product line community [6, 8]. Since the introduction of feature models, the literature has contributed with a number of operations of analysis, tools, paradigms and algorithms to support the analysis process.

In this article, we present a structured literature review on existing proposals for the automated analysis of feature models. We used a structured and systematic method to perform the literature review inspired by the guidelines proposed by Kitchenham [51] and Webster *et al.* [98]. The main contribution in this article is to bring together previously–scattered studies to put the basis for future research as well as introduce new researchers and practitioners in this thriving area. We present a conceptual framework to understand the different proposals and classify new contributions in the future. 53 primary studies were analysed from where we report 30 operations of analysis and 4 different groups of proposals to automate those operations. As a result of our literature review, we also report some challenges that remain open for research.

The main target audience of this literature review are researchers in the field of automated analysis, tool developers or practitioners that are interested in analysis of feature models as well as researchers and professionals on information systems interested in software product lines, their models and analyses.

The remaining of the paper is structured as follows: Section 2 presents feature models in a nutshell. Section 3 presents the method used in the literature review. Section 4 describes the conceptual framework that we use to clasify primary studies and define some cocepts used along the paper. Section 5 presents the main results of this review where different analysis operations are presented and explained and primary studies are classified according to the automated method used for analysis. Section 6 discusses the results about performance analysis of feature models. Section 7 discusses the results obtained and describe some challenges to be faced in the future. Finally, Section 8 presents some conclusions.

2. Feature Models

A feature model represents the information of all possible products of a software product line in terms of features and relationships among them. Feature models are a special type of information models widely used in software product line engineering. A feature model is represented as a hierarchically arranged set of features composed by:

1. relationships between a parent (or compound) feature and its child features (or subfeatures).
2. cross–tree (or cross–hierarchy) constraints that are typically inclusion or exclusion statements of the form: *if feature F is included, then features A and B must also be included (or excluded).*

Figure 1 depicts a simplified feature model inspired by the mobile phone industry. The model illustrates how features are used to specify and build software for mobile phones. The software loaded in the phone is determined by the features that it supports. According to the model, all phones must include support for *calls*, and displaying information in either a *basic*, *colour* or *high resolution* screen. Furthermore, the software for mobile phones may optionally include support for *GPS* and multimedia devices such as *camera*, *MP3* player or both of them.

Feature models are used in different scenarios of software production ranging from model driven development [85], feature oriented programming [5], software factories [44] or generative programming [27], all of them around software product line development. Although feature models are studied in software product line engineering, these information models can be used in different contexts ranging from requirements gathering

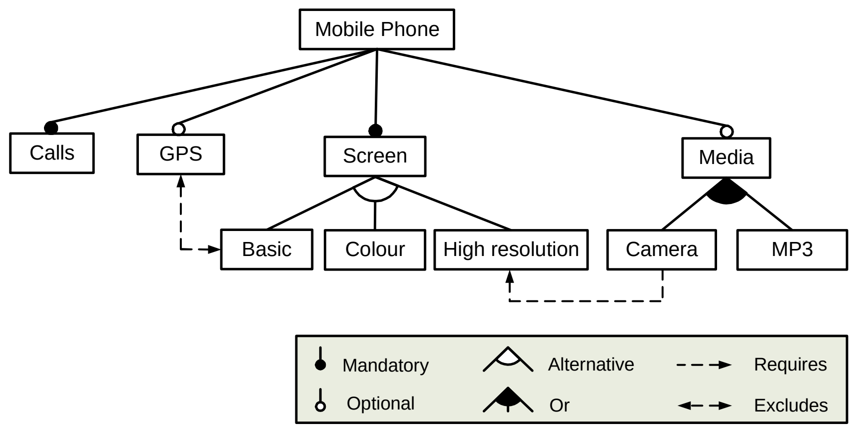


Figure 1: A sample feature model

[23] to data model structures, hence the potential importance of feature models in the information systems domain.

The term *feature model* was coined by Kang et *al.* in the FODA report back in 1990 [48] and has been one of the main topics of research in software product lines since then. There are different feature model languages. We refer the reader to [73] for a detailed survey on the different feature model languages. Following, we review the most well known notations for those languages.

2.1. Basic feature models

We group as basic feature models those allowing the following relationships among features:

- **Mandatory.** A child feature has a mandatory relationships with its parent when the child is included in all products in which its parent feature appears. For instance, every mobile phone system in our example must provide support for *calls*.
- **Optional.** A child feature has an optional relationship with its parent when the child can be optionally included in all products in which its parent feature appears. In the example, software for mobile phones may optionally include support for *GPS*.
- **Alternative.** A set of child features have an alternative relationship with their parent when only one feature of the children can be selected when its parent feature is part of the product. In the example, mobile phones may include support for a *basic*, *colour* or *high resolution* screen but only one of them.
- **Or.** A set of child features have an or-relationship with their parent when one or more of them can be included in the products in which its parent feature appears. In Figure 1, whenever *Media* is selected, *Camera*, *MP3* or both can be selected.

Notice that a child feature can only appear in a product if its parent feature does. The root feature is a part of all the products within the software product line. In addition to the parental relationships between features, a feature model can also contain cross-tree constraints between features. These are typically of the form:

- **Requires.** If a feature A requires a feature B, the inclusion of A in a product implies the inclusion of B in such product. Mobile phones including a *camera* must include support for a *high resolution* screen.
- **Excludes.** If a feature A excludes a feature B, both features cannot be part of the same product. *GPS* and *basic* screen are incompatible features.

More complex cross-tree relationships have been proposed later in the literature [4] allowing constraints in the form of generic propositional formulas, e.g. "A and B implies not C".

2.2. Cardinality–based feature models

Some authors propose extending FODA feature models with UML-like multiplicities (so-called *cardinalities*) [29, 69]. Their main motivation was driven by practical applications [26] and "conceptual completeness". The new relationships introduced in this notation are defined as follows:

- **Feature cardinality.** A feature cardinality is a sequence of intervals denoted $[n..m]$ with n as lower bound and m as upper bound. These intervals determines the number of instances of the feature that can be part of a product. This relationship may be used as a generalization of the original mandatory ($[1,1]$) and optional ($[0,1]$) relationships defined in FODA.
- **Group cardinality.** A group cardinality is an interval denoted $\langle n..m \rangle$, with n as lower bound and m as upper bound limiting the number of child features that can be part of a product when its parent feature is selected. Thus, an alternative relationship is equivalent to a $\langle 1..1 \rangle$ group cardinality and an or–relationship is equivalent to $\langle 1..N \rangle$, being N the number of features in the relationship.

2.3. Extended feature models

Sometimes it is necessary to extend feature models to include more information about features. This information is added in terms of so–called *feature attributes*. These type of models where additional information is included are called *extended, advanced or attributed feature models*

FODA [48], the seminal report on feature models, already contemplated the inclusion of some additional information in feature models. For instance, relationships between features and feature attributes were introduced. Later, Kang *et al.* [49] make an explicit reference to what they called "non–functional" features related to feature attributes. In addition, other group of authors have also proposed the inclusion of attributes in feature models [4, 6, 10, 11, 28, 30, 77, 100]. There is no consensus on a notation to define attributes. However, most proposals agree that an attribute should consist at least of a *name*, a *domain* and a *value*. Figure 2 depicts a sample feature model including attributes using the notation proposed by Benavides *et al.* in [10]. As illustrated, attributes can be used to specify extra-functional information such as cost, speed or RAM memory required to support the feature.

Extended feature models can also include complex constraints among attributes and features like: "*If attribute A of feature F is lower than a value X, then feature T can not be part of the product*".

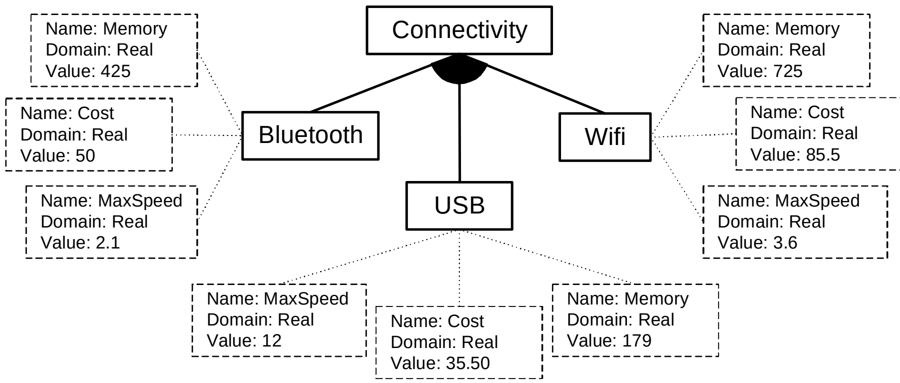


Figure 2: A sample extended feature model

3. Review method

Within the context of this paper we have carried out a literature review in order to examine studies proposing automated analysis of feature models. To perform this review we followed a systematic and structured method inspired by the guidelines of Kitchenham [51] and Webster *et al.* [98]. Following, we detail the main data regarding the review process and its structure.

3.1. Research questions

Performing this review we intend to answer the following research questions:

- *RQ1: What operations of analysis on feature models have been proposed?* This question motivates the following sub-questions:
  - What operations have been formally described?
- *RQ2: What kind of automated support has been proposed and how it is performed?* This question motivates the following sub-questions:
  - Which analysis operations have been automated?
  - Which tools have been proposed to automate the analysis?
  - What is the feature modelling notation supported by each approach?
  - What proposals present any performance evaluation of their results?

After reviewing all this information we also want to answer a more general question:

- *RQ3: What are the challenges to be faced in the future?*

3.2. Source material

As recommended by Webster *et al.* [98], we used the following sources of information:

- Leading journals and conferences related to the software product line community, namely: Software Product Line Conference (SPLC), International Conference on Software Engineering (ICSE), International Conference on Software Reuse (ICSR), Generative Programming and Component Engineering (GPCE), Conference on Advances Information Systems Engineering (CAiSE) and Requirements Engineering Conference (RE). We also reviewed special issues of different journals related to the former publications as well as general top journals in software engineering publishing results on software product lines such as IEEE Transactions on Software Engineering (TSE) and ACM Transactions on Software Engineering and Methodology (TOSEM). We may mention that no studies were found in the latter two journals. Finally, we also looked at specific workshops and events where one of the main topics are automated analysis on feature models, namely, Workshop on Automated Analyses of Software Product Lines (ASPL) and Variability Management for Software–intensive Systems (VaMoS).

- References in papers identified in the previous step. We analysed the references of the papers identified in our initial search to determine prior articles to be considered.

- As a last step, we searched, using web–based tools like google scholar [43], scopus [74] or citeseer [21], the citation of other papers to the key papers identified in previous steps.

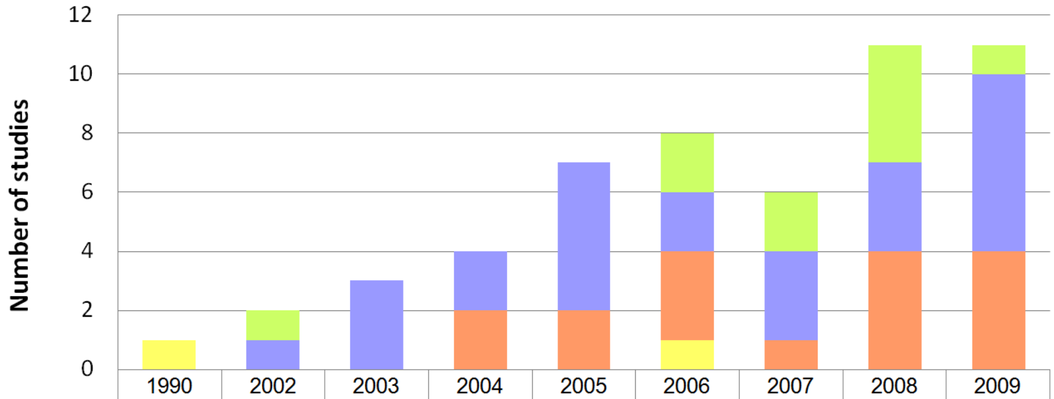
All papers included in our work are peer-reviewed and were presented in international events or journals with the exception of the work of Kang *et al.* [48], seminal report in the field of feature modelling.

3.3. Search process

For the selection of candidate papers from the literature we carried out a three–steps search process:

1. We examined the selected proceedings and journals using a manual scan of titles. We selected those papers whose title suggests any kind of analysis or automation in software product lines, e.g. "Product line model validation".
2. We examined the abstracts of the papers identified in the previous step. Excepcionally, we overview the context of certain papers when unsure. As a result, we narrowed the search to 72 candidate papers.
3. Finally, we read the papers carefully to determine whether they propose some kind of analysis of feature models. From the original 72 papers, 19 were discarded resulting in a total of **53 papers** that were in the scope of this review (see Figure 3). We refer the reader to [12] for details of matrix and data extracted. These 53 papers are referred as *primary studies* [18].

The former method was incremental and some papers were included after analysing some others. Figure 3 classifies primary studies according to the year and type of publication. From the 53 papers included in the review, 10 were published in journals, 25 in conferences, 16 in workshops, 1 in the formal post proceeding of a summer school and 1 in a technical report. The graph indicates that there was an important gap between 1990 and 2002 and since then the tendency seems to be ascendant.



	1990	2002	2003	2004	2005	2006	2007	2008	2009
Journals		[92]				[6,105]	[73,97]	[42,46,60,80]	[100]
Conferences		[56]	[20,57,93]	[90,106]	[4,10,11,78,95]	[39,72]	[3,38,91]	[62,103,107]	[1,40,61,79,101,104]
Workshops				[9,94]	[30,96]	[13,41,82]	[15]	[45,63,75,102]	[64,70,84,88]
Others	[48]					[14]			

Figure 3: Classification of papers per year and type of publication

3.4. Inclusion and exclusion criteria

Articles on the following topics, published between January 1st 1990 and December 31st 2009, were included: *i*) papers proposing any analysis operation on feature models in which the original model is not modified, *ii*) papers proposing the automation of any analysis on feature models, and *iii*) performance studies of analysis operations.

Works of the same authors but with very similar content were intentionally classified and evaluated as separated primary studies for a more rigorous analysis. Later, in the presentation of results, we grouped those works with no major differences.

Some related works were discarded to keep the size and complexity of the review in a manageable level, namely: *i*) papers proposing operations where the input feature model is modified by returning a new feature model, i.e. only operations proposing information extraction where considered, *ii*) papers presenting any application of the analysis of feature models rather than proposing new analyses, and *iii*) papers dealing with the analysis of other kinds of variability models like OVM [66], decision models [71] and further extensions of feature models like *probabilistic feature models* [31].

3.5. Data collection

All 72 candidate papers identified in our search process were analysed and classified into a research technical report [12] (more than 100 pages). The data extracted from each paper was:

- Full reference including its source, e.g. conference.
- Brief summary of the paper remarking its main contribution in terms of analysis.
- Analysis operations proposed and whether the authors provide support for them (addressing RQ1 and RQ2)
- Paradigm used for the automation (addressing RQ2).
- Solver or tool used to automated the analyses (addressing RQ2)

- Feature model notation supported (addressing RQ2).
- Whether the approach is formalized (addressing RQ1).
- Performance evaluation (addressing RQ2).

Based on the the information obtained, we selected the primary studies according to our inclusion and exclusion criteria. All papers were read at least two times by two different authors to reduce misunderstandings or missing information.

We contacted the first author of each paper and sent them the paper to contrast that the information collected was correct. Some minor changes were proposed and corrected.

3.6. Structure of the review

To present the data, we used a concept–centric approach in contrast to author–centric. That is, we focus on the main concepts that determine the comparative framework of the review (i.e. research questions) rather than simply present a summary of each article. To make the transition from author–centric to concept–centric we followed the recommendation given in [98] and compiled a concept table with the information described in previous section as we read each article. Author–centric tables are available in the technical report [12]. Concept–centric tables are presented in this paper (see Section 5.3 and 6).

4. Conceptual framework

In this section, we propose a conceptual framework that we provide after extracting and synthesizing data from primary studies. This framework attempts to provide a high-level vision of the analysis process and clarify the meaning of various usually ambiguous terms found in the literature. This is the result of the common concepts and practices identified in the primary studies of our review.

As a result of the literature review we found that the automated analysis of feature models can be defined as the *computer–aided extraction of information from feature models*. This extraction is mainly carried out in a two–step process depicted in Figure 4. Firstly, the input parameters (i.e. feature model) are translated into a specific representation or paradigm such as propositional logic, constraint programming, description logic or ad–hoc data structures. Then, off–the–shelf solvers or specific algorithms are used to automatically analyse the representation of the input parameters and provide the result as an output.

The analysis of feature models is performed in terms of *analysis operations*. An operation takes a set of parameters as input and returns a result as output. In addition to feature models, typical input and output parameters are:

- *Configuration*. Given a feature model with a set of features F, a configuration is a 2–tuple of the form (S,R) such that $S, R \subseteq F$ being S the set of features to be selected and R the set of features to be removed such that $S \cap R = \varnothing$.
  - *Full configuration*. If $S \cup R = F$ the configuration is called *full configuration*.
  - *Partial configuration*. if $S \cup R \subset F$ the configuration is called *partial configuration*

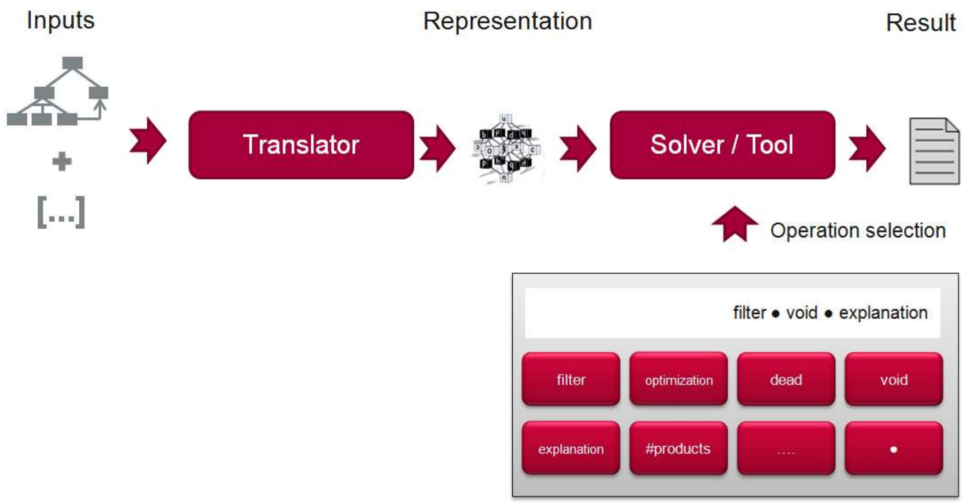


Figure 4: Process for the automated analysis of feature models

As an example, consider the model in Figure 1 and the full (FC) and partial (PC) configurations described below:

```
FC = ({MobilePhone,Calls,Screen,Colour},
      {GPS,Basic,High resolution,Media,Camera,MP3})

PC = ({MobilePhone,Calls,Camera},{GPS})
```

- *Product.* A product is equivalent to a full configuration where only selected features are specified and omitted features are implicitly removed. For instance, the following product is equivalent to the full configuration described above:

```
P = {MobilePhone,Calls,Screen,Colour}
```

5. Analysis operations and automated support

5.1. Analysis operations on feature models

In this section, we answer *RQ1 : What operations of analysis on feature models have been proposed?* For each operation, its definition, an example and possible practical applications are presented.

5.1.1. Void feature model

This operation takes a feature model as input and returns a value informing whether such feature model is void or not. A feature model is *void* if it represents no products. The reasons that may make a feature model to be void are related with a wrong usage of cross–tree constraints, i.e. feature models without cross-tree constraints can not be void.

As an example, Figure 5 depicts a void feature model. Constraint *C-1* makes not possible the selection of the mandatory features *B* and *C*, what adds a contradiction to the model because both features are mandatory.

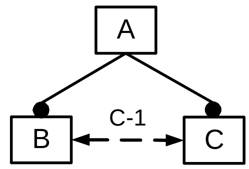


Figure 5: A void feature model

The automation of this operation is specially helpful when debugging large scale feature models in which the manual detection of errors is recognized to be an error-prone and time–consuming task [4, 48, 80]. This operation is also referred by some authors as *"model validation"*, *"model cosnsistency checking"*, *"model satisfiability checking"*, *"model solvability checking"* and *"model constraints checking"*.

5.1.2. Valid product

This operation takes a feature model and a product (i.e. set of features) as input and returns a value determining whether the product belongs to the set of products represented by the feature model or not. For instance, consider the products $P1$ and $P2$, described below, and the feature model of Figure 1.

```
P1={MobilePhone,Screen,Colour,Media,MP3}
P2={MobilePhone,Calls,Screen,High resolution,GPS}
```

Product $P1$ is not valid since it does not include the mandatory feature *Calls*. On the other hand, product $P2$ does belong to the set of products represented by the model.

This operation may be helpful for software product line analysts and managers to determine whether a given product is available in a software product line. This operation is sometimes also referred as *"valid configuration checking"*,*"valid single system"*, *"configuration consistency"*, *"feature compatibility"*, *"product checking"* and *"product specification completeness"*.

5.1.3. Valid partial configuration

This operation takes a feature model and a partial configuration as input and returns a value informing whether the configuration is valid or not, i.e. a partial configuration is valid if it does not include any contradiction. Consider as an example the partial configurations $C1$ and $C2$, described below, and the feature model of Figure 1.

```
C1 = ({MobilePhone,Calls,Camera}, {GPS,High resolution})
C2 = ({MobilePhone,Calls,Camera}, {GPS})
```

$C1$ is not a valid partial configuration since it selects support for camera and remove high resolution screen what is explicitly required by the software product line. $C2$ does not include any contradiction and therefore could still be extended to a valid full configuration.

This operation results helpful during the product derivation stage to give the user an idea about the progress of the configuration. A tool implementing this operation could inform the user as soon as a configuration becomes invalid, thus saving time and effort.

5.1.4. All products

This operation takes a feature model as input and returns all the products represented by the model. For instance, the set of all the products of the feature model presented in Figure 1 is detailed below:

```
P1 = {MobilePhone,Calls,Screen,Basic}
P2 = {MobilePhone,Calls,Screen,Basic,Media,MP3}
P3 = {MobilePhone,Calls,Screen,Colour}
P4 = {MobilePhone,Calls,Screen,Colour,GPS}
P5 = {MobilePhone,Calls,Screen,Colour,Media,MP3}
P6 = {MobilePhone,Calls,Screen,Colour,Media,MP3,GPS}
P7 = {MobilePhone,Calls,Screen,High resolution}
P8 = {MobilePhone,Calls,Screen,High resolution,Media,MP3}
P9 = {MobilePhone,Calls,Screen,High resolution,Media,MP3,Camera}
P10 = {MobilePhone,Calls,Screen,High resolution,Media,Camera}
P11 = {MobilePhone,Calls,Screen,High resolution,GPS}
P12 = {MobilePhone,Calls,Screen,High resolution,Media,MP3,GPS}
P13 = {MobilePhone,Calls,Screen,High resolution,Media,Camera,GPS}
P14 = {MobilePhone,Calls,Screen,High resolution,Media,Camera,MP3,GPS}
```

This operation may be helpful to identify new valid requirements combinations not considered in the initial scope of the product line. The set of products of a feature model is also referred in the literature as *"all valid configurations"* and *"list of products"*.

5.1.5. Number of products

This operation returns the number of products represented by the feature model received as input. Note that a feature model is void iff the number of products represented by the model is zero. As an example, the number of products of the feature model presented in Figure 1 is *14*.

This operation provides information about the flexibility and complexity of the software product line [10, 30, 92]. A big number of potential products may reveal a more flexible as well as more complex product line. The number of products of a feature models is also referred in the literature as *"variation degree"*.

5.1.6. Filter

This operation takes as input a feature model and a configuration (potentially partial) and returns the set of products including the input configuration that can be derived from the model. Note that this operation does not modify the feature model but filter the features that are considered.

For instance, the set of products of the feature model in Figure 1 applying the partial configuration $(S, R) = (\{Calls, GPS\}, \{Colour, Camera\})$, being S the set of features to be selected and R the set of features to be removed, is:

```
P1 = {MobilePhone,Calls,Screen,High resolution,GPS}
P2 = {MobilePhone,Calls,Screen,High resolution,Media,MP3,GPS}
```

Filtering may be helpful to assist users during the configuration process. Firstly, users can filter the set of products according to their key requirements. Then, the list of resultant products can be inspected to select the desired solution [30].

5.1.7. Anomalies detection

A number of analysis operations address the detection of anomalies in feature models i.e. undesirable properties such as redundant or contradictory information. These operations take a feature model as input and return information about the anomalies detected. We identified five main types of anomalies in feature models reported in the literature. These are:

Dead features. A feature is *dead* if it cannot appear in any of the products of the software product line. Dead features are caused by a wrong usage of cross–tree constraints. These are clearly undesired since they give the user a wrong idea of the domain. Figure 6 depicts some typical situations that generate dead features.

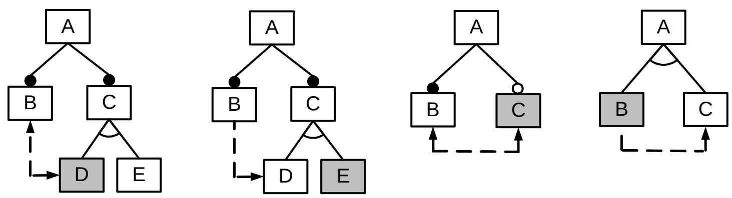


Figure 6: Common cases of dead features. Grey features are dead

Conditionally dead features. A feature is *conditionally dead* if it becomes dead under certain circumstances (e.g. when selecting another feature) [45]. Both, unconditional and conditional dead features are often referred in the literature as *"contradictions"* or *"inconsistencies"*. In Figure 7 feature B becomes dead whenever feature D is selected. Note that, with this definition, features in an alternative relationship are conditionally dead.

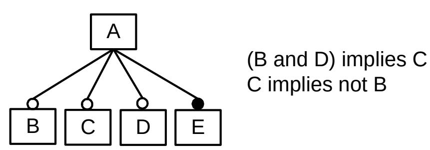


Figure 7: An example of a conditionally dead feature

False optional features. A feature is *false optional* if it is included in all the products of the product line despite not being modelled as mandatory. Figure 8 depicts some examples of false optional features.

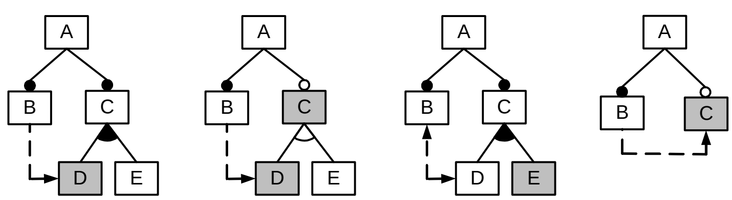


Figure 8: Some examples of false optional features. Grey features are false optional

Wrong cardinalities. A group cardinality is wrong if it cannot be instantiated [84]. These appear in cardinality–based feature models where cross–tree constraints are involved. An example of wrong cardinality is provided in Figure 9. Notice that features B

and D excludes each other and therefore the selection of three subfeatures, as stated by the group cardinality, is not possible.

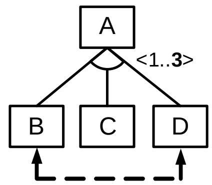


Figure 9: An example of wrong cardinality

Redundancies. A feature model contains redundancies when some semantic information is modelled in multiple ways [94]. Generally, this is regarded as a negative aspect since it may decrease the maintainability of the model. Nevertheless, it may also be used as a means of improving readability and understandability of the model. Figure 10 depicts some examples of redundant constraints in feature models.

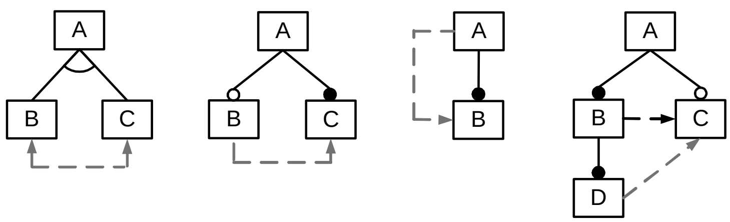


Figure 10: Some examples of redundancies. Gray constraints are redundant

5.1.8. Explanations

This operation takes a feature model and an analysis operation as inputs and returns information (so-called explanations) about the reasons of *why* or *why not* the corresponding response of the operation [84]. Causes are mainly described in terms of the features and/or relationships involved in the operation and explanations are ofter related to anomalies. For instance, Figure 11 presents a feature model with a dead feature. A possible explanation for the problem would be *"Feature D is dead because of the excludes constraint with feature B"*. We refer the reader to [84] for a detailed analysis on explanation operations.

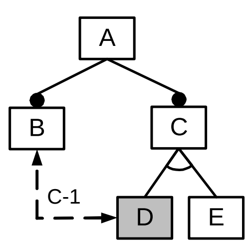


Figure 11: Grey feature is dead because relationship C–1

Explanations are a challenging operation in the context of feature model error analysis, (a.k.a. feature model debugging) [6, 80, 84]. In order to provide an efficient tool support, explanations must be as accurate as possible when detecting the source of an error, i.e. it should be minimal. This become even a more challenging task when considering extended feature models and relationships between feature attributes.

5.1.9. Corrective explanations

This operation takes a feature model and an analysis operation as inputs and returns a set of corrective explanations indicating changes to be made in the original inputs in order to change the output of the operation. In general, a *corrective explanation* provides suggestions to solve a problem, usually once this has been detected and explained.

For instance, some possible corrective explanations to remove the dead feature in Figure 11 would be *"remove excludes constraint C-1"* or *"model feature B as optional"*. This operation is also referred in the literature as *"corrections"*.

5.1.10. Feature model relations

These operations take two different feature models as inputs and returns a value informing how the models are related. The set of features in both models are not necessarily the same. These operations are helful to determine how a model has evolved over time. Thüm *et al.* [79] classify the possible relationships between two feature models as follows:

Refactoring. A feature model is a refactoring of another one if they represents the same set of products while having a different structure. For instance, model in Figure 12(b) is a refactoring of model in Figure 12(a) since they represent the same products i.e. {{*A*,*B*},{{*A*,*B*,*C*}, {*A*,*B*,*D*},{*A*,*B*,*C*,*D*}}. Refactorings are useful to restructure a feature model without changing its semantics. When this property is fulfilled the models are often referred as *"equivalent"*.

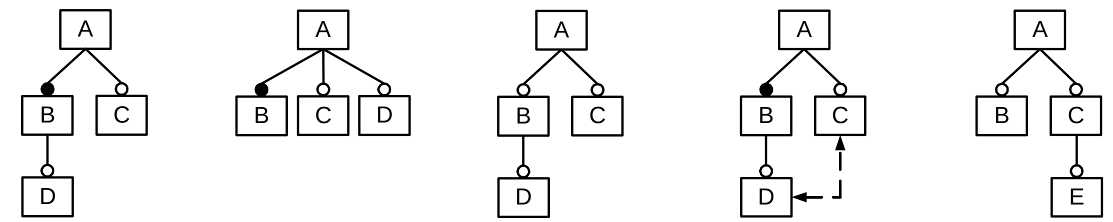
Generalization. A feature model, F, is a generalization of another one, G, if the set of products of F maintains and extends the set of products of G. For example, feature model in Figure 12(c) is a generalization of the model in Figure 12(a) because it add a new product ({*A*}) and does not remove any existing one. Generalization occur naturally while extending a software product line.

Specialization. A feature model, F, is a specialization of another one, G, if the set of products of F is a subset of the set of products of G. For example, Figure 12(d) depicts a specialization of the model in Figure 12(a) since it removes a product from the original model ({*A*,*B*,*C*,*D*}) and adds no new ones.

Arbitrary edit. There is no explicit relationship between the input models, i.e. there is not any of the relationships defined above. Models in Figure 12(a) and Figure 12(e) illustrate an example of this. Thüm *et al.* [79] advise avoiding arbitrary edits and replacing these by a sequence of specialization, generalizations and refactorings edits for a better understanding of the evolution of a feature model.

5.1.11. Optimization

This operation takes a feature model and a so-called objective function as inputs and returns the product fulfilling the criteria established by the function. An objective



(a) Original (b) Refactoring (c) Generalization (d) Specialization (e) Arbitrary

Figure 12: Types of relationships between two feature models

function is a function associated with an optimization problem which determines how good a solution is.

This operation is chiefly useful when dealing with extended feature models where attributes are added to features. In this context, operations of optimization may be used to select a set of features maximizing or minimizing the value of a given feature attribute. For instance, mobile phones minimizing connectivity cost in Figure 2 should include support for *USB* connectivity exclusively, i.e. *USB* is the cheapest.

5.1.12. Core features

This operation takes a feature model as input and returns the set of features that are part of all the products in the software product line. For instance, the set of core features of the model presented in Figure 1 is {*MobilePhone,Calls,Screen*}.

Core features are the most relevant features of the software product line since they are supposed to appear in all products. Hence, this operation is useful to determine which features should be developed in first place [81] or to decide which features should be part of the core architecture of the software product line [65].

5.1.13. Variant features

This operation takes a feature model as input and returns the set of variant features in the model [84]. Variant features are those that do not appear in all the products of the software product line. For instance, the set of variant features of the feature model presented in Figure 1 is {*Basic,Colour,High resolution,Media,Camera, MP3,GPS*}.

5.1.14. Atomic sets

This operation takes a feature model as input and returns the set of atomic sets of the model. An *atomic set* is a group of features (at least one) that can be treated as a unit when performing certain analyses. The intuitive idea behind atomic sets is that mandatory features and their parent features always appear together in products and therefore can be grouped without altering the result of certain operations. Once atomic sets are computed, this can be used to create a reduced version of the model by simple replacing each feature by the atomic set that contains it.

Figure 13 depicts an example of atomic sets computation. From the original model 4 atomic sets are derived reducing the number of features from 7 to 4. Note that the reduced model is equivalent to the original one since both represent the same set of products.

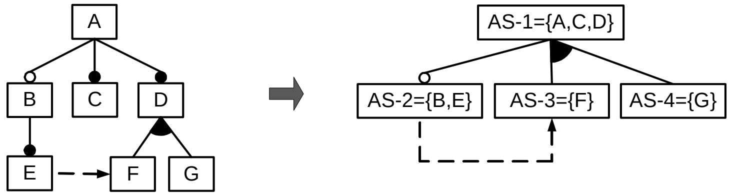


Figure 13: Atomic sets computation

Using this technique, mandatory features are safely removed from the model. This operation is used as an efficient preprocessing technique to reduce the size of feature models prior to their analysis [75, 106].

5.1.15. Dependency analysis

This operation takes a feature model and a partial configuration as input and returns a new configuration with the features that should be selected and/or removed as a result of the propagation of constraints in the model [60]. As an example, consider the input and output configurations described below and the model in Figure 1.

```
Input = ({MobilePhone,Calls,Camera}, {MP3})
Output = ({MobilePhone,Calls,Camera,Media,Screen,High resolution}, {MP3,Basic,Colour})
```

Features *Screen* and *High resolution* are added to the configuration to satisfy the requires constraint with *Camera*. *Media* is also included to satisfy the parental relationship with *Camera*. Similarly, features *Basic* and *Colour* are removed to fulfil the constraints imposed by the alternative relationship.

This operation is the basis for constraint propagation during the interactive configuration of feature models [60]

5.1.16. Multi–step configuration

A multi–step configuration problem is defined as the process of producing a series of intermediate configurations, i.e. a configuration path, going from a feature model configuration to another [101]. An operation of analysis solving a multi–step configuration problem takes as input a feature model, an initial configuration, a desired final configuration, a number of steps in the configuration path K, a global constraint that can not be violated (usually referred to feature attributes) and a function determining the cost to transition from one configuration in step T to another in step U. As a result, the operation provides an ordered list of K configurations that determines the possible steps that can be taken to go from the initial configuration to the desired final configuration without violating the feature model and global constraints. For a detailed example and a rigorous definition of the operation we refer the reader to [101].

5.1.17. Other operations

In this section, we group those operations that perform some computation based on the values of previous operations. We also classify in this group those analysis operations proposed as part of other algorithms.

Homogeneity. This operation takes a feature model as input and returns a number that provides an indication of the degree to which a feature model is homogeneous [40]. An more homogeneous feature model would be one with few features that are unique in one product (i.e. a unique feature apprears only in one product) while a less homogeneous one would be one with a lot of unique features. According to [40] it is calculated as follows:

$$Homogeneity = 1 - \frac{\#uf}{\#products}$$

$\#uf$ is the number of features that are unique in one product and $\#products$ denotes the total number of products represented by the feature model. The range of this indicator is [0,1]. If all the products have unique features the indicator is 0 (lowest degree of homogeneity). If there is no unique features, the indicator is 1 (highest degree of homogeneity).

Commonality. This operation takes a feature model and a configuration as inputs and returns the percentage of products represented by the model including the input configuration. An as example, consider the partial configurations described below and the model in Figure 1:

```
C1 = {{Calls}, {}}
C2 = {{Calls},{MP3}}
```

The commonality of both configurations is calculated as follows:

$$Commonality(C1) = \frac{|filter(FM, \{\{Calls\}, \{\}\})|}{\#products(FM)} = \frac{14}{14} = 1$$

$$Commonality(C2) = \frac{|filter(FM, \{\{Calls\}, \{MP3\}\})|}{\#products(FM)} = \frac{7}{14} = 0.5$$

The range of this indicator is [0,1]. Configuration $C1$ appear in the 100% of the products whereas $C2$ is included only in the 50% of them.

This operation may be used to prioritize the order in which the features are going to be developed [81] or to decide which features should be part of the core architecture of the software product line [65].

Variability factor. This operation takes a feature model as input and returns the ratio between the number of products and 2^n where n is the number of features considered. In particular, 2^n is the potential number of products represented by feature model assuming that any combination of features is allowed. The root and non-leaf features are often not considered. As an example, the variability of the feature model presented in Figure 1 taking into account only leaf features is:

$$\frac{N.Products}{2^n} = \frac{14}{2^7} = 0.0625$$

An extremely flexible feature model would be one with all its features as optionals. For instance, the feature model of Figure 14 has the following variability factor:

$$\frac{N.Products}{2^n} = \frac{8}{2^3} = 1$$

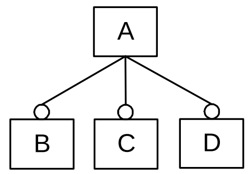


Figure 14: Sample feature model with three optional features

The range of this indicator would depend on the features considered to calculate the factor. The feature model variability can be used to measure the flexibility of the feature model. For instance, a small factor means that the number of combinations of features is very limited compared to the total number of potential products.

Degree of orthogonality. This operation takes a feature model and a subtree (represented by its root feature) as input and returns their degree of orthogonality. Czarnecki *et al.* [30] defines the *degree of orthogonality* as the ratio between the total number of products of the feature model and the number of products of the subtree. Only local constraints in the subtree are considered for counting the products. For instance, the formula below shows the degree of orthogonality for the subtree *Screen* in Figure 1.

$$Orthogonality(Screen) = \frac{14}{3} = 4.66$$

The range of this indicator is [0,1]. A high degree of orthogonality indicates that decisions can be taken locally without worrying about the influence in the configuration of other parts of the tree [30].

Extra Constraint Representativeness (ECR). This operation takes a feature model as input and returns the degree of representativeness of the cross-tree constraints in the tree. Mendonça *et al.* [62, 61] defines the *Extra Constraint Representativeness (ECR)* as the ratio of the number of features involved in cross-tree constraints (repeated features counted once) to the number of features in the feature tree. For instance, ECR in Figure 1 is calculated as follows:

$$ECR = \frac{4}{10} = 0.4$$

The range of this indicator is [0,1]. This operation has successfully used to design and evaluate heuristics for the automated analysis of feature models [62].

Lowest Common Ancestor (LCA). This operation takes a feature model and a set of features as input and returns a feature that is the lowest common ancestor of the input features. Mendonça *et al.* [62] defines the *Lowest Common Ancestor (LCA)* of a set of features, $LCA(FM, \{f_1, ..., f_n\})$, as the shared ancestor that is located farthest from the root. In Figure 1, $LCA(FM, \{Basic, Camera\}) = MobilePhone$.

Root features. This operation takes a feature model and a set of features as inputs and returns a set of features that are the *roots* features in the model. Given

$l = LCA(FM, \{f_1, ..., f_n\})$, Mendonça *et al.* [62] defines the roots of a set of features, $Roots(FM, \{f_1, ..., f_n\})$ as the subset of child features of l that are ancestor of $f_1, ..., f_n$. In Figure 1, $Roots(FM, \{Basic, Camera\}) = \{Media, Screen\}$.

5.2. Automated support

Previously, we have presented the different analysis operations that we found in the literature. In this section, we address *RQ2: What kind of automated support has been proposed and how it is performed?* To answer this question, we classified the primary studies in four different groups according to the logic paradigm or method used to provide the automated support. In particular, we next present the group of approaches using *Propositional Logic*(PL), *Constraint Programming*(CP), *Description Logic*(DL), and other contributions based not classified in the former groups proposing on ad-hoc solutions, algorithms or paradigms.

5.2.1. Propositional logic based analyses

A *propositional formula* consists of a set of primitive symbols or variables and a set of logical connectives constraining the values of the variables, e.g. $\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$.

A *SAT solver* is a software package that takes as input a propositional formula and determines if the formula is satisfiable, i.e. there is a variable assignment that makes the formula evaluates to true. Input formulas are usually specified in *Conjunctive Normal Form* (CNF). CNF is a standard form to represent propositional formulas that is used by most of SAT solvers where only three connectives are allowed: $\neg, \wedge, \vee$. It has been proved that every propositional formula can be converted into an equivalent CNF formula [25]. SAT solving is a well known NP-complete problem [25], however, current SAT solvers can deal with big problems where in most of the cases the performance is not an issue [58].

Similarly, a *Binary Decision Diagram* (BDD) solver is a software package that takes a propositional formula as input (not necessarily in CNF) and translates it into a graph representation (the BDD itself) which allows determining if the formula is satisfiable and providing efficient algorithms for counting the number of possible solutions [19]. The size of the BDD is crucial because it can be exponential in the worst case. Although it is possible to find a good variable ordering that reduces the size of the BDD, the problem of finding the best variable ordering remains NP-complete [17].

The mapping of a feature model into a propositional formula can change depending on the solver that is used later for analysis. In general, the following steps are performed: *i*) each feature of the feature model maps to a variable of the propositional formula, *ii*) each relationship of the model is mapped into one or more small formulas depending on the type of relationship, in this step some auxiliary variables can appear, *iii*) the resulting formula is the conjunction of all the resulting formulas of step *ii* plus and additional constraint assigning true to the variable that represents the root, i.e. $root \iff true$.

Concrete rules for translating a feature model into a propositional formula are listed in Figure 15. Also, the mapping of our running example of Figure 1 is presented. We may mention that the mapping of the propositional formulas listed in Figure 15 into CNF is straightforward (see [25]).

There are some works in the literature that propose the usage of propositional formulas for the automated analysis of feature models (see Table 3). In these studies the analysis is performed in two steps. Firstly, the feature model is translated into a propositional formula. Then, an off–the–shelf solver is used to automatically analyse the formula and subsequently the feature model. A summary of the solvers used for analysis is shown in Table 1.

Tool	Proposals
SAT Solver [16]	[4, 13, 15, 61, 75, 79]
Alloy [35]	[41, 78]
BDD Solver [99]	[13, 15, 30, 62, 75, 90, 91, 107, 104]
SMV [36]	[105, 106]
Not specified	[56, 57]

Table 1: Propositional logic based tools used for FM analysis

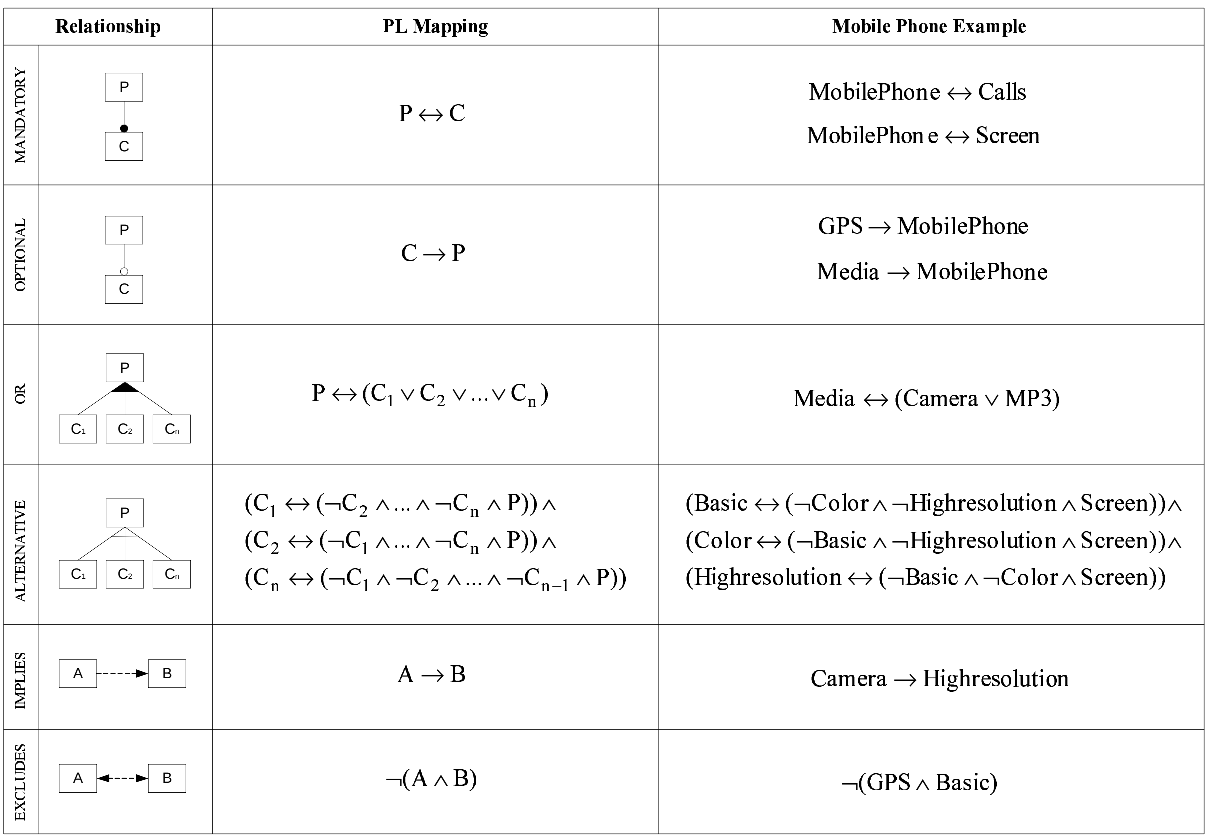


Figure 15: Mapping from feature model to propositional logic

To underline the most important contributions in terms of innovations with respect to prior work we may mention the following studies: Mannion *et al.* [56, 57] was the first to connect propositional formulas and feature models. Zhang et *al.* [106] reported a method to calculate *atomic sets*, later explored by Segura [75]. Batory [4] shows the connections among grammars, feature models and propositional formulas, this was the first time that a SAT solver was proposed to analyse feature models. In addition, a *Logic*

Truth Maintenance System (a system that maintains the consequences of a propositional formula) was designed to analyse feature models. Sun et *al.* [78] propose using Z, a formal specification language, to provides semantics to feature models. Alloy was used to implement those semantics and analyse feature models. Benavides *et al.*[13, 15, 75] propose using a multi–solver approach where different solvers are used (e.g. BDD or SAT solvers) depending on the kind of analysis operations to be performed. For instance, they suggest that BDD solvers seems to be more efficient in general than SAT solvers for counting the number of products of a feature model. Mendonca et *al.* [62] used also BDDs for analysis and compared different classical heuristics found in the literature for variable ordering of BDDs with new specific heuristics for analysis of BDDs representing feature models. They experimentally showed that existing BDD heuristics fails to scale for large feature models meanwhile their novel heuristics can scale for models with up to 2,000 features. Thüm et *al.* [79] present an automated method for classifying feature model edits, i.e. changes in an original feature model, according to a taxonomy. The method is based on propositional logic algorithms using a SAT solver and constraint propagation algorithms. Yan *al.* [104] propose an optimization method to reduce the size of the logic representation of the feature models by removing irrelevant constraints. Mendonca et. *al.* [61] shows by means of an experiment that the analysis of feature models with similar properties to those found in the literature using SAT solvers is computationally affordable.

5.2.2. Constraint programming based analyses

A *Constraint Satisfaction Problem* (CSP) [86] consists of a set of variables, a set of finite domains for those variables and a set of constraints restricting the values of the variables. *Constraint programming* can be defined as the set of techniques such as algorithms or heuristics that deal with CSPs. A CSP is solved by finding states (values for variables) in which all constraints are satisfied. In contrast to propositional formulas, CSP solvers can deal not only with binary values (true or false) but also with numerical values such as integers or intervals.

A CSP solver is a software package that takes a problem modelled as a CSP and determines whether it exists any solution for the problem. From a modelling point of view, CSP solvers provide a richer set of modelling elements in terms of variables (e.g. sets, finite integer domains, etc.) and constraints (not only propositional connectives) than propositional logic solvers.

The mapping of a feature model into CSP can change depending on the concrete solver that is used later for the analysis. In general, the following steps are performed: *i*) each feature of the feature model maps to a variable of the CSP with a domain of 0..1 or TRUE, FALSE, depending on the kind of variable supported by the solver, *ii*) each relationship of the model is mapped into a constraint depending on the type of relationship, in this step some auxiliary variables can appear, *iii*) the resulting CSP is the one defined by the variables of steps *i* and *ii* with the corresponding domains and a constraint that is the conjunction of all precedent constraints plus and additional constraint assigning true to the variable that represents the root, i.e. $root \iff true$ or $root == 1$, depending on the variables domains.

Concrete rules for translating a feature model into a CSP are listed in Figure 16. Also, the mapping of our running example of Figure 1 is presented.

There are some works in the literature that propose the usage of constraint programming for the automated analysis of feature models (see Table 3). Analyses are performed in two steps. Firstly, the feature model is translated into a CSP. Then, an off–the–shelf solver is used to automatically analyse the CSP and subsequently the feature model. A summary of the solvers used for analysis is shown in Table 2.

Tool	Proposals
JaCoP [37]	[13, 14, 15, 75]
Choco [33]	[14, 103, 101]
OPL studio [47]	[9, 10, 11]
GNU Prolog [34]	[38]
Not specified	[82, 80]

Table 2: CSP based tools used for FM analysis

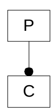
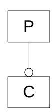
Benavides *et al.* were the first authors proposing the usage of constraint programming for analyses on feature models [9, 10, 11]. In those works, a set of mapping rules to translate feature models into a CSP were provided. Benavides *et al.* proposals provide support for the analysis of extended feature models (i.e. including feature attributes) and the operation of optimization. Authors also provide tool support [15, 83] and they have compared the performance of different solvers when analysing feature models [14, 13, 75]. Trinidad *et al.* [82, 80] focus on the detection and explanation of errors on feature models based on Reiter's theory of diagnosis [68] and constraint programming. Djebbi *et al.* [38] propose a method to extract information from feature models in terms of queries. A set of rules to translate feature models to boolean constraints are given. Additionally, they also describe a tool under development enabling the analysis of feature models using constraint programming. White *et al.* [103] propose a method to detect conflicts in a given configuration and propose changes in the configuration in terms of features to be selected or deselected that remedy the problem. Their technique is based on translating a feature model into a CSP adding some extra variables in order to detect and correct the possible errors after applying optimization operations. In [101], White *et al.* provide support for the analysis of multi–step configuration problems.

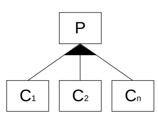
5.2.3. Description logic based analyses

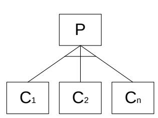
Description logics are a family of knowledge representation languages enabling the reasoning within knowledge domains by using specific logic reasoners [2]. A problem described in terms of description logic is usually composed by a set of concepts (a.k.a. classes), a set of roles (e.g. properties or relationships) and set of individuals (a.k.a. instances).

A description logic reasoner is a software package that takes as input a problem described in description logic and provides facilities for consistency and correctness checking and other reasoning operations.

We found four primary studies proposing the usage of description logic to analyse feature models. Wang *et al.* [96] were the first to enable the automated analysis of

Relationship		CSP Mapping	Mobile Phone Example
MANDATORY		P = C	Mobilephone = Calls Mobilephone = Screen
OPTIONAL		if (P = 0) C = 0	if (Mobilephone = 0) GPS = 0 if (Mobilephone = 0) Media = 0

OR		if (P > 0)
		Sum(C1,C2,...Cn) in {1..n}
		else
		C1= 0, C2=0,…., Cn=0
			if (Media > 0)
			Sum(Camera,MP3) in {1..2}
			else
			Camera = 0, MP3 = 0

ALTERNATIVE		if (P > 0)
		Sum(C1,C2,...Cn) in {1..1}
		else
		C1= 0, C2=0,…., Cn=0
			if (Screen > 0)
			Sum(Basic,Colour,High resolution) in {1..1}
			else
			Basic = 0,Colour = 0, High resolution = 0

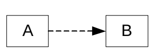
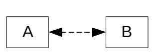
REQUIRES		if (A > 0) B>0	if (Camera > 0) High resolution > 0
EXCLUDES		if (A > 0) B=0	if (GPS > 0) Basic = 0

Figure 16: Mapping from feature model to CSP

feature models using description logic. In their work, the authors introduce a set of mapping rules to translate feature models into OWL-DL ontologies [32]. OWL-DL is an expressive yet decidable sub language of OWL [32]. Then, the authors suggest using description logic reasoning engines such as RACER[67] to perform automated analysis over the OWL representations of the models. In [97], the authors extend their previous proposal [96] with support for explanations by means of an OWL debugging tool. Fan *et al.* [39] also propose translating feature models into description logic and using reasoners such as RACER to perform their analyses. In [1], Abo *et al.* propose using semantic web technologies to enable the analyses. They use OWL for modelling and the Pellet [22] reasoner for the analysis.

5.2.4. Other studies

There are some primary studies that are not classified in the former groups, namely: *i*) studies in which the conceptual logic used is not clearly exposed and *ii*) studies using ad–hoc algorithms, paradigms or tools for analysis.

Kang et *al.* did an explicit mention to automated analysis of feature models in the original FODA report [48, pag. 70]. A prolog–based prototype is also reported. However, not detailed information is provided to replicate their prolog coding. After the FODA report, Deursen *et al.* [92] were the first authors proposing some kind of automated support for the automated analysis of feature models. In their work, they propose a textual feature diagram algebra together with a prototype implementation using the ASF+SDF Meta-Environment [52]. Von der Massen *et al.* [93] present Requiline, a requirement engineering tool for software product lines. The tool is mainly implemented using a relational data base and ad–hoc algorithms. Later, Von der Massen *et al.* [95] propose a method to calculate a rough approximation of the number of products of a feature model, what they call *variation degree*. The technique is described using mathematical expressions. In [3], Bachmeyer*et al.* present *conceptual graph feature models*. Conceptual graphs are a formalism to express knowledge. Using this transformation, they provide an algorithm that is used to compute analysis. Hemakumar [45] proposes a method to statically detect conditional dead features. The method is based on model checking techniques and incremental consistency algorithms. Mendonça *et al.* [59, 60] study dependencies among feature models and cross–tree constraints using different techniques obtaining a noticeable improvement in efficiency. Gheyi *et al.* [42] present a set of algebraic laws in feature models to check configurability of feature model refactorings. They use the PVS tool to do some analysis although this is not the main focus of the paper. White *et al.* [100] present an extension of their previous work [102]. The same method is presented but giving enough details to make it reproducible since some details were missed in their previous work. The method is called *Filtered Cartesian Flattering* which maps the problem of optimally selecting a set of features according to several constraints to a *Multi–dimensional Multi–choice Knapsack Problem* and then they use several existing algorithms to this problem that perform much faster while offering an approximate answer. Van den Broek *et al.* [88] propose transforming feature models into generalised feature trees and computing some of their properties. A *generalised feature tree* is a feature model in which cross-tree constraints are removed and features can have multiple occurrences. Some algorithms and an executable specification in the functional programming language

Miranda is provided. The strength of their proposal lies in the efficiency of the analysis operation. Fernandez *et al.* [40] propose an algorithm to compute the total number of products on what they call *Neutral Feature Trees*, these trees allow complex cross-tree constraints. Computing the total number of products the authors are also able to calculate the *homogeneity* of a feature tree as well as the *commonality* of a given feature. They finally compare the computational complexity of their approach with respect to previous work.

5.3. Summary and analysis of operations and support

A summary of the analysis operations (RQ1) and automated support (RQ2) identified in the literature is shown in Table 3. Operations are listed horizontally and ordered by the total number of papers mentioning it. Primary studies are listed vertically. Related works of the same author are grouped in a single column. Primary studies are grouped according to the paradigm they use for the analyses as follows: *i*) Propositional Logic (PL), *ii*). Constraint Programming (CP) *iii*) Description Logic (DL), *iv*) works that integrate more than one paradigm and/or solver (Multi), *v*) studies that use their own tools not categorized in the former groups (Others), and *vi*) proposals that present different operations but do not provide automated support for them (No support).

The cells of the matrix indicates the information about a primary study in terms of operations supported. Cells marked with '+' indicate that the proposal of the column provides support for the operation of the row. We use the symbol '$\sim$' for proposals with no automated support for the corresponding operation but explicit definition of it. We also remark the primary study that first proposed an operation using the symbols '$\oplus$' (when support is provided) and '$\ominus$' (when no support is provided). To fully answer the research questions we also extracted some additional information about different aspects of the primary studies, namely: *i*) feature model notations supported: 'B' (basic feature model), 'C' (cardinality–based feature model) *ii*) whether the approach support extended feature models or not, *iii*) whether the approach is described formally. This information is also reported in the final rows of Table 3.

Table 4 depicts a chronological view of the data presented in Table 3. More specifically, it shows the amount of references to operations, notation, formalization and kind of automated support found in the literature for each year. Vertically, we list all the years where primary studies were published. Last column indicates the total number of primary studies referring the operation, the notation of feature models, the formalization provided and the type of automated support used for analysis. The table also shows the number of new operations proposed by each year.

As illustrated in Tables 3 and 4, there are 11 out of 30 operations that only appeared in one primary study. Likewise, 6 operations were treated in more than 10 studies from which 4 were already mentioned in the original FODA report back in 1990 [48]. This denotes, in our opinion, that FODA authors were quite visionary in predicting the importance of automated analysis on feature models and pinpointing some of most referred operations. We may remark that 11 new operations were proposed in the last two years of our study and 22 of them were referred in 2009 suggesting that the analysis of feature models is an active research field.

	Batory [4]	Czarnecki *et al.* [30]	Gheyi *et al.* [41]	Mannion *et al.* [56, 57]	Mendonca *et al.* [62]	Mendonca *et al.* [61]	Sun *et al.* [78]	Thüm *et al.* [79]	van der Storm [90, 91]	Zhang *et al.* [106, 105]	Zhang *et al.* [107]	Yan *et al.* [104]	Benavides *et al.* [9, 10, 11]	Benavides *et al.* [14]	Djebii *et al.* [38]	Trinidad *et al.* [82, 80]	White *et al.* [103]	White *et al.* [101]	Abo *et al.* [1]	Fan *et al.* [39]	Wang *et al.* [96, 97]	Benavides *et al.* [13]	Benavides *et al.* [15]	Segura [75]	Bachmeyer *et al.* [3]	Cao *et al.* [20]	Fernandez *et al.* [40]	Hemakumar [45]	Gheyi *et al.* [42]	Kang *et al.* [48]	Mendonca *et al.* [60]	Osman *et al.* [63, 64]	Salinesi *et al.* [70]	Van den Broek *et al.* [88]	Van Deursen *et al.* [92]	Von der Massen *et al.* [93]	Von der Massen *et al.* [95]	White *et al.* [102, 100]	Batory *et al.* [6]	Schobbens *et al.* [46, 72, 73]	Trinidad *et al.* [84]	Von der Massen *et al.* [94]
	PL												CP						DL			Multi			Others														No support			
Void feature model	+	+		+		+	+		+	+	+	+	+	+	+	+				+	+	+	+	+		+			+	⊕		+	+	+	+	+				~	~	
#Products		+		⊕									+	+	+							+	+	+			+							+	⊕		+				~	
Dead features		~				+				+	+	+				+			+											⊕		+	+	+					~		~	~
Valid product	+	+	+	+			+										+				+				+					⊕						+			~	~	~	
All products	+		+	⊕			+						+										+			+								+	⊕						~	
Explanations	+	~					+									+			+		+									⊕		+		+					~		~	
Refactoring			+				⊕	+													+								+											~	~	
Optimization													⊕		+																	+						+	~		~	
Commonality													⊕			+							+				+														~	
Filter		+											⊕		+																			+							~	
Valid partial configuration	+	+							+																					⊕											~	
Atomic sets					+					⊕														+							+											
False optional features										⊕	+					+																									~	⊖
Corrective explanations																	+															+										⊖
Dependency analysis																														⊕	+											
ECR					⊕	+																																				
Generalization			⊕					+																																		
Core features						+																																			⊖	
Variability factor													⊕																												~	
Arbitrary edit								+																																		
Conditional dead features																												+														
Homogeneity																											+															
LCA					+																																					
Muti–step configuration																		+																								
Roots features					+																																					
Specialization								+																																		
Degree of orthogonality		~																																								
Redundancies																																										~
Variant features																																									~	
Wrong cardinalities																																									~	
Feature model notation	B	C	B	B	B	B	B	B	B	B	C	B	B	C	C	B	B	B	B	B	B	B	C	B	B	B	C	B	B	B	C	C	C	B	B	B	B	B	B	C	C	B
Extended feature model													+		+		+	+	+											+		+						+	+		+	
Formalization			+		+	+	+	+		+	+		+			+		+		+					+		+		+			+					+			+		

+ Supported ~ No support ⊕ Supported(first reference) ⊖ No support (first reference) B Basic feature model C Cardinality–based feature models

Table 3: Summary of operations of analysis and support

	1990	2002	2003	2004	2005	2006	2007	2008	2009	Total
	Operations									
Void feature model	+	+	+	+	+	+	+	+	+	35
#Products		+	+	+	+	+	+	+	+	16
Dead features	+			+	~	+		+	+	17
Valid product	+	+	+		+	+	+	+	~	17
All products		+	+	+	+	+	+		+	13
Explanations	+				+	~	+	+	+	13
Refactoring					+	+	+	+	+	9
Optimization				+	+	~	+	+	+	9
Commonality					+	+	+		+	6
Filter				+	+		+		+	7
Valid partial configuration	+			+	+				~	5
Atomic sets				+				+		4
False optional features				+		+		+	~	6
Corrective explanations				~				+		3
Dependency analysis	+							+		2
ECR								+	+	2
Generalization						+			+	2
Core features									+	2
Variability					+				~	3
Arbitrary edit									+	1
Conditional dead features								+		1
Homogeneity									+	1
LCA								+		1
Muti–step configuration									+	1
Roots								+		1
Specialization									+	1
Degree of orthogonality					~					1
Redundancies				~						1
Variant features									~	1
Wrong cardinalities									~	1
New operations	6	2	0	6	4	1	0	4	7	30
	Notation and formalization									
Basic FMs	+	+	+	+	+	+	+	+	+	40
Cardinality-based FMs					+	+	+	+	+	13
Extended feature models	+			+	+	+	+	+	+	13
Formalization				+	+	+	+	+	+	22
	Support									
Propositional logic		+	+	+	+	+	+	+	+	18
Constraint programming				+	+	+	+	+	+	12
Description logic					+	+	+		+	4
Others	+	+	+		+		+	+	+	16

(light grey) 1 study, (medium grey) 2-3 studies, (dark grey) >3 studies

Table 4: Number of primary studies referring operations, notations and support for each year

Regarding the notation used, 40 out of 53 primary studies used basic feature model notation for analysis of feature models. However, it seems to be an increasing interest in the analysis of cardinality–based and extended feature models since 2004.

With respect to automated support for analysis, 18 out of 53 studies used propositional logic meanwhile only 4 of them used description logic. Constraint programming was referred in 12 studies leaded by three different groups of authors. We remark that no support for extended feature models was found in the studies using propositional logic. There are also 16 studies proposing ad–hoc solutions and this tendency seems to be in progression in the last years which may suggest that researchers are looking for more specific and efficient algorithms to perform analysis operations.

We also found that there are 22 studies proposing a formal or rigorous definition of analysis operations. This tendency seems to be ascendant since 2004 which may indicate that there is an increasing interest by the research community to accurately define an report analysis operations.

	Batory [4]	Czarnecki *et al.* [30]	Sun *et al.* [78]	Trinidad *et al.* [82, 80]	White *et al.* [103]	Abo *et al.* [1]	Wang *et al.* [96, 97]	Kang *et al.* [48]	Osman *et al.* [63, 64]	Van den Broek *et al.* [88]	Batory *et al.* [6]	Trinidad *et al.* [84]	Von der Massen *et al.* [94]
	PL			CP		DL		Others			No		
Valid product	+	~	+		+		+	+			~	~	
Void feature model	+		+	+			+		+	+		~	
Dead features				+		+				+		~	~
Valid partial configuration	+	~						+				~	
False optional				+								~	~
Dependency analysis								+					
Core features												~	
Optimization												~	
Redundancies													~
Variant features												~	
Wrong cardinalities												~	

Table 5: Summary of the proposals reporting explanations

Explanations are acknowledged to be an important operation for feature model error analysis in the literature [6, 84]. As presented in Sections 5.1.8 and 5.1.9, this operations take as input a feature model and an operation and return as a result the source of the errors in the model and the possible actions to correct them respectively. Table 5 shows a detailed view of the operations that haven been used in explanations and corrective explanations. As illustrated, there are only four operations with support for explanations in more than one study. All logical paradigms have been used for explaining different analysis operations. We found that explanations have been largely studied in related problems in the communities of propositional logic, constraint programming and description logic for years. This has provided researchers with helpful guidelines and methods to assist on the the implementation of explanations in the analysis of feature models. We also remark that all the operations of explanations are referred to the analysis of basic or cardinality–based feature models but we have not found any study dealing with explanations in extended feature models. Only Trinidad *et al.* [84] provided some hints about explaining the optimization operation but no explicit method to support this operations was presented.

6. Performance evaluation

Performance analyses play a key role in the evaluation of the analysis techniques and tools. From the results obtained, the strengths and weaknesses of the proposals are highlighted helping researchers to improve their solutions, identify new research directions and show the applicability of the analysis operations.

Table 6 summarizes the proposals reporting performance results on the analysis of feature models. We consider as performance results any data (e.g. time, memory) suggesting how a proposal behave in practice. Works based on propositional logic, constraint programming and ad–hoc solutions have presented a similar number of performance evaluations meanwhile only one proposal has presented results of description logic based support. Regarding operations, 18 out of 30 analysis operations identified in the literature have been used in performance analyses. However, only 7 of them have been evaluated by more than one proposal, providing some comparable results.

In general terms, the available results suggest that CP-based and PL-based automated support provide similar performance in general [13, 75]. PL-based solutions relying on BDDs (Binary Decision Diagrams) seems to be an exception providing execution times much faster than the ones provided by the rest of known approaches, specially when computing the number of solutions [13, 62, 75, 107]. The major drawback of this technique is the size of the BDD representing the feature model that can be exponential in the worst case. Several authors have worked in the development of new heuristics and techniques to reduce the size of the BDDs used on the analysis of feature models [62, 107]. Others focus on providing automated support using different paradigms in order to combine the best of all of them in terms of performance [13, 15].

A key aspect in the experimental work related to the analysis of feature models lies on the types of the subject problems used for the experiments. We found two main types of feature models used for experimentation: realistic and automatically generated feature models. We refer as *realistic* models to those modelling real–world domains or a simplified version of them. Some of the realistic feature models most quoted in the revised literature are: e-Shop [53] with 287 features, graph product line [55] with up to 64 features, BerkeleyDB [50] with 55 features and home integration system product line [10] with 15 features.

Although there are reports from industry of feature models with hundreds or even thousands of features [6, 54, 76], only a portion of them is typically published. This has led authors to generate feature models automatically to show the scalability of their approaches with large problems. These models are generated either randomly [13, 14, 60, 64, 75, 100, 101, 103, 104, 107] or trying to imitate the properties of the realistic models found in the literature [61, 79]. Several algorithms for the automated generation of feature models have been proposed [75, 79, 104].

In order to understand the relationship between realistic feature models and automatically generated models in experimentation, we counted the number of works using each

	Gheyi *et al.* [41]	Mendonca *et al.* [61]	Thüm *et al.* [79]	Zhang *et al.* [107]	Yan *et al.* [104]	Benavides *et al.* [9, 10, 11]	Benavides *et al.* [14]	White *et al.* [103]	White *et al.* [101]	Wang *et al.* [97]	Benavides *et al.* [13]	Segura [75]	Hemakumar [45]	Mendonca *et al.* [60]	Osman *et al.* [64]	White *et al.* [102, 100]
	PL					CP				DL	Multi		Others			
Void feature model		+		+	+		+				+	+				
#Products							+				+	+				
Dead features															+	
Valid product	+							+		+						
All products	+					+										
Explanations										+					+	
Refactoring	+		+													
Optimization																+
Atomic sets												+				
Corrective explanations								+								
Dependency analysis														+		
Generalization	+		+													
Arbitrary edit			+													
Conditional dead features													+			
Muti-step configuration									+							
Specialization			+													

Table 6: Summary of the studies reporting performance results for analysis operations

type by year. The results are shown in Figure 17. For each type of model, we also show the number of features of the largest feature model for each year. The figure shows an increasing trend in the number of empirical works since 2004 being specially notable in the last two years. First works used small realistic feature models in their experiments. However, since 2006, far more automatically generated feature models than realistic ones have been used. Regarding the size of the problems, there is a clear ascendant tendency ranging from the model with 15 features used in 2004 to the model with 20,000 features used in 2009. These findings reflect an increasing concern to evaluate and compare the performance of different solutions using larger and more complex feature models. This suggests that the analysis of feature models is maturing.

7. Discussions and challenges

In this section, we discuss the results obtained from the literature review. Based on these results, we identify a number of challenges (RQ3) to be addressed in the future. Challenges are part of the authors' own personal view of open questions, based on the analysis presented in this paper.

- *Formal definition of analysis operations.* As we have presented, most of the proposals define operations in terms of informal descriptions. For implementing a tool, it is desirable to have precise definition of the operations. Formal definitions of operations would facilitate both, communication among the community and tool development. Schobbens et al. [72, 73] and Benavides [7] have made some progress

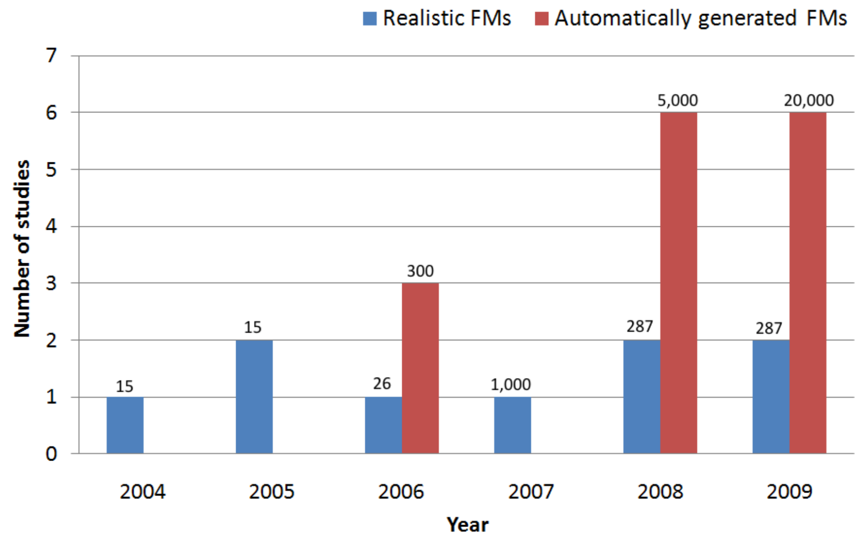


Figure 17: Type and maximum size of the feature models used in performance evaluations for each year

in this direction. Note that [7] was not included as a primary study because it has not published in a peer reviewed format.

Challenge 1: Formally describe all the operations of analysis and provide a formal framework for defining new operations.

- *Extended feature model analyses.* Analysis on basic or cardinality–based feature models are covered by most of the studies. However, extended feature models where numerical attributes are included, miss further coverage. When including attributes in feature models the analysis becomes more challenging because not only attribute–value pairs can be contemplated, also, more complex relationships can be included like *"feature Camera requires Scree.resolution ≥ 640x480"*. This type of relationships can affect operations of analysis and can include new ones. For instance, the number of products of a feature model can be reduced or augmented if these relationships are considered.

Challenge 2: Include feature attribute relationships for analyses on feature models and propose new operations of analysis leveraging extended feature models.

- *Performance and scalability of the operations.* Performance testing is being studied more and more and recent works show empirical evidences of the computational complexity of some analysis operations. We believe that a more rigorous analysis of computational complexity is needed. Furthermore, a set of standard benchmarks would be desirable to show how the theoretical computational complexity is run in practice.

Challenge 3: Further studies about computational complexity of analysis.

Challenge 4: Develop standard benchmarks for analysis operations.

- *Tools used for analysis.* As we have presented, there are mainly three groups of solvers used for analysis: constraint programming, description logic and propositional logic based solvers. From the primary studies, we detected that mainly proposals using constraint programming–based solvers are able to deal with extended feature models, i.e. feature models with attributes. Propositional logic–based solvers that use binary decisions diagrams as internal representations seem to be much more efficient for counting the number of products but present serious limitations regarding memory consumption. Description logic–based solvers have not been studied in depth to show their strengths and limitations when analysing feature models. Finally, it seems clear that not all solvers and paradigms will performance equally well for all the identified operations. A characterisation of feature models, operations and solvers seems to be an interesting topic to be explored in the future.

  Challenge 5: Study how propositional logic and description logic–based solvers can be used to add attributes on feature models.

  Challenge 6: Compare in depth description logic–based solvers with respect to analysis operations and other solvers.

  Challenge 7: Characterise feature models, analysis operations and solvers to select the best choice in each case.

8. Conclusions

The automated analysis of feature models is thriving. The extended use of feature models together with the many applications derived from their analysis has made this discipline to gain importance among researchers in software product lines. As a result, a number of analysis operations and approaches providing automated support for them are rapidly proliferating. In this paper, we revised the state of the art on the automated analysis of feature models by running a structured literature review covering 53 primary studies and outlining the main advances made up to now. As a main result, we presented a catalogue with 30 analysis operations identified in the literature and classified the existing proposal providing automated support for them according to their underlying logical paradigm. We also provided information about the tools used to perform the analyses and the results and trends related to the performance evaluation of the published proposals. From the analysis of current solutions, we conclude that the analysis of feature models is maturing with an increasing number of contributions, operations, tools and empirical works. We also identified a number of challenges for future research mainly related to the formalization and computational complexity of the operations, performance comparison of the approaches and the support of extended feature models.

Acknowledgments

We would like to thank Don Batory, Deepak Dhungana, Jose Galindo, Abdelrahman Osman Elfaki, Anna Queralt, Fabricia C. Roos, Ernets Teniente, Thomas Thum, Pablo

Trinidad and Pim Van den Broek for their helful commments in earlier versions of this article.

References

[1] L. Abo, F. Kleinermann, and O. De Troyer. Applying semantic web technology to feature modeling. In *SAC*, pages 1252–1256, 2009.

[2] F. Baader, D. Calvanese, D.L. McGuinness, D. Nardi, and P.F. Patel-Schneider, editors. *The description logic handbook: theory, implementation, and applications.* Cambridge University Press, New York, NY, USA, 2003.

[3] R. Bachmeyer and H. Delugach. A conceptual graph approach to feature modeling. In *ICCS*, pages 179–191, 2007.

[4] D. Batory. Feature models, grammars, and propositional formulas. In *Software Product Lines Conference*, volume 3714 of *Lecture Notes in Computer Sciences*, pages 7–20. Springer–Verlag, 2005.

[5] D. Batory. A tutorial on feature oriented programming and the ahead tool suite. In *Summer school on Generative and Transformation Techniques in Software Engineering*, 2005.

[6] D. Batory, D. Benavides, and A. Ruiz-Cortés. Automated analysis of feature models: Challenges ahead. *Communications of the ACM*, December:45–47, 2006.

[7] D. Benavides. *On the Automated Analyisis of Software Product Lines using Feature Models. A Framework for Developing Automated Tool Support.* PhD thesis, University of Seville, 2007.

[8] D. Benavides, D. Batory, P. Heymans, and A. Ruiz-Cortés, editors. *First Workshop on Analyses of Software Product Lines*, September 2008.

[9] D. Benavides, A. Ruiz-Cortés, and P. Trinidad. Coping with automatic reasoning on software product lines. In *Proceedings of the 2nd Groningen Workshop on Software Variability Management*, November 2004.

[10] D. Benavides, A. Ruiz-Cortés, and P. Trinidad. Automated reasoning on feature models. In *Advanced Information Systems Engineering: 17th International Conference, CAiSE 2005*, volume 3520 of *Lecture Notes in Computer Sciences*, pages 491–503. Springer–Verlag, 2005.

[11] D. Benavides, A. Ruiz-Cortés, and P. Trinidad. Using constraint programming to reason on feature models. In *The Seventeenth International Conference on Software Engineering and Knowledge Engineering, SEKE 2005*, pages 677–682, 2005.

[12] D. Benavides, S. Segura, and A. Ruiz-Cortés. Automated analysis of feature models: A detailed literature review. Technical Report ISA-09-TR-02, ISA research group, 2009. Available at http://www.isa.us.es/.

[13] D. Benavides, S. Segura, P. Trinidad, and A. Ruiz-Cortés. A first step towards a framework for the automated analysis of feature models. In *Managing Variability for Software Product Lines: Working With Variability Mechanisms*, 2006.

[14] D. Benavides, S. Segura, P. Trinidad, and A. Ruiz-Cortés. Using java csp solvers in the automated analyses of feature models. *Lecture Notes in Computer Science*, 4143:389–398, 2006.

[15] D. Benavides, S. Segura, P. Trinidad, and A. Ruiz-Cortés. FAMA: Tooling a framework for the automated analysis of feature models. In *Proceeding of the First International Workshop on Variability Modelling of Software-intensive Systems (VAMOS)*, pages 129–134, 2007.

[16] D. Le Berre. SAT4J solver, www.sat4j.org. published on line.

[17] B. Bollig and I. Wegener. Improving the variable ordering of obdds is np-complete. *IEEE Trans. Comput.*, 45(9):993–1002, 1996.

[18] P. Brereton, B. Kitchenham, D. Budgen, M. Turner, and M. Khalil. Lessons from applying the systematic literature review process within the software engineering domain. *Journal of Systems and Software*, 80(4):571–583, 2007.

[19] R. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, 35(8):677–691, 1986.

[20] F. Cao, B. Bryant, C. Burt, Z. Huang, R. Raje, A. Olson, and M. Auguston. Automating feature-oriented domain analysis. In *International Conference on Software Engineering Research and Practice (SERP'03)*, pages 944–949, June 2003.

[21] Citeseer. http://citeseer.ist.psu.edu/. published on line.

[22] Clark and Parsia. Pellet: the open source owl reasoner, http://clarkparsia.com/pellet/. published on line.

[23] A. Classen, P. Heymans, and P.Y. Schobbens. What's in a feature: A requirements engineering perspective. In *Fundamental Approaches to Software Engineering*, volume 4961, pages 16–30. Springer, 2008.

[24] P. Clements and L. Northrop. *Software Product Lines: Practices and Patterns*. SEI Series in Software Engineering. Addison–Wesley, August 2001.

[25] S. Cook. The complexity of theorem-proving procedures. In *Conference Record of Third Annual ACM Symposium on Theory of Computing*, pages 151–158, 1971.

[26] K. Czarnecki, T. Bednasch, P. Unger, and U. Eisenecker. Generative programming for embedded software: An industrial experience report. In *Generative Programming and Component Engineering, ACM SIGPLAN/SIGSOFT Conference, GPCE 2002*, pages 156–172, 2002.

[27] K. Czarnecki and U.W. Eisenecker. *Generative Programming: Methods, Techniques, and Applications*. Addison–Wesley, may 2000. ISBN 0–201–30977–7.

[28] K. Czarnecki, S. Helsen, and U. Eisenecker. Staged configuration through specialization and multilevel configuration of feature models. *Software Process: Improvement and Practice*, 10(2):143–169, 2005.

[29] K. Czarnecki, Simon Helsen, and Ulrich W. Eisenecker. Formalizing cardinality-based feature models and their specialization. *Software Process: Improvement and Practice*, 10(1):7–29, 2005.

[30] K. Czarnecki and P. Kim. Cardinality-based feature modeling and constraints: A progress report. In *Proceedings of the International Workshop on Software Factories At OOPSLA 2005*, 2005.

[31] K. Czarnecki, S. She, and A. Wasowski. Sample spaces and feature models: There and back again. In *proceedings of the Software Product Line Conference (SPLC)*, pages 22–31, 2008.

[32] M. Dean and G. Schreiber. OWL web ontology language reference. W3C recommendation, W3C, February 2004.

[33] CHOCO Developers. CHOCO solver, http://choco-solver.net/. published on line.

[34] GNU Prolog developers. GNU prolog, www.gprolog.org/. published on line.

[35] Various developers. Alloy analyzer, http://alloy.mit.edu/. published on line.

[36] Various developers. SMV system , www.cs.cmu.edu/∼modelcheck. published on line.

[37] JaCoP development team. JaCoP solver, http://jacop.osolpro.com/. published on line.

[38] O. Djebbi, C. Salinesi, and D. Diaz. Deriving product line requirements: the red-pl guidance approach. *Asia-Pacific Software Engineering Conference*, 0:494–501, 2007.

[39] S. Fan and N. Zhang. Feature model based on description logics. In *Knowledge-Based Intelligent Information and Engineering Systems*, 2006.

[40] D. Fernandez-Amoros, R. Heradio, and J. Cerrada. Inferring information from feature diagrams to product line economic models. In *Proceedings of the Sofware Product Line Conference*, 2009.

[41] R. Gheyi, T. Massoni, and P. Borba. A theory for feature models in alloy. In *Proceedings of the ACM SIGSOFY First Alloy Workshop*, pages 71–80, Portland, United States, nov 2006.

[42] R. Gheyi, T. Massoni, and P. Borba. Algebraic laws for feature models. *Journal of Universal Computer Science*, 14(21):3573–3591, 2008.

[43] Google Scholar. http://scholar.google.es. published on line.

[44] J. Greenfield, K. Short, S. Cook, and S. Kent. *Software Factories: Assembling Applications with Patterns, Models, Frameworks, and Tools*. Wiley, August 2004.

[45] Adithya Hemakumar. Finding contradictions in feature models. In *First International Workshop on Analyses of Software Product Lines (ASPL'08)*, pages 183–190, 2008.

[46] P. Heymans, P.Y. Schobbens, J.C. Trigaux, Y. Bontemps, R. Matulevicius, and A. Classen. Evaluating formal properties of feature diagram languages. *Software IET*, 2(3):281–302, 2008.

[47] ILOG. OPL studio, www.ilog.com/products/oplstudio/. published on line.

[48] K. Kang, S. Cohen, J. Hess, W. Novak, and S. Peterson. Feature–Oriented Domain Analysis (FODA) Feasibility Study. Technical Report CMU/SEI-90-TR-21, Software Engineering Institute, Carnegie Mellon University, November 1990.

[49] K.C. Kang, S. Kim, J. Lee, K. Kim, E. Shin, and M. Huh. FORM: A feature–oriented reuse method with domain–specific reference architectures. *Annals of Software Engineering*, 5:143–168, 1998.

[50] C. Kastner, S. Apel, and D. Batory. A case study implementing features using aspectj. In *SPLC '07: Proceedings of the 11th International Software Product Line Conference*, pages 223–232, Washington, DC, USA, 2007. IEEE Computer Society.

[51] B. Kitchenham. Procedures for performing systematic reviews. Technical report, Keele University and NICTA, 2004.

[52] P. Klint. A meta-environment for generating programming environments. *ACM Trans. Softw. Eng. Methodol.*, 2(2):176–201, April 1993.

[53] S.Q. Lau. Domain analysis of e-commerce systems using featurebased model templates. master's thesis. Dept. of ECE, University of Waterloo, Canada, 2006.

[54] F. Loesch and E. Ploedereder. Optimization of variability in software product lines. In *SPLC '07: Proceedings of the 11th International Software Product Line Conference*, pages 151–162, Washington, DC, USA, 2007. IEEE Computer Society.

[55] R.E Lopez-Herrejon and D. Batory. A standard problem for evaluating product-line methodologies. In *GCSE '01: Proceedings of the Third International Conference on Generative and Component-Based Software Engineering*, pages 10–24, London, UK, 2001. Springer-Verlag.

[56] M. Mannion. Using first-order logic for product line model validation. In *Proceedings of the Second Software Product Line Conference (SPLC'02)*, LNCS 2379, pages 176–187, San Diego, CA, 2002. Springer.

[57] M. Mannion and J. Camara. Theorem proving for product line model verification. In *Software Product-Family Engineering (PFE)*, volume 3014 of *Lecture Notes in Computer Science*, pages 211–224. Springer Berlin / Heidelberg, 2003.

[58] F. Marić. Formalization and implementation of modern sat solvers. *Journal of Automated Reasoning*, 43(1):81–119, June 2009.

[59] M. Mendonça, T.T. Bartolomei, and D. Cowan. Decision-making coordination in collaborative product configuration. In *Proceedings of the 2008 ACM symposium on Applied computing (SAC '08)*, pages 108–113, New York, NY, USA, 2008. ACM.

[60] M. Mendonça, D.D. Cowan, W. Malyk, and T. Oliveira. Collaborative product configuration: Formalization and efficient algorithms for dependency analysis. *Journal of Software*, 3(2):69–82, 2008.

[61] M. Mendonça, A. Wasowski, and K. Czarnecki. SAT–based analysis of feature models is easy. In *Proceedings of the Sofware Product Line Conference*, 2009.

[62] Marcílio Mendonça, Andrzej Wasowski, Krzysztof Czarnecki, and Donald D. Cowan. Efficient compilation techniques for large scale feature models. In *Generative Programming and Component Engineering, 7th International Conference, GPCE , Proceedings*, pages 13–22, 2008.

[63] A. Osman, S. Phon-Amnuaisuk, and C.K. Ho. Knowledge based method to validate feature models. In *First International Workshop on Analyses of Software Product Lines*, pages 217–225, 2008.

[64] A. Osman, S. Phon-Amnuaisuk, and C.K. Ho. Using first order logic to validate feature model. In *Third International Workshop on Variability Modelling in Software-intensive Systems (VaMoS)*, pages 169–172, 2009.

[65] J. Pena, M. Hinchey, A. Ruiz-Cortés, and P. Trinidad. Building the core architecture of a multiagent system product line: With an example from a future nasa mission. In *7th International Workshop on Agent Oriented Software Engineering*, Lecture Notes in Computer Sciences. Springer–Verlag, 2006.

[66] K. Pohl, G. Böckle, , and F. van der Linden. *Software Product Line Engineering: Foundations, Principles, and Techniques*. Springer–Verlag, 2005.

[67] Racer Systems GmbH Co. KG. RACER, www.racer-systems.com. published on line.

[68] R Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1):57–95, 1987.

[69] M. Riebisch, K. Böllert, D. Streitferdt, and I. Philippow. Extending feature diagrams with UML multiplicities. In *6th World Conference on Integrated Design & Process Technology (IDPT2002)*, June 2002.

[70] C. Salinesi, C. Rolland, and R. Mazo. Vmware: Tool support for automatic verification of structural and semantic correctness in product line models. In *Third International Workshop on Variability Modelling of Software-intensive Systems*, pages 173–176, 2009.

[71] K. Schmid and I. John. A customizable approach to full lifecycle variability management. *Sci. Comput. Program.*, 53(3):259–284, 2004.

[72] P. Schobbens, P. Heymans, J. Trigaux, and Y. Bontemps. Feature Diagrams: A Survey and A Formal Semantics. In *Proceedings of the 14th IEEE International Requirements Engineering Conference (RE'06)*, Minneapolis, Minnesota, USA, September 2006.

[73] P. Schobbens, J.C. Trigaux P. Heymans, and Y. Bontemps. Generic semantics of feature diagrams. *Computer Networks*, 51(2):456–479, Feb 2007.

[74] Scopus. www.scopus.com. published on line.

[75] S. Segura. Automated analysis of feature models using atomic sets. In *First Workshop on Analyses of Software Product Lines (ASPL 2008). SPLC'08*, pages 201–207, Limerick, Ireland, September 2008.

[76] M. Steger, C. Tischer, B. Boss, A. Müller, O. Pertler, W. Stolz, and S. Ferber. Introducing pla at bosch gasoline systems: Experiences and practices. In *SPLC*, pages 34–50, 2004.

[77] D. Streitferdt, M. Riebisch, and I. Philippow. Details of formalized relations in feature models using OCL. In *Proceedings of 10th IEEE International Conference on Engineering of Computer–Based Systems (ECBS 2003), Huntsville, USA. IEEE Computer Society*, pages 45–54, 2003.

[78] J. Sun, H. Zhang, Y.F. Li, and H. Wang. Formal semantics and verification for feature modeling. In *Proceedings of the 10th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS)*, 2005.

[79] T. Thüm, D. Batory, and C. Kästner. Reasoning about edits to feature models. In *International Conference on Software Engineering*, pages 254–264, 2009.

[80] P. Trinidad, D. Benavides, A. Durán, A. Ruiz-Cortés, and M. Toro. Automated error analysis for the agilization of feature modeling. *Journal of Systems and Software*, 81(6):883–896, 2008.

[81] P. Trinidad, D. Benavides, and A. Ruiz-Cortés. Improving decision making in software product lines product plan management. In J. Dolado, I. Ramos, and J. Cuadrado-Gallego, editors, *Proceedings of the V ADIS 2004 Workshop on Decision Support in Software Engineering*, volume 120. CEUR Workshop Proceedings (CEUR-WS.org), 2004.

[82] P. Trinidad, D. Benavides, and A. Ruiz-Cortés. A first step detecting inconsistencies in feature models. In *CAiSE Short Paper Proceedings*, 2006.

[83] P. Trinidad, D. Benavides, A. Ruiz-Cortés, S. Segura, and A.Jimenez. Fama framework. In *12th Software Product Lines Conference (SPLC)*, 2008.

[84] Pablo Trinidad and Antonio Ruiz Cortés. Abductive reasoning and automated analysis of feature models: How are they connected? In *Third International Workshop on Variability Modelling of Software-Intensive Systems. Proceedings*, pages 145–153, 2009.

[85] S. Trujillo, D. Batory, and Oscar Díaz. Feature oriented model driven development: A case study for portlets. In *International Conference on Software Engineering*, pages 44–53, 2007.

[86] E. Tsang. *Foundations of Constraint Satisfaction*. Academic Press, 1995.

[87] M. Tseng and J. Jiao. *Handbook of Industrial Engineering: Technology and Operations Management*, chapter Mass Customization, page 685. Wiley, 2001.

[88] P. van den Broek and I. Galvao. Analysis of feature models using generalised feature trees. In *Third International Workshop on Variability Modelling of Software-intensive Systems*, number 29 in ICB-Research Report, pages 29–35, Essen, Germany, January 2009. Universität Duisburg-Essen.

[89] F. van der Linden, K. Schmid, and E. Rommes. *Software Product Lines in Action: The Best Industrial Practice in Product Line Engineering*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007.

[90] T. van der Storm. Variability and component composition. In *Software Reuse: Methods, Techniques and Tools: 8th International Conference, ICSR 2004. Proceedings*, volume 3107 of *Lecutre Notes in Computer Sciences*, pages 157–166. Springer, July 2004.

[91] Tijs van der Storm. Generic feature-based software composition. In *Software Composition*, volume 4829 of *LNCS*, pages 66–80. Springer, 2007.

[92] A. van Deursen and P. Klint. Domain–specific language design requires feature descriptions. *Journal of Computing and Information Technology*, 10(1):1–17, 2002.

[93] T. von der Massen and H. Lichter. Requiline: A requirements engineering tool for software product lines. In F. van der Linden, editor, *Proceedings of the Fifth International Workshop on Product Family Engineering (PFE)*, LNCS 3014, Siena, Italy, 2003. Springer Verlag.

[94] T. von der Massen and H. Lichter. Deficiencies in feature models. In Tomi Mannisto and Jan Bosch, editors, *Workshop on Software Variability Management for Product Derivation - Towards Tool Support*, 2004.

[95] T. von der Massen and H. Litcher. Determining the variation degree of feature models. In *Software Product Lines Conference, LNCS 3714*, pages 82–88, 2005.

[96] H. Wang, Y. Li, J. Sun, H. Zhang, and J. Pan. A semantic web approach to feature modeling and verification. In *Workshop on Semantic Web Enabled Software Engineering (SWESE'05)*, November 2005.

[97] H. Wang, Y.F. Li, J. un, H. Zhang, and J. Pan. Verifying Feature Models using OWL. *Journal of Web Semantics*, 5:117–129, June 2007.

[98] J. Webster and R. Watson. Analyzing the past to prepare for the future: Writing a literature review. *MIS Quarterly*, 26(2):xiii–xxiii, 2002.

[99] J. Whaley. JavaBDD, http://javabdd.sourceforge.net/. published on line.

[100] J. White, B. Doughtery, and D. Schmidt. Selecting highly optimal architectural feature sets with filtered cartesian flattening. *Journal of Systems and Software*, 82(8):1268–1284, 2009.

[101] J. White, B. Doughtery, D. Schmidt, and D. Benavides. Automated reasoning for multi-step software product-line configuration problems. In *Proceedings of the Sofware Product Line Conference*, pages 11–20, 2009.

[102] J. White and D. Schmidt. Filtered cartesian flattening: An approximation technique for optimally selecting features while adhering to resource constraints. In *First International Workshop on Analyses of Software Product Lines (ASPL)*, pages 209–216, 2008.

[103] J. White, D. Schmidt, D. Benavides P. Trinidad, and A. Ruiz-Cortes. Automated diagnosis of product-line configuration errors in feature models. In *Proceedings of the Sofware Product Line Conference*, 2008.

[104] H. Yan, W. Zhang, H. Zhao, and H. Mei. An optimization strategy to feature models' verification by eliminating verification-irrelevant features and constraints. In *ICSR*, pages 65–75, 2009.

[105] W. Zhang, H. Mei, and H. Zhao. Feature-driven requirement dependency analysis and high-level software design. *Requirements Engineering*, 11(3):205–220, June 2006.

[106] W. Zhang, H. Zhao, and H. Mei. A propositional logic-based method for verification of feature models. In J. Davies, editor, *ICFEM 2004*, volume 3308, pages 115–130. Springer–Verlag, 2004.

[107] Wei Zhang, Hua Yan, Haiyan Zhao, and Zhi Jin. A bdd–based approach to verifying clone-enabled feature models' constraints and customization. In *High Confidence Software Reuse in Large Systems, 10th International Conference on Software Reuse, ICSR, Proceedings*, LNCS, pages 186–199. Springer, 2008.

Appendix A

Summary of papers analysed

A.1 1990

Paper title:	Feature–Oriented Domain Analysis (FODA) Feasibility Study		
Authors:	K. Kang and S. Cohen and J. Hess and W. Novak and S. Peterson		
Publication:	Technical Report	Year:	1990
Acronym:	kang90-tr	Pages:	148
DOI/URL:	Personal web page		

Summary

Kang et *al.* proposed feature models for the first time in [31]. In addition to a whole method for domain analysis, authors propose feature models as a key modelling technique to capture variabilities and commonalities. In the original report, explicit mention to automated analysis of feature models was already made [31, pag. 70]. Indeed, void feature model, valid product, dead features detection and explanations were already proposed as operation of analysis. A prolog–based prototype is also reported. However, not detailed information is provided to replicate their prolog coding.

Analyses

Paradigm:	Prolog		
FM notation:	Basic FMs	Extended FM:	Yes
Formalization:	No		

Operations

Operation	Alternative name	Support
valid product		Yes
void FM		Yes
valid partial configuration		Yes
dependency analysis		Yes
dead features	feature reachability	Yes
explanations		Yes

Internal statistics

Self-citations:

FaMa ext.:

Ideas:

Table A.1: Kang et al. 1990 FODA

A.2 2002

Paper title:	Domain–Specific Language Design Requires Feature Descriptions		
Authors:	A. van Deursen and P. Klint		
Publication:	JCIT	Year:	2002
Acronym:	deursen02-jcit	Pages:	20
DOI/URL:	Eprint version		

Summary

To the best of our knowledge, Deursen *et al.* [61] were the first authors proposing some kind of automated support for the automated treatment of feature model after their introduction in the FODA report [31] back in 1990. In their work, they propose a textual feature diagram algebra together with a prototype implementation using the ASF+SDF Meta-Environment [32]. In particular, they provide support for the automated extraction of the number and list of products of a feature model and what they call constraint satisfaction (i.e. checking if the model is not void).

Analyses

Paradigm:	Ad–hoc		
FM notation:	Basic FMs	Extended FM:	No
Formalization:	No		

Operations

Operation	Alternative name	Support
void FM	constraint satisfaction	Yes
all products		Yes
number of products	variability	Yes

Internal statistics

Self-citations:

FaMa ext.:

Ideas:

Table A.2: Deursen et al. 2002 JCIT

Paper title:	Using First-Order Logic for Product Line Model Validation		
Authors:	M. Mannion		
Publication:	SPLC	Year:	2002
Acronym:	mannion02-splc	Pages:	12
DOI/URL:	10.1007/3-540-45652-X		

Summary

Mannion [33] was the first who connected propositional formulas to feature models. In his work, feature models are used as requirements models for software product lines. Rules for translating such models into propositional formulas are provided and some operations are identified on the automated analysis of feature models.

Analyses

Paradigm:	Propositional Logic		
FM notation:	Basic FMs	Extended FM:	No
Formalization:	No		

Operations

Operation	Alternative name	Support
valid product	valid single system	Yes
void FM	PL model validation	Yes
all products	all possible products	Yes
number of products		Yes

Internal statistics

Self-citations:

FaMa ext.:

Ideas:

Table A.3: Mannion 2002 SPLC

A.3 2003

Paper title:	Automating Feature-Oriented Domain Analysis		
Authors:	F. Cao, B. Bryant, C. Burt, Z. Huang, R. Raje, A. Olson and M. Auguston		
Publication:	SERP	Year:	2003
Acronym:	cao03-serp	Pages:	6
DOI/URL:	Personal web page		

Summary

Cao *et al.* [12] present ad-hoc algorithms for extracting the list of products of a basic feature model in the context of generative programming [13]. In their work, they deal with the simplification (also called normalization) of feature models in order to make them easier to be processed. They also present a tool prototype based on their algorithm.

Analyses

Paradigm:	Ad–hoc		
FM notation:	Basic FMs	Extended FM:	No
Formalization:	No		

Operations

Operation	Alternative name	Support
void FM	constraint checking	Yes
all products	list of feature instances	Yes

Internal statistics

Self-citations:

FaMa ext.:

Ideas:

Table A.4: Cao et al. 2003 SERP

Paper title:	Theorem Proving for Product Line Model Verification		
Authors:	M. Mannion and J. Camara		
Publication:	PFE (SPLC-Europe)	Year:	2003
Acronym:	mannion03-pfe	Pages:	14
DOI/URL:	10.1007/b97155		

Summary

In [34], Mannion and J. Camara extend the work presented in [33]. The authors propose an ad-hoc algorithm to deal with the propositional formulas representing the feature models and gave some details about the computational complexity of their approach.

Analyses

Paradigm:	Propositional Logic		
FM notation:	Basic FMs	Extended FM:	No
Formalization:	No		

Operations

Operation	Alternative name	Support
valid product	valid single system	Yes
void FM	PL model validation	Yes
all products		Yes
number of products		Yes

Internal statistics

Self-citations:

FaMa ext.:

Ideas:

Table A.5: Mannion et al. 2003 PFE

Paper title:	RequiLine: A Requirements Engineering Tool for Software Product Lines		
Authors:	T. Massen and H. Lichter		
Publication:	PFE	Year:	2003
Acronym:	massen03-pfe	Pages:	13
DOI/URL:	10.1007/b97155		

Summary

Von der Massen *et al.* [62] present Requiline, a requirement engineering tool for software product lines. Feature modeling is supported by means of a custom feature model meta-model inspired by FODA. The tool is mainly implemented using a relational data base and ad–hoc algorithms. Some analyses can be performed by using a consistency checker integrated in the tool. Custom SQL queries are also allowed to extract information from the requirement and featurem models stored in the repository.

Analyses

Paradigm:	Ad–hoc		
FM notation:	Basic FMs	Extended FM:	No
Formalization:	No		

Operations

Operation	Alternative name	Support
valid product	configuration consistency	Yes
void FM	model consistency	Yes

Internal statistics

Self-citations:

FaMa ext.:

Ideas:

Table A.6: Massen et al. 2003 PFE

A.4 2004

Paper title:	Coping with Automatic Reasoning on Software Product Lines		
Authors:	D. Benavides, A. Ruiz-Cortés and P. Trinidad		
Publication:	SVM	Year:	2004
Acronym:	benavides04-svm	Pages:	13
DOI/URL:	–		

Summary

This is a preliminary work later published in [6]. In this work, authors provide with a set of mapping rules to translate feature models into a CSP. They also provide an algortihm to translate extended feature models to CSP.

Analyses

Paradigm:	Constraint Programmming		
FM notation:	Basic FMs	Extended FM:	Yes
Formalization:	Yes		

Operations

Operation	Alternative name	Support
void FM	validation	Yes
all products		Yes
number of products		Yes
filter		Yes
optimization		Yes

Empirical evaluation

Number of instances:		Available:	No
Type of problems:	Published and real	Format:	XML
Environment description:	No		

Internal statistics

Self-citations:

FaMa ext.:

Ideas:

Table A.7: Benavides et al. 2004 SVM

Paper title:	Deficiencies in Feature Models		
Authors:	T. Massen and H. Lichter		
Publication:	SVMPD (SPLC workshop)	Year:	2004
Acronym:	massen04-svmpd	Pages:	14
DOI/URL:	Workshop web site		

Summary

Von der Massen *et al.* [63] were the first categorizing the types of problems that might arise in feature models. They classify these into three types: redundancies, anomalies and inconsistencies. Both, anomalies and inconsistencies are specific cases of dead features and false optionals. Some hints about how fixing these problems (i.e. corrective explanations) are also provided. They present Requiline [62] as a feature modeling prototype tool able to detect inconsistencies.

Analyses

Paradigm:			
FM notation:	Basic FMs	Extended FM:	No
Formalization:	No		

Operations

Operation	Alternative name	Support
false optional	anomalies	No
dead features	inconsistencies	No
redundancies		No
c. explanations		No

Internal statistics

Self-citations:	
FaMa ext.:	Yes
Ideas:	Implement redundancies detection in FaMa

Table A.8: Massen et al. 2004 SVMPD

Paper title:	Variability and Component Composition		
Authors:	T. van der Storm		
Publication:	ICSR	Year:	2004
Acronym:	storm04-icsr	Pages:	10
DOI/URL:	10.1007/b98465		

Summary

Van der Storm [59] present a method to deal with variability in component composition. In terms of analysis of feature models, they provide a method to verify if a feature model is void and check for valid partial assignments. They represent FMs using FDL (*Feature Description Language*) and then provide a mapping to translate a FDL to BDD.

Analyses

Paradigm:	Propositional Logic		
FM notation:	Basic FMs	Extended FM:	No
Formalization:	No		

Operations

Operation	Alternative name	Support
void FM	internal consistency	Yes
valid partial configuration		Yes

Internal statistics

Self-citations:

FaMa ext.:

Ideas:

Table A.9: Storm 2004 ICSR

Paper title:	A Propositional Logic-Based Method for Verification of Feature Models		
Authors:	W. Zhang, H. Zhao and H. Mei		
Publication:	ICFEM	Year:	2004
Acronym:	zhang04-icfem	Pages:	16
DOI/URL:	10.1007/b102837		

Summary

Zhang *et al.* [75] propose automating the analysis of feature models by means of the SMV [20] System, a tool supporting the analysis of propositional formulas. One of the main contributions of their work is the simplification of feature models by grouping feature into so-called *atomic sets*, later explored by Segura [51]. Moreover, a systematic way to detect dead features is provided as well.

Analyses

Paradigm:	Propositional Logic		
FM notation:	Basic FMs	Extended FM:	No
Formalization:	Yes		

Operations

Operation	Alternative name	Support
void FM	satisfiability	Yes
dead features	usability	Yes
false optional	chance to be removed	Yes
atomic sets		Yes

Internal statistics

Self-citations:

FaMa ext.:

Ideas:

Table A.10: Zhang et al. 2004 ICFEM

A.5 2005

Paper title:	Automated Reasoning on Feature Models		
Authors:	D. Benavides, P. Trinidad and A. Ruiz-Cortés		
Publication:	CAiSE	Year:	2005
Acronym:	benavides05-caise	Pages:	14
DOI/URL:	10.1007/11431855_34		

Summary

Authros propose the usage of constraint programming for the automated analysis of feature models [6]. In this work, authors provide with a set of mapping rules to translate feature models into a CSP. In contrast to the rest of identified approaches, this is the only proposal providing support for the analysis of extended feature models (i.e. including feature attributes) and the operation of optimization.

Analyses

Paradigm:	Constraint Programmming		
FM notation:	Basic FMs	Extended FM:	Yes
Formalization:	Yes		

Operations

Operation	Alternative name	Support
void FM		Yes
all products		Yes
number of products		Yes
variability		Yes
commonality		Yes
filter		Yes
optimization		Yes

Empirical evaluation

Number of instances:		Available:	No
Type of problems:	Published and real	Format:	
Environment description:	Yes		

Internal statistics

Self-citations: benavides04-svm

FaMa ext.:

Ideas:

Table A.11: Benavides et al. 2005 CAiSE

Paper title:	Using Constraint Programming to Reason on Feature Models		
Authors:	D. Benavides, P. Trinidad and A. Ruiz-Cortés		
Publication:	SEKE	Year:	2005
Acronym:	benavides05-seke	Pages:	6
DOI/URL:	Personal web site		

Summary

In [7], the authors present a short extension of the paper published in [6].

Analyses

Paradigm:	Constraint Programmming		
FM notation:	Basic FMs	Extended FM:	Yes
Formalization:	Yes		

Operations

Operation	Alternative name	Support
void FM	validation	Yes
all products		Yes
number of products		Yes
variability		Yes
commonality		Yes
filter		Yes
optimization		Yes

Empirical evaluation

Number of instances:	5	Available:	No
Type of problems:	Published and real	Format:	
Environment description:	Yes		

Internal statistics

Self-citations:	benavides04-svm, benavides05-caise
FaMa ext.:	
Ideas:	

Table A.12: Benavides et al. 2005 SEKE

Paper title:	Feature Models, Grammars, and Propositional Formulas		
Authors:	D. Batory		
Publication:	SPLC	Year:	2005
Acronym:	batory05-splc	Pages:	14
DOI/URL:	10.1007/11554844_3		

Summary

In [4], Batory shows the connections between feature models, grammars and propositional formulas. Batory argue that feature models can be represented as context-free grammars plus propositional formulas what can be the base for the construction of feature model compilers and domain specific languages. A set of rules for translating grammars representing feature models into propositional formulas is provided. Furthermore, a *Logic Truth Maintenance System* (a system that maintains the consequences of a propositional formula) is presented for the automated analysis of feature models. This system is constructed using a SAT solver and known boolean constraint propagation algorithms.

Analyses

Paradigm:	Propositional Logic		
FM notation:	Basic FMs	Extended FM:	No
Formalization:	No		

Operations

Operation	Alternative name	Support
valid product	product specification completeness	Yes
valid partial configuration		Yes
explanations	FM debugging	Yes
void FM	satisfiable	Yes
all products		Yes

Internal statistics

Self-citations:	benavides05-caise
FaMa ext.:	Yes, use LMTS for explanations
Ideas:	

Table A.13: Batory 2005 SPLC

Paper title:	Cardinality-Based Feature Modeling and Constraints: A Progress Report		
Authors:	K. Czarnecki and P. Kim		
Publication:	WSF (OOPSLA workshop)	Year:	2005
Acronym:	czarnecki05-oopsla	Pages:	9
DOI/URL:	Workshop Web site		

Summary

Czarnecki *et al.* [16] propose using Object-Constraint Language (OCL) to capture constraints in cardinality-based feature models. The authors overview some of the analyses that can be performed on feature models and present a tool prototype implementing some of these using BDD. As a novel contribution, the authors suggest using the degree of orthogonality as a way to measure how local decision in a subtree can influence choices in other parts of the model.

Analyses

Paradigm:	Propositional Logic(BDD)		
FM notation:	Cardinality-based FMs	Extended FM:	No
Formalization:	No		

Operations

Operation	Alternative name	Support
valid product	valid configuration	Yes
valid partial configuration		Yes
void FM		Yes
number of products		Yes
filter		Yes
dead features		
explanations		
degree of orthogonality		

Internal statistics

Self-citations:	benavides05-caise
FaMa ext.:	Yes
Ideas:	Degree of orthogonality

Table A.14: Czarnecki et al. 2005 OOPSLA

Paper title:	Determining the Variation Degree of Feature Models		
Authors:	T. Massen and H. Lichter		
Publication:	SPLC	Year:	2005
Acronym:	massen05-splc	Pages:	7
DOI/URL:	10.1007/11554844_9		

Summary

Von der Massen *et al.* [64] propose a method to calculate a rough approximation of the number of products of a feature model, what they call *variation degree*. The technique is described using mathematical expressions. Not explicit automated support is provided.

Analyses

Paradigm:	Ad–hoc		
FM notation:	Basic FMs	Extended FM:	No
Formalization:	Yes		

Operations

Operation	Alternative name	Support
number of products	variation degree	Yes

Internal statistics

Self-citations:	
FaMa ext.:	Yes
Ideas:	Include this algorithm in FaMa

Table A.15: Massen et al. 2005 SPLC

Paper title:	Formal Semantics and Verification for Feature Modeling		
Authors:	J. Sun, H. Zhang, Y.F. Li and H. Wang		
Publication:	ICECCS	Year:	2005
Acronym:	sun05-iceccs.tex	Pages:	10
DOI/URL:	10.1109/ICECCS.2005.48		

Summary

Sun *et al.* [52] propose a formalization of feature models using Z [71]. They also propose enabling the automated analysis of feature model by encoding them in Alloy and using the Alloy Analyzer[1]. Alloy is a structural modelling language based on first-order logic. Alloy Analyzer is a tool reasoning over alloy models that internally uses a SAT solver to check model satisfiability. Specially relevant in this approach is the identification and treatment of explanations when a feature model is void, i.e. it represents no products.

Analyses

Paradigm:	Propositional Logic (First Order Logic)		
FM notation:	Basic FMs	Extended FM:	No
Formalization:	Yes		

Operations

Operation	Alternative name	Support
valid product	valid configuration	Yes
void FM	FM solvability/consistency	Yes
all products	all configurations	Yes
refactoring		Yes
explanations	causes of inconsistency	Yes

Internal statistics

Self-citations:

FaMa ext.:

Ideas:

Table A.16: Sun et al. 2005 ICECCS

[1] http://alloy.mit.edu/

Paper title:	A Semantic Web Approach to Feature Modeling and Verification		
Authors:	H. Wang, Y.F. Li, J. Sun, H. Zhang and J. Pan		
Publication:	SWESE	Year:	2005
Acronym:	wang05-swese	Pages:	15
DOI/URL:	Workshop Web site		

Summary

Wang *et al.* [65] were the first to enable the automated analysis of feature models using DL. In their work, the authors introduce a set of mapping rules to translate feature models into OWL-DL ontologies [19]. OWL-DL is an expressive yet decidable sub language of OWL [19]. Then, the authors suggest using DL reasoning engines such as RACER[2] to perform automated analysis over the OWL representations of the models.

Analyses

Paradigm:	Description Logic		
FM notation:	Basic FMs	Extended FM:	No
Formalization:	No		

Operations

Operation	Alternative name	Support
valid product	valid configuration	Yes
explanations		Yes

Internal statistics

Self-citations:

FaMa ext.:

Ideas:

Table A.17: Wang et al. 2005 SWESE

[2]http://www.racer-systems.com/

A.6 2006

Paper title:	Automated Analyses of Feature Models: Challenges Ahead		
Authors:	D. Batory, D. Benavides and A. Ruiz-Cortés		
Publication:	CACM	Year:	2006
Acronym:	batory06-cacm	Pages:	3
DOI/URL:	10.1145/1183264		

Summary

In [5], Batory *et al.* propose a set of challenges ahead on the automated analysis of feature models. One of those challenges is feature model debugging and explanations.

Analyses

Paradigm:			
FM notation:	Basic FMs	Extended FM:	Yes
Formalization:	No		

Operations

Operation	Alternative name	Support
valid product	feature compatibility	No
optimization		No
dead features		No
explanations	debugging	No

Internal statistics

Self-citations:	benavides05-gttse, benavides05-caise
FaMa ext.:	
Ideas:	

Table A.18: Batory et al. 2006 CACM

Paper title:	Using Java CSP Solvers in the Automated Analyses of Feature Models		
Authors:	D. Benavides, S. Segura, P. Trinidad and A. Ruiz-Cortés		
Publication:	GTTSE	Year:	2006
Acronym:	benavides05-gttse	Pages:	10
DOI/URL:	10.1007/11877028_16		

Summary

Benavides *et al.* [10] present a performance comparison between two CSP solvers for the automated analysis of feature models. They also provide as a novel contribution a mapping from cardinality–based feature models to CSP.

Analyses

Paradigm:	Constraint Programmming		
FM notation:	Cardinality-based FMs	Extended FM:	No
Formalization:	No		

Operations

Operation	Alternative name	Support
void FM		Yes
number of products		Yes

Empirical evaluation

Number of instances:	5	Available:	No
Type of problems:	Published and random	Format:	
Environment description:	Yes		

Internal statistics

Self-citations:	benavides05-caise, benavides05-seke, benavides05-ewmt
FaMa ext.:	
Ideas:	

Table A.19: Benavides et al. 2006 GTTSE

Paper title:	Feature Model Based on Description Logics		
Authors:	S. Fan and N. Zhang		
Publication:	KES	Year:	2006
Acronym:	fan06-kes	Pages:	8
DOI/URL:	10.1007/11893004_145		

Summary

Fan *et al.* [22] also propose translating feature models into DL and using reasoners as RACER to perform their anlayses. According to the authors, their proposal address cardinality-based feature models. However, a clear example of how this can be done is missed.

Analyses

Paradigm:	Description Logic		
FM notation:	Basic FMs	Extended FM:	No
Formalization:	Yes		

Operations

Operation	Alternative name	Support
void FM	FM consistency	Yes

Internal statistics

Self-citations:	benavides05-caise,benavides04-svm
FaMa ext.:	Introduce DL reasoners in FaMa
Ideas:	

Table A.20: Fan et al. 2006 KES

Paper title:	A Theory for Feature Models in Alloy		
Authors:	R. Gheyi, T. Massoni and P. Borba		
Publication:	First Alloy Workshop	Year:	2006
Acronym:	gheyi06-alloy	Pages:	10
DOI/URL:	Workshop Web site		

Summary

Gheyi *et al.* [24] propose using Alloy and the Alloy Analyzer to automate the analysis of feature models. As a novel contribution, the authors detail how their proposal may be used to check the correctness of feature model refactoring rules as described in one of their papers[2].

Analyses

Paradigm:	Propositional Logic (Alloy)		
FM notation:	Basic FMs	Extended FM:	No
Formalization:	Yes		

Operations

Operation	Alternative name	Support
valid product		Yes
all products		Yes
refactoring		Yes
generalization		Yes

Empirical evaluation

Number of instances:	11	Available:	Yes
Type of problems:	Random	Format:	
Environment description:	Yes		

Internal statistics

Self-citations:	benavides05-caise
FaMa ext.:	Yes, include Alloy analyzer
Ideas:	Use Alloy to validate the merging of FMs

Table A.21: Gheyi et al. 2006 Alloy

Paper title:	A first step towards a framework for the automated analysis of feature models		
Authors:	D. Benavides, S. Segura, P. Trinidad and A. Ruiz-Cortés		
Publication:	SPLC WS	Year:	2006
Acronym:	benavides06-splc	Pages:	5
DOI/URL:	Techincal report web site		

Summary

Benavides *et al.* [9] present a performance comparison between three of the most used solvers for the automated analysis of feature models, namely: CSP, SAT and BDD. They provide some experimental evidences that,in general, BDD perform faster than CSP and SAT, however, BDD require more memory than CSP and SAT. They run the experiments using two operations of analysis: void feature model and number of products.

Analyses

Paradigm:	Propositional Logic and Constraint Programmming		
FM notation:	Basic FMs	Extended FM:	No
Formalization:	No		

Operations

Operation	Alternative name	Support
void FM		Yes
number of products		Yes

Empirical evaluation

Number of instances:	200	Available:	No
Type of problems:	Random	Format:	
Environment description:	Yes		

Internal statistics

Self-citations:	benavides05-caise, batory06-cacm, benavides05-seke, benavides05-gttse
FaMa ext.:	
Ideas:	

Table A.22: Benavides et al. 2006 SPLC

Paper title:	Feature Diagrams: A Survey and a Formal Semantics		
Authors:	P. Schobbens, P. Heymans and J. Trigaux		
Publication:	RE	Year:	2006
Acronym:	schobbens06-re	Pages:	10
DOI/URL:	10.1109/RE.2006.23		

Summary

In [49] (which is a previous work of [50]), Schobbens *et al.* survey the state-of-the-art of feature model notations and compare the expressiveness and succinctness of the different proposals. They do not provide any explicit automated support for the automated analyses of feature models. However, they formally describe what they call *decision procedures*.

Analyses

Paradigm:			
FM notation:	VFD	Extended FM:	No
Formalization:	Yes		

Operations

Operation	Alternative name	Support
valid product	product checking	No
void FM	checking satisfiabiliy	No
refactoring	equivalence	No

Internal statistics

Self-citations:	benavides05-caise
FaMa ext.:	
Ideas:	

Table A.23: Schobbens et al. 2006 RE

Paper title:	Isolated Features Detection in Feature Models		
Authors:	P. Trinidad, D. Benavides and A. Ruiz-Cortés		
Publication:	CAiSE Forum	Year:	2006
Acronym:	trinidad06-caise	Pages:	4
DOI/URL:	Workshop proceedings		

Summary

In [56], Trinidad *et al.* propose the detection of what they call insolated features (a.k.a dead features). The propose to use Constraint Programmming to detect those features using commonality factor. They also propose a possible optimization using variation degree as described by [64].

Analyses

Paradigm:	Constraint Programmming		
FM notation:	Basic FMs	Extended FM:	No
Formalization:	Yes		

Operations

Operation	Alternative name	Support
commonality		Yes
dead features	insolated feature detection	Yes

Internal statistics

Self-citations:	benavides05-caise
FaMa ext.:	Introduce the optimization using variation degree
Ideas:	

Table A.24: Trinidad et al. 2006 CAiSE

Paper title:	Feature-driven requirement dependency analysis and high-level software design		
Authors:	W. Zhang, H. Mei and H. Zhao		
Publication:	RE	Year:	2006
Acronym:	zhang06-re	Pages:	16
DOI/URL:	10.1007/s00766-006-0033-x		

Summary

In [74], *Zhang et al.* introduce four kind of dependencies between features and show how these can be analysed and used to design high-level software architecture. As a part of their work, they show how their previous approach [75] can be used to detect 17 kinds of anomalies and inconsistencies presented by von der Massen and Lichter [63].

Analyses

Paradigm:	Propositional Logic		
FM notation:	Basic FMs	Extended FM:	No
Formalization:	No		

Operations

Operation	Alternative name	Support
void FM	consistency	Yes
dead features	chance to be bound	Yes
false optional	chance to be removed	Yes

Internal statistics

Self-citations:	benavides05-caise
FaMa ext.:	
Ideas:	

Table A.25: Zhang et al. 2006 RE

A.7 2007

Paper title:	Deriving Product Line Requirements: the RED-PL Guidance Approach		
Authors:	O. Djebbi, C. Salinesi and D. Diaz		
Publication:	APSEC	Year:	2007
Acronym:	djebbi07-apsec	Pages:	8
DOI/URL:	10.1109/ASPEC.2007.63		

Summary

Djebbi *et al.* [21] propose a method to support product requirement derivation. As a part of their approach, they suggest extracting information from feature model in terms of queries. A set of rules to translate feature models to boolean constraints are given. Additioinally, they also describe a tool under development enabling the analysis of feture models using constraint programming.

Analyses

Paradigm:	Constraint Programmming		
FM notation:	Cardinality-based FMs	Extended FM:	Yes
Formalization:	No		

Operations

Operation	Alternative name	Support
void FM		Yes
number of products		Yes
filter		Yes
optimization		Yes

Internal statistics

Self-citations:	benavides06-splc
FaMa ext.:	
Ideas:	

Table A.26: Djebbi et al. 2007 APSEC

Paper title:	A Conceptual Graph Approach to Feature Modeling		
Authors:	R. Bachmeyer and H. Delugach		
Publication:	ICCS	Year:	2007
Acronym:	bachmeyer07-iccs	Pages:	13
DOI/URL:	10.1007/978-3-540-73681-3_14		

Summary

In [3], Bachmeyer*et al.* present *conceptual graph feature model*. Conceptual graphs are a formalism to express knowledge. Using this transformation, they provide an algorithm that is used to compute analysis.

Analyses

Paradigm:	Ad–hoc, Conceptual Graph		
FM notation:	Basic FMs	Extended FM:	No
Formalization:	Yes		

Operations

Operation	Alternative name	Support
valid product		Yes

Internal statistics

Self-citations:

FaMa ext.:

Ideas:

Table A.27: Bachmeyer et al. 2007 ICCS

Paper title:	FAMA: Tooling a Framework for the Automated Analysis of Feature Models		
Authors:	D. Benavides, S. Segura, P. Trinidad and A. Ruiz–Cortés		
Publication:	VaMoS	Year:	2007
Acronym:	benavides07-vamos	Pages:	6
DOI/URL:	Workshop Web Site		

Summary

In [11], the authors propose an Eclipse plug–in to edit and analyse feature models. The input and output format of the models is XML. Three different solvers are used to analyse those models: BDD, SAT and CSP solvers.

Analyses

Paradigm:	Propositional Logic and Constraint Programmming		
FM notation:	Cardinality-based FMs	Extended FM:	No
Formalization:	No		

Operations

Operation	Alternative name	Support
void FM		Yes
all products		Yes
number of products		Yes
commonality		Yes

Internal statistics

Self-citations:	benavides05-caise, batory06-cacm, benavides06-jisbd, benavides05-gttse, benavides06-splc, benavides05-ewmt
FaMa ext.:	
Ideas:	

Table A.28: Benavides et al. 2007 VaMoS

Paper title:	Generic semantics of feature diagrams		
Authors:	P. Schobbens, P. Heymans, J. Trigaux and Y. Bontemps		
Publication:	Computer Networks	Year:	2007
Acronym:	schobbens07-cn	Pages:	24
DOI/URL:	10.1016/j.comnet.2006.08.008		

Summary

Schobbens *et al.* [50] survey the state-of-the-art of feature model notations and compare the expressiveness and succinctness of the different proposals. In this context, the authors propose a generic semantic on feature models to generalize all the work studied (what they call *Free Feature Diagrams*). These formal semantics are provided using mathematical notation. They do not provide any explicit automated support for the automated analyses of feature models. However, they claim their formal semantics could be easily translated to propositional formulas for that purpose.

Analyses

Paradigm:			
FM notation:	Cardinality-based FMs	Extended FM:	No
Formalization:	Yes		

Operations

Operation	Alternative name	Support
valid product	product checking	No
void FM	satisfiability	No
refactoring	same valid models	No

Internal statistics

Self-citations:	benavides05-caise
FaMa ext.:	Include VFD as language
Ideas:	Catalog of operations and complexity for the operations

Table A.29: Schobbens et al. 2007 CN

Paper title:	Generic Feature-Based Software Composition		
Authors:	T. van der Storm		
Publication:	SC	Year:	2007
Acronym:	storm07-sc	Pages:	15
DOI/URL:	10.1007/978-3-540-77351-1_6		

Summary

Van der Storm [60] present a method to analyse feature model together with other artifacts. In terms of analysis they provide similar support to the one provided in [59] but slightly changing the mapping to propositional formulas to generate the BDD.

Analyses

Paradigm:	Propositional Logic		
FM notation:	Basic FMs	Extended FM:	No
Formalization:	No		

Operations

Operation	Alternative name	Support
void FM	internal consistency	Yes

Internal statistics

Self-citations:	batory06-cacm
FaMa ext.:	
Ideas:	

Table A.30: Storm 2007 SC

Paper title:	Verifying feature models using OWL		
Authors:	H. Wang, Y.F. Li, J. Sun, H. Zhang and J. Pan		
Publication:	Journal of Web Semantics	Year:	2007
Acronym:	wang07-jws	Pages:	13
DOI/URL:	10.1016/j.websem.2006.11.006		

Summary

Wang *et al.* [66] extend their previous proposal [65] with support for explanations by means of an OWL debugging tool. Additionally, a CASE tool for the visual development and analysis of feature models is presented.

Analyses

Paradigm:	Description Logic		
FM notation:	Basic FMs	Extended FM:	No
Formalization:	No		

Operations

Operation	Alternative name	Support
valid product	valid configuration	Yes
void FM	consistent FM	Yes
refactoring	semantically equivalence	Yes
explanations	debugging	Yes

Empirical evaluation

Number of instances:	1	Available:	No
Type of problems:	Large system	Format:	
Environment description:	Yes		

Internal statistics

Self-citations:

FaMa ext.:

Ideas:

Table A.31: Wang et al. 2007 JWS

A.8 2008

Paper title:	Evaluating formal properties of feature diagram languages		
Authors:	P. Heymans, P.Y. Schobbens, J.C. Trigaux, Y. Bontemps, R. Matulevicius and A. Classen		
Publication:	Software IET	Year:	2008
Acronym:	heymans08-iet	Pages:	22
DOI/URL:	10.1049/iet-sen:20070055		

Summary

In [27], Heymans *et al.* overview their previous works surveying feature diagram notations and providing them with a generic syntax and semantics [49, 50]. As a novel contribution, a general method to compare the semantic of feature diagrams is presented. Some notes about the complexity of several analysis operations are given.

Analyses

Paradigm:			
FM notation:		Extended FM:	No
Formalization:	No		

Operations

Operation	Alternative name	Support
valid product	model checking	No
void FM	satisfiability	No
refactoring		No

Internal statistics

Self-citations:	benavides05-caise, benavides06-jisbd
FaMa ext.:	
Ideas:	

Table A.32: Heymans et al. 2008 Software IET

Paper title:	Finding Contradictions in Feature Models		
Authors:	A. Hemakumar		
Publication:	ASPL	Year:	2008
Acronym:	hemakumar08-aspl	Pages:	8
DOI/URL:	Workshop Web site		

Summary

Hemakumar [26] proposes a new operation of analysis that is called *contradiction*. A contradiction is a also defined as a *conditional* dead feature. An *unconditionally* dead feature as described by [54] as a feature that is present in no product. A conditional dead feature is a feature that becomes dead after selecting one or more features. It is not obvious if this type of operation can be classified as an operation of analysis because, for this detection, some previous selection of features are required. However, Hemakumar proposes a method to statically detect conditional dead features, that is why we classify this operation as an operation of analysis. The method is based on model checking techniques and incremental consistency algorithms and what they reveal is that even small feature models the time required to detect those features is extremely high in general. The experiments also report that incremental consistency algorithms perform better.

Analyses

Paradigm:	Model checking and ad-hoc algorithms		
FM notation:	Basic FMs	Extended FM:	No
Formalization:	No		

Operations

Operation	Alternative name	Support
conditional dead features	contradiction	Yes

Empirical evaluation

Number of instances:	11	Available:	No
Type of problems:	Published	Format:	Grammar
Environment description:	Yes		

Internal statistics

Self-citations: benavides05-caise, batory06-cacm, trinidad08-jss

FaMa ext.: Yes

Ideas:

Table A.33: Hemakumar 2008 ASPL

Paper title:	Algebraic laws for feature models		
Authors:	R. Gheyi, T. Massoni and P. Borba		
Publication:	JUCS	Year:	2008
Acronym:	gheyi08-jucs	Pages:	19
DOI/URL:	10.3217/jucs-014-21-3573		

Summary

Gheyi *et al.* [25] present a set of algebraic laws in feature models to check configurability of FM refactorings. They use the PVS tools to do some analysis although this is not the main focus of the paper.

Analyses

Paradigm:	PVS		
FM notation:	Basic FMs	Extended FM:	No
Formalization:	Yes, using PVS		

Operations

Operation	Alternative name	Support
void FM		Yes
refactoring		Yes

Internal statistics

Self-citations:	benavides05-caise
FaMa ext.:	
Ideas:	

Table A.34: Gheyi et al. 2008 JUCS

Paper title:	Collaborative Product Configuration: Formalization and Efficient Algorithms for Dependency Analysis		
Authors:	M. Mendonça, D. Cowan, W. Malyk, and T. Oliveira		
Publication:	Journal of Software	Year:	2008
Acronym:	mendonca08-js	Pages:	14
DOI/URL:	Journal Web site		

Summary

In [37], Mendonça *et al.* extend their previous work [36] formalizing their approach and providing some algorithms for dependency analysis in the context of collaborative product configuration. Dependencies in the feature tree and cross-tree constraints are analysed using different techniques obtaining a noticeable improvement in efficency. Implicit support for atomic sets computation is also proposed.

Analyses

Paradigm:	Ad–hoc		
FM notation:	Cardinality-based FMs	Extended FM:	No
Formalization:	No		

Operations

Operation	Alternative name	Support
dependency analysis		Yes
atomic sets		Yes

Empirical evaluation

Number of instances:	4	Available:	No
Type of problems:	Random	Format:	
Environment description:	Yes		

Internal statistics

Self-citations:	benavides07-vamos
FaMa ext.:	Yes
Ideas:	Process tree constraints and cross-tree constraints using different techniques to improve efficiency

Table A.35: Mendonca 2008 JS

Paper title:	Efficient compilation techniques for large scale feature models		
Authors:	M. Mendonça, A. Wasowski, K. Czarnecki and D. Cowan:		
Publication:	GPCE	Year:	2008
Acronym:	mendonca08-gpce	Pages:	9
DOI/URL:	10.1145/1449913.1449918		

Summary

Mendoca *et al.* [39] survey existing heuristics to compile a feature model problems into a BDD representation. Using an experimental platform existing BDD heuristics fail to scale for large feature models of up to 2,000 features. They propose new heuristics for BDD ordering that are shown to be largely better than traditional ones. For defining their heuristics, three new operations of analysis are defined, namely: *i*) extra constraint representativeness *ii*) lowest common ancestor and *iii*) roots. For detailed definition of this operations we refer the reader to [39, page 14].

Analyses

Paradigm:	Propositional Logic (BDD)		
FM notation:	Basic FMs	Extended FM:	No
Formalization:	Yes		

Operations

Operation	Alternative name	Support
CTC representativeness		Yes
lowest common ancestor		Yes
roots		Yes
atomic sets		Yes

Empirical evaluation

Number of instances:		Available:	Yes
Type of problems:	Published and random	Format:	XML
Environment description:	Yes		

Internal statistics

Self-citations:	batory06-cacm
FaMa ext.:	
Ideas:	Apply ECR as an heuristic for GFT [58]. Apply the same ideas to CSP (heuristics for CSPs)

Table A.36: Mendonça et al. 2008 GPCE

Paper title:	Knowledge Based Method to Validate Feature Models		
Authors:	A. Osman, S. Phon-Amnuaisuk and C.K. Ho		
Publication:	ASPL	Year:	2008
Acronym:	osman08-aspl	Pages:	9
DOI/URL:	Workshop Web site		

Summary

Osman *et al.* [41] propose a knowledge base method to validate feature models. In their work, feature models are represented as a knowledge base containing predicates and rules defined using first order logic. During the configuration process, choices are added to the knowledge base by means of new rules which are then accepted or rejected if constraints are not fulfilled. Inconsistencies and redundancies are identified by looking for specific causes of error. The lacking of any proofs about the completeness of their approach is the weakest point of their work.

Analyses

Paradigm:	Propositional Logic (First Oder Logic)		
FM notation:	Cardinality-based FMs	Extended FM:	Yes
Formalization:	Yes		

Operations

Operation	Alternative name	Support
void FM		Yes
optimization		Yes
dead features		Yes
explanations		Yes
c. explanations	corrections	Yes

Internal statistics

Self-citations:	benavides05-caise, benavides06-jisbd, trinidad06-caise, trinidad08-jss, batory06-cacm
FaMa ext.:	
Ideas:	

Table A.37: Osman et al. 2008 ASPL

Paper title:	Automated Analysis of Feature Models using Atomic Sets		
Authors:	S. Segura		
Publication:	ASPL (SPLC workshop)	Year:	2008
Acronym:	segura08-aspl	Pages:	7
DOI/URL:	Workshop Web site		

Summary

Segura [51] proposes using atomic sets as a generic pre-processing technique for the automated analysis of feature models. An algorithm for their computation and a performance evaluation measuring its effectiveness are presented.

Analyses

Paradigm:	Propositional Logic and Constraint Programmming		
FM notation:	Basic FMs	Extended FM:	No
Formalization:	No		

Operations

Operation	Alternative name	Support
void FM		Yes
number of products		Yes
atomic sets		Yes

Empirical evaluation

Number of instances:	200	Available:	No
Type of problems:	Random	Format:	
Environment description:	Yes		

Internal statistics

Self-citations:	benavides05-caise, batory06-cacm, benavides06-jisbd, benavides06-splc, benavides07-vamos, trinidad08-splc
FaMa ext.:	
Ideas:	

Table A.38: Segura 2008 ASPL

Paper title:	Automated Diagnosis of Product-line Configuration Errors in Feature Models.		
Authors:	J. White, D. Benavides, D. C. Schmidt, P. Trinidad and A. Ruiz-Cortés		
Publication:	SPLC	Year:	2008
Acronym:	white08-splc	Pages:	10
DOI/URL:	10.1109/SPLC.2008.16		

Summary

White *et al.* [70] propose a method called CURE (Configuration Understanding and REmedy). CURE allows detecting conflicts in a given configuration and propose changes in the configuration in terms of features to be selected or deselected that remedy the problem. Their technique is based on translating a feature model into a CSP adding some extra variables in order to detect and explain the possible errors after applying optimization operations. CURE also proposes some extensions the method in order to make it more scalable. These extensions are based on the way the optimization operations are performed. In one of the possible extensions they propose adding cost attributes to features and proposing changes in the configuration minimizing the total cost of the change. Finally, empirical results are presented showing the scalability of the approach to feature model with over 5,000 features.

Analyses

Paradigm:	Constraint Programmming		
FM notation:	Basic FMs	Extended FM:	Yes
Formalization:	No		

Operations

Operation	Alternative name	Support
valid product	valid configuration	Yes
c. explanations	configuration error diagnosis	Yes

Empirical evaluation

Number of instances:		Available:	No
Type of problems:	random	Format:	
Environment description:	Yes		

Internal statistics

Self-citations:	benavides05-caise, batory06-cacm, benavides07-vamos, trinidad08-jss
FaMa ext.:	
Ideas:	Compare the performance of SAT solver implementing this operation

Table A.39: White et al. 2008 SPLC

Paper title:	Filtered Cartesian Flattening: An Approximation Technique for Optimally Selecting Features while Adhering to Resource Constraints		
Authors:	J. White and D. Schmidt		
Publication:	ASPL	Year:	2008
Acronym:	white08-aspl	Pages:	8
DOI/URL:	Personal web page		

Summary

White *et al.* [69] propose a method called *Filtered Cartesian Flattering* to map the problem of optimally selecting a set of features according to several constraints to a *Multi–dimensional Multi–choice Knapsack Problem* and then they use several existing algorithms to this problem that perform much faster while offering an aproximate answer.

Analyses

Paradigm:	MKKP, specific algorithms		
FM notation:	Basic FMs	Extended FM:	Yes
Formalization:	No		

Operations

Operation	Alternative name	Support
optimization	optimal feature selection	Yes

Empirical evaluation

Number of instances:	18,500	Available:	
Type of problems:	Random	Format:	
Environment description:	Yes		

Internal statistics

Self-citations:	benavides05-caise
FaMa ext.:	include this method
Ideas:	Apply the ideas of JA Parejo to the same problem and scenario

Table A.40: White et al. 2008 ASPL

Paper title:	Automated error analysis for the agilization of feature modeling		
Authors:	P. Trinidad, D. Benavides, A. Durán, A. Ruiz-Cortés and M. Toro		
Publication:	JSS	Year:	2008
Acronym:	trinidad08-jss	Pages:	14
DOI/URL:	10.1016/j.jss.2007.10.030		

Summary

Trinidad *et al.* [55] focus on the detection and explanation of errors on feature models. To this purpose, the authors propose a framework structured in three levels: a feature model level in which the problem of error treatment is described, a diagnosis level, where an abstract solution relying on Reiter's theory of diagnosis [45] is proposed, and an implementation layer, where the abstract solution is implemented using constraint programming. Through the usage of this framework the authors provide support for the detection of dead features and explanations (e.g. detailing the causes that make a FM model to be void).

Analyses

Paradigm:	Constraint Programmming		
FM notation:	Basic FMs	Extended FM:	No
Formalization:	Yes		

Operations

Operation	Alternative name	Support
void FM		Yes
dead features		Yes
false optional	full mandatory	Yes
explanations		Yes

Internal statistics

Self-citations:	benavides05-caise, batory06-cacm, benavides07-vamos
FaMa ext.:	
Ideas:	

Table A.41: Trinidad et al. 2008 JSS

Paper title:	A BDD-Based Approach to Verifying Clone-Enabled Feature Models' Constraints and Customization		
Authors:	W. Zhang, H. Yan, H. Zhao and Z. Jin		
Publication:	ICSR	Year:	2008
Acronym:	zhang08-icsr	Pages:	14
DOI/URL:	10.1007/978-3-540-68073-4_18		

Summary

Zhang *et al.* [76] deal with the problem of analysis of cloned features when using cardinality–based feature models. They propose using a BDD approach to analyse feature models and they compare their proposal with respect to their previous work [75] concluding that the BDD representation is more efficient.

Analyses

Paradigm:	Propositional Logic		
FM notation:	Cardinality-based FMs	Extended FM:	No
Formalization:	Yes		

Operations

Operation	Alternative name	Support
void FM	verification of FM constraints	Yes
dead features		Yes
false optional	chance to be removed	Yes

Empirical evaluation

Number of instances:	40	Available:	No
Type of problems:	Manual	Format:	
Environment description:	Yes		

Internal statistics

Self-citations:

FaMa ext.:

Ideas:

Table A.42: Zhang et al. 2008 ICSR

A.9 2009

Paper title:	Applying semantic web technology to feature modeling		
Authors:	L. Abo and F. Kleinermann and O. De Troyer		
Publication:	SAC	Year:	2009
Acronym:	abo09-sac	Pages:	5
DOI/URL:	10.1145/1529282.1529563		

Summary

In [1], Abo *et al.* propose using semantic web technologies for analysis of feature models. They use OWL, SWRL for modelling and the Pellet for reasoning and tool support.

Analyses

Paradigm:	Description Logic		
FM notation:	Basic FMs	Extended FM:	Yes
Formalization:	No		

Operations

Operation	Alternative name	Support
dead features		Yes
explanations		Yes

Internal statistics

Self-citations:	benavides05-caise,benavides06-cacm
FaMa ext.:	Use this approach to detect dead features
Ideas:	Compare computation of dead features with respect to SAT–based analysis

Table A.43: Abo et al. 2009 SAC

Paper title:	Analysis of Feature Models using Generalised Feature Trees		
Authors:	P.van den Broek and I. Galvão		
Publication:	VaMoS	Year:	2009
Acronym:	broek09-vamos	Pages:	7
DOI/URL:	Workshop proceedings		

Summary

Van den Broek *et al.* [58] propose enabling the analyses of feature model by transforming these into generalised feature trees and computing some of their properties. A *generalised feature tree* is a feature model in which cross-tree constraints are removed and features can have multiple occurrences. Some algorithms and a executable specification in the functional programming language Miranda is provided. The strength of their proposal lies in the efficiency of the analysis operation. However, the time to construct a generalised feature tree is exponential in the number of cross-tree constraint being this the main limitation of the approach.

Analyses

Paradigm:	Ad–hoc		
FM notation:	Basic FMs	Extended FM:	No
Formalization:	No		

Operations

Operation	Alternative name	Support
void FM	existence of products	Yes
all products	list of products	Yes
number of products		Yes
filter	product with certain features	Yes
dead features		Yes
explanations		Yes

Internal statistics

Self-citations:	benavides05-caise
FaMa ext.:	Include GFT
Ideas:	

Table A.44: Broek et al. 2009 VaMoS

Paper title:	Inferring Information from Feature Diagrams to Product Line Economic Models		
Authors:	D. Fernandez-Amoros, R. Heradio and J. Cerrada		
Publication:	SPLC	Year:	2009
Acronym:	fernandez09-splc	Pages:	10
DOI/URL:	–		

Summary

Fernandez *et al.* [23] propose an algorithm to compute the total number of products on what they call *Neutral Feature Trees*, these trees allows complex cross-tree constraints. Computing the total number of products they are able to calculate also *homogeneity* of a feature tree as well as *commonality* of a given feature. They finally compare the computational complexity of their approach with respect to previous work.

Analyses

Paradigm:	ad-hoc algorithms		
FM notation:	Cardinality-based FMs	Extended FM:	No
Formalization:	Yes		

Operations

Operation	Alternative name	Support
number of products		Yes
commonality		Yes
homogeneity		Yes

Internal statistics

Self-citations:	benavides07-phd,trinidad08-splc
FaMa ext.:	Include this algorithm in FaMa
Ideas:	

Table A.45: Fernandez et al. 2009 SPLC

Paper title:	SAT–based analysis of feature models is easy		
Authors:	M. Mendonca and A. Wasowski and K. Czarnecki		
Publication:	SPLC	Year:	2009
Acronym:	mendonca09-splc	Pages:	10
DOI/URL:	–		

Summary

Mendoca *et al.* [38] present empirical evidences that some analysis of feature models are easily tractable by state of the art SAT–solvers. They specially report the absence of the phase transition phenomena in their experiments. For experiments and analysis they use SAT4J solver.

Analyses

Paradigm:	Propositional Logic		
FM notation:	Basic FMs	Extended FM:	No
Formalization:	Yes		

Operations

Operation	Alternative name	Support
void FM	FM consistency	Yes
core features	common feature	Yes
dead features		Yes
CTC representativeness	cross–tree constraint ratio	Yes

Empirical evaluation

Number of instances:	52,500	Available:	Yes
Type of problems:	random	Format:	XML, grammar
Environment description:	Yes		

Internal statistics

Self-citations:	batory06-cacm,benavides07-phd,benavides07-vamos,benavides05-caise,segura09-vamos,trinidad06-caise,trinidad08-splc,white08-splc
FaMa ext.:	
Ideas:	Run the same type of experiments with CSP solvers

Table A.46: Mendonca et al. 2009 SPLC

Paper title:	Using First Order Logic to Validate Feature Model		
Authors:	A. Osman, S. Phon-Amnuaisuk and C.K. Ho		
Publication:	VaMoS	Year:	2009
Acronym:	osman09-vamos	Pages:	4
DOI/URL:	Workshop proceedings		

Summary

Osman *et al.* [42] extend the work presented in [41] with a new operations to prevent inconsistencies. According to the authors, this basically decomposes complex dependencies (e.g. many–to–many) into one–to–one requires/excludes constraints. However, the proposed example present some inconsistencies and a formal proof is missed. Some performance results are also presented.

Analyses

Paradigm:	Ad–hoc		
FM notation:	Cardinality-based FMs	Extended FM:	No
Formalization:	Yes		

Operations

Operation	Alternative name	Support
dead features		Yes
explanations		Yes

Empirical evaluation

Number of instances:		Available:	No
Type of problems:	random	Format:	
Environment description:	No		

Internal statistics

Self-citations:	benavides05-caise
FaMa ext.:	
Ideas:	

Table A.47: Osman et al. 2009 VaMoS

Paper title:	VMWare: Tool Support for Automatic Verification of Structural and Semantic Correctness in Product Line Models		
Authors:	C. Salinesi, C. Rolland and R. Mazo		
Publication:	VaMoS	Year:	2009
Acronym:	salinesi09-vamos	Pages:	4
DOI/URL:	Workshop proceedings		

Summary

In [48], Salinesi *et al.* presents an approach for the automated verification of feature-based product line models. As part of their proposal, the authors list a collection of correctness criteria of feature models and present a prototype tool (i.e. VMWare) implementing them using ad-hoc algorithms. Most criteria are related to structural aspects checked at the metamodel level (e.g. root uniqueness).

Analyses

Paradigm:	Ad–hoc		
FM notation:	Cardinality-based FMs (FORE)	Extended FM:	No
Formalization:	No		

Operations

Operation	Alternative name	Support
void FM		Yes
dead features		Yes

Internal statistics

Self-citations:	batory06-cacm,trinidad08-splc
FaMa ext.:	
Ideas:	

Table A.48: Salinesi et al. 2009 VaMoS

Paper title:	Reasoning about Edits to Feature Models,		
Authors:	T. Thuem, D. Batory and C. Kaestner		
Publication:	ICSE	Year:	2009
Acronym:	thum09-icse	Pages:	11
DOI/URL:	FTP site		

Summary

Thum *et al.* [53] present an automated support for clasifying feature model edits, i.e. changes in an original feature model, according to a taxonomy. The operation of analysis takes as input two feature models (the original one and the one after changes on the model) and classifies the second feature model as a *refactoring* (no products added, no products deleted), a *generalization* (some products added, no products deleted), a *specialization* (no products added, some products deleted) or an *arbitraty edit* (some products added and some products deleted). Their method is based on propositonal logic algorithms.

Analyses

Paradigm:	Propositional Logic		
FM notation:	Basic FMs	Extended FM:	No
Formalization:	Yes		

Operations

Operation	Alternative name	Support
refactoring		Yes
generalization		Yes
specialization		Yes
arbitrary edit		Yes

Empirical evaluation

Number of instances:		Available:	Yes
Type of problems:	Published and random	Format:	Grammar
Environment description:	Yes		

Internal statistics

Self-citations:	benavides05-caise, benavides07-phd, benavides06-jisbd, benavides06-splc, benavides05-seke, white08-splc
FaMa ext.:	Yes
Ideas:	

Table A.49: Thum et al. 2009 ICSE

Paper title:	Abductive Reasoning and Automated Analysis of Feature Models: How are they connected?		
Authors:	P. Trinidad and A. Ruiz–Cortés		
Publication:	VAMOS	Year:	2009
Acronym:	trinidad09-vamos	Pages:	9
DOI/URL:	Workshop web site		

Summary

Trinidad *et al.* [57] present a calalog of operations based on previous work [8]. What they add is a classification of the operations of analysis in terms of *abductive* and *deductive* reasoning. Typical operations of analysis are classified as deductive operations meanwhile, operations of so–called explanations are classified as abductive operations. A complete catalogue of operations is presented and classified although no automated support is explicitly proposed.

Analyses

Paradigm:			
FM notation:		Extended FM:	Yes
Formalization:			

Operations

Operation	Alternative name	Support
valid product		No
void FM		No
all products		No
refactoring		No
core features		No
variant features		No
number of products		No
variability		No
commonality		No
filter		No
optimization		No
dead features		No
explanations		No
wrong cardinality		No
false optional		No
valid partial configuration		No

Internal statistics

Self-citations:	benavides05-caise, benavides07-phd, benavides06-jisbd, trindidad08-jss, trinidad08-splc, white08-splc
FaMa ext.:	
Ideas:	

Table A.50: Trinidad et al. 2009 VAMOS

Paper title:	Selecting Highly Optimal Architectural Feature Sets with Filtered Cartesian Flattening		
Authors:	J. White, B. Doughtery and D. Schmidt		
Publication:	JSS	Year:	2009
Acronym:	white09-jss	Pages:	36
DOI/URL:	DOI:10.1016/j.jss.2009.02.011		

Summary

White *et al.* [67] present an extension to their previous work [69]. The same method is presented but giving enough details to make it reproducible since some details were missed in their previous work. The method is called *Filtered Cartesian Flattering* which map the problem of optimally selecting a set of features according to several constraints to a *Multi–dimensional Multi–choice Knapsack Problem* and then they use several existing algorithms to this problem that perform much faster while offering an aproximate answer. Some empirical evidences of the scalability of the method are presented. In addtion, a method to generate large–scale feature selection problems randomly is also reported.

Analyses

Paradigm:	MKKP, specific algorithms		
FM notation:	Basic FMs	Extended FM:	Yes
Formalization:	No		

Operations

Operation	Alternative name	Support
optimization	optimal feature selection	Yes

Empirical evaluation

Number of instances:	500,000	Available:	
Type of problems:	Random	Format:	
Environment description:	Yes		

Internal statistics

Self-citations:	benavides05-caise, benavides07-vamos, white08-aspl
FaMa ext.:	include this method
Ideas:	Apply the ideas of JA Parejo to the same problem and scenario

Table A.51: White et al. 2009 JSS

Paper title:	Automated Reasoning for Multi-step Software Product-line Configuration Problems		
Authors:	J. White, B. Doughtery, D. Schmidt and D. Benavides		
Publication:	SPLC	Year:	2009
Acronym:	white09-splc	Pages:	10
DOI/URL:	Personal web page		

Summary

White *et al.* [68] propose an automated method to solve what they call *multi–step configuration problem*. A mapping from this type of problems to CSP is provided. The idea is to optimize the steps when configuring a feature model. From an original configuration to another in a given number of steps, which is the path tha optimize a given criteria.

Analyses

Paradigm:	Constraint Programmming		
FM notation:	Basic FMs	Extended FM:	Yes
Formalization:	Yes		

Operations

Operation	Alternative name	Support
multi–step configuration		Yes

Empirical evaluation

Number of instances:	1,000	Available:	Yes
Type of problems:	Random	Format:	
Environment description:	Yes		

Internal statistics

Self-citations:	benavides05-caise, benavides07-vamos, white08-splc
FaMa ext.:	include this operation
Ideas:	Apply the ideas of JA Parejo to this optimization problem as well

Table A.52: White et al. 2009 SPLC

Paper title:	An Optimization Strategy to Feature Models' Verification by Eliminating Verification-Irrelevant Features and Constraints		
Authors:	H. Yan and W. Zhang and H. Zhao and H. Mei		
Publication:	ICSR	Year:	2009
Acronym:	yan09-icsr	Pages:	11
DOI/URL:	10.1007/978-3-642-04211-9_7		

Summary

In [72], Yan *et al.* propose a method to eliminate irrelevant features and constraints in order to optimize feature model analysis. They provide experimental results showing the benefits of the approach using a BDD–based tool.

Analyses

Paradigm:	Propositional Logic		
FM notation:	Basic FMs	Extended FM:	No
Formalization:	No		

Operations

Operation	Alternative name	Support
void FM		Yes
dead features		Yes

Empirical evaluation

Number of instances:	-	Available:	No
Type of problems:	random	Format:	-
Environment description:	Yes		

Internal statistics

Self-citations:	segura08-splc,benavides08-splc
FaMa ext.:	Use this approach to simplify feature models
Ideas:	Compare computation when applying Sergio's genetic algorithm

Table A.53: Yan et al. 2009 ICSR

A.10 Papers out of the scope

Paper title:	Employing Fuzzy Logic in Feature Diagrams to Model Variability in Software Product-Lines		
Authors:	S. Robak and A. Pieczynski		
Publication:	ECBS	Year:	2003
Acronym:	robak03-ecbs	Pages:	7
DOI/URL:	10.1109/ECBS.2003.1194812		

Summary

Robak et al. [46] propose using fuzzy logic to annotate with weights variant features in feature models. According to the authors, this may result helpful to customize feature models adapting them to the different profiles of the skateholders. The automated analysis of feature models is out of the scope of the paper.

Analyses

Paradigm:	Fuzzy logic		
FM notation:	Basic FMs	Extended FM:	No
Formalization:	No		

Operations

Operation	Alternative name	Support

Internal statistics

Self-citations:	
FaMa ext.:	Yes
Ideas:	Analysis feature models using fuzzy logic

Table A.54: Robak et al. 2003 ECBS

Paper title:	Staged Configuration Using Feature Models		
Authors:	K. Czarnecki and S. Helsen and U. Eisenecker		
Publication:	SPLC	Year:	2004
Acronym:	czarnecki04-splc	Pages:	18
DOI/URL:	10.1007/b100081		

Summary

Czarneki *et al.* [14] propose cardinality–based feature models and present a claimed new concept called *staged configurations* which is equivalent to interactive configuration in the AI community. An UML metamodel for Cardinality-based FMs is presented later changed in [15]. Typical operations of specialization of Cardinality-based FMs are presented and discussed. From the perspective of automated analysis, no operation is presented.

Analyses

Paradigm:			
FM notation:	Cardinality-based FMs	Extended FM:	Yes
Formalization:	No		

Operations

Operation	Alternative name	Support

Internal statistics

Self-citations:	
FaMa ext.:	
Ideas:	Master thesis: use a FM of a web framework, e.g. struts or spring, and provide a generator for different scenarios: MVC applications, AOP stuff and so on.

Table A.55: Czarnecki et al. 2004 SPLC

Paper title:	Features with fuzzy probability		
Authors:	A. Pieczyriski, S. Robak and A. Walaszek-Babiszewska		
Publication:	ECBS	Year:	2004
Acronym:	pieczynski04-ecbs	Pages:	6
DOI/URL:	10.1109/ECBS.2004.1316715		

Summary

In [44], Pieczynski *et al.* extend their previous works [46, 47] dealing with the use of fuzzy logic and feature diagrams. As a novel contribution, the authors show how their approach can be used to study markets and make predictions. The implementation of an expert system is detailed but not in the context of the automated analysis of feature models.

Analyses

Paradigm:	Fuzzy logic		
FM notation:	Basic FMs	Extended FM:	No
Formalization:	No		

Operations

Operation	Alternative name	Support

Internal statistics

Self-citations:

FaMa ext.:

Ideas:

Table A.56: Pieczynski et al. 2004 ECBS

Paper title:	Application of Fuzzy Weighted Feature Diagrams to Model Variability in Software Families		
Authors:	S. Robak and A. Pieczynski		
Publication:	ICAISC	Year:	2004
Acronym:	robak04-icaisc	Pages:	6
DOI/URL:	10.1007/b98109		

Summary

In [47] Robak et al. summarize the work presented in [46]. The automated analysis of feature models is out of the scope of the paper.

Analyses

Paradigm:	Fuzzy logic		
FM notation:	Basic FMs	Extended FM:	No
Formalization:	No		

Operations

Operation	Alternative name	Support

Internal statistics

Self-citations:	
FaMa ext.:	Yes
Ideas:	Analysis feature models using fuzzy logic

Table A.57: Robak et al. 2004 ICAISC

Paper title:	Grammatically Interpreting Feature Compositions		
Authors:	W. Zhao, B. Bryant, F. Cao, R. Raje, M. Auguston, C. Burt and A. Olson		
Publication:	SEKE	Year:	2004
Acronym:	zhao04-seke	Pages:	7
DOI/URL:	Personal web page		

Summary

Zhao *et al.* [77] present a meta–language to specify feature models. For that purpose, they use *two levels grammars*. No explicit operations of analysis description is provided.

Analyses

Paradigm:	Two Level Grammar		
FM notation:	Basic FMs	Extended FM:	Yes
Formalization:	No		

Operations

Operation	Alternative name	Support

Internal statistics

Self-citations:

FaMa ext.:

Ideas:

Table A.58: Zhao et al. 2004 SEKE

Paper title:	Formalizing cardinality-based feature models and their specialization		
Authors:	K. Czarnecki, S. Helsen and U. Eisenecker		
Publication:	SPIP	Year:	2005
Acronym:	czarnecki05b-spip	Pages:	24
DOI/URL:	10.1002/spip.213		

Summary

Czarneki *et al.* [15] formalize cardinality–based feature models and define the difference between a specialization and a configuration. They provide some typical specialization steps. A mapping of Cardinality-based FMsto context–free grammars is provided. In terms of analysis this paper do not provide any explicit method or operation of analysis.

Analyses

Paradigm:			
FM notation:	Cardinality-based FMs	Extended FM:	Yes
Formalization:	Yes		

Operations

Operation	Alternative name	Support

Internal statistics

Self-citations:	
FaMa ext.:	
Ideas:	

Table A.59: Czarnecki et al. 2005 SPIP

Paper title:	Staged configuration through specialization and multilevel configuration of feature models		
Authors:	K. Czarnecki, S. Helsen and U. Eisenecker		
Publication:	SPIP	Year:	2005
Acronym:	czarnecki05b-spip	Pages:	27
DOI/URL:	10.1002/spip.225		

Summary

Czarneki *et al.* [15] present again cardinality–based feature models and define the difference between staged configuration using *stepwise specialization* of a feature model where only one feature model is involved or using what they call *multi–level* configuration where more than one stake holder can be part of the process and more than feature model is configured. The UML metamodel of Cardinality-based FMs presented in [14] is revisited. From the point of view of FM analysis, no operation is provided.

Analyses

Paradigm:			
FM notation:	Cardinality-based FMs	Extended FM:	Yes
Formalization:	No		

Operations

Operation	Alternative name	Support

Internal statistics

Self-citations:

FaMa ext.:

Ideas:

Table A.60: Czarnecki et al. 2005 SPIP (II)

Paper title:	Ontology-Based Feature Modeling and Application-Oriented Tailoring		
Authors:	X. Peng, W. Zhao, Y. Xue and Y. Wu		
Publication:	ICSR	Year:	2006
Acronym:	peng06-icsr	Pages:	14
DOI/URL:	10.1007/11763864_7		

Summary

In [43], Peng *et al.* present an ontology–based feature model meta–model. They specify such a meta–model using OWL [3]. Using the Jena API for description logic and semantic web, they propose several rules for syntactic validation of feature models. Validation of a feature model based on their meta–model is proposed. This paper is not on the scope of feature models because what they do is analysis on different artifacts and data of the SPL, for instance, they consider binding times in their analysis.

Analyses

Paradigm:	TBD		
FM notation:	Basic FMs	Extended FM:	No
Formalization:	No		

Operations

Operation	Alternative name	Support

Internal statistics

Self-citations:

FaMa ext.:

Ideas:

Table A.61: Peng et al. 2006 ICSR

[3]http://www.w3.org/TR/2003/PR-owl-ref-20031215/

Paper title:	A Process-Centric Approach for Coordinating Product Configuration Decisions		
Authors:	M. Mendonça, D. Cowan and T. Oliveira		
Publication:	HICSS	Year:	2007
Acronym:	mendonca07-hicss	Pages:	10
DOI/URL:	10.1109/HICSS.2007.27		

Summary

Mendoca *et al.* [35] present an approach for the support of collaborative product condiguration. The analysis of feture models is not part of the approach.

Analyses

Paradigm:			
FM notation:	Cardinality-based FMs	Extended FM:	No
Formalization:	No		

Operations

Operation	Alternative name	Support

Internal statistics

Self-citations:

FaMa ext.:

Ideas:

Table A.62: Mendonça et al. 2007 HICSS

Paper title:	Disambiguating the Documentation of Variability in Software Product Lines: A Separation of Concerns, Formalization and Automated Analysis		
Authors:	A. Metzger, K. Pohl, P. Heymans, P. Schobbens and G. Saval		
Publication:	RE	Year:	2007
Acronym:	metzger07-re	Pages:	11
DOI/URL:	10.1109/RE.2007.61		

Summary

Metzger *et al.* [40] propose using OVM models to document product line variability and feature models to document software variability (i.e. ability of the reusable artifact to be customized). A formalization of both types of models and their relationships is provided. Then, some reasoning operations to check the consistency between both models are proposed. The analyses of feature model is not directly addressed in this work because all the operations proposed are about the analysis of both models and their relationships at a time.

Analyses

Paradigm:	Propositional Logic		
FM notation:	VFD	Extended FM:	No
Formalization:	Yes		

Operations

Operation	Alternative name	Support
realizability		Yes
indentical combinations		Yes
commonality		Yes
dead variant features		Yes

Internal statistics

Self-citations:	benavides05-caise,benavides06-jisbd
FaMa ext.:	handle with multiple models
Ideas:	

Table A.63: Metzger et al. 2007 RE

Paper title:	Reasoning about Feature Models in Higher-Order Logic		
Authors:	M. Janota and J. Kiniry		
Publication:	SPLC	Year:	2007
Acronym:	janota07-splc	Pages:	10
DOI/URL:	10.1109/SPLINE.2007.36		

Summary

Janota *et al.* [30] formalize using high order logic a generic and configurable feature model meta–model. This meta–model can be instantiated using different specific feature model dialects (e.g basic and cardinality–based feature models). Using the theorem prover PVS, they allow some reasoning but at the meta–model level.

Analyses

Paradigm:	High order logic		
FM notation:	Cardinality-based FMs	Extended FM:	No
Formalization:	Yes		

Operations

Operation	Alternative name	Support

Internal statistics

Self-citations:	benavides05-caise
FaMa ext.:	
Ideas:	

Table A.64: Janota et al. 2007 SPLC

Paper title:	Feature Diagrams and Logics: There and Back Again		
Authors:	K. Czarnecki and A. Wasowski		
Publication:	SPLC	Year:	2007
Acronym:	czarnecki07-splc	Pages:	10
DOI/URL:	10.1109/SPLINE.2007.4339252		

Summary

Czarneki *et al.* [18] present some techniques and algorithms to restore a feature model from a set of propositional formulas. The process is semi–automatic. Although they mention some operations of analysis, these are out of the scope of the paper.

Analyses

Paradigm:	Propositional Logic		
FM notation:	Basic FMs	Extended FM:	No
Formalization:	No		

Operations

Operation	Alternative name	Support
void FM	Existence of configurations	No
dead features		Yes

Internal statistics

Self-citations:	benavides05-caise, benavides06-jisbd
FaMa ext.:	
Ideas:	

Table A.65: Czarnecki et al 2007 SPLC

Paper title:	Quality aware software product line engineering		
Authors:	L. Etxeberria, G. Sagardui and L. Belategi		
Publication:	JBCS	Year:	2008
Acronym:	etxeberria08-jbcs	Pages:	13
DOI/URL:	10.1590/S0104-65002008000100006		

Summary

Etxeberria *et al.* [] present a survey of existing approaches considering quality requirements in software product lines. The automated reasoning on feature attributes is mentioned as a desirable feature but it is not explicitly addressed in the paper.

Analyses

Paradigm:			
FM notation:		Extended FM:	No
Formalization:	No		

Operations

Operation	Alternative name	Support

Internal statistics

Self-citations:	benavides05-caise, benavides06-splc
FaMa ext.:	
Ideas:	

Table A.66: Etxeberria et al. 2008 JBCS

Paper title:	Sample Spaces and Feature Models: There and Back Again		
Authors:	K. Czarnecki, S. She and A. Wasowski		
Publication:	SPLC	Year:	2008
Acronym:	czarnecki08-splc	Pages:	10
DOI/URL:	10.1109/SPLC.2008.49		

Summary

We do not consider this paper because the feature model notation and semantics used change all the operations catalog Czarneki *et al.* [17] propose what they call *probabilistic feature models* (PFMs) which are formalized as a set of formulas in a certain probabilistic logic. PFMs extend feature model with *soft constraints*, constraints that are probabilistic. Later, the concept of *feature model mining* is introduced. Feature model mining aims to retrieve models from a set of products for reverse engineering purposes. In terms of analysis the only operation mention and discussed. This contribution is novel in terms of analysis since this is the first time that analysis using probabilistic logig is introduced. We do not consider this paper because the feature model notation and semantics used change all the operations catalog

Analyses

Paradigm:	prbabilistic logic		
FM notation:	Probabilistic FMs	Extended FM:	Yes
Formalization:	Yes		

Operations

Operation	Alternative name	Support
probabilistic consistency	consistency of PFM	Yes

Internal statistics

Self-citations:	batory06-cacm
FaMa ext.:	Include a PFM reasoner
Ideas:	Apply PPL to SOA group, Usar http://jopt.sourceforge.net/ en FaMa, proyecto de Master

Table A.67: Czarnecki et al 2008 SPLC

Paper title:	Do SAT Solvers Make Good Configurators?		
Authors:	M. Janota		
Publication:	ASPL	Year:	2008
Acronym:	janota08-aspl	Pages:	5
DOI/URL:	Workshop Web site		

Summary

Janota [29] tackles the problem of configuration of feature models. The author suggest using SAT solvers for this purpose and present some novel algorithms to enable interactive configuration guaranteeing backtrack-freeness.

Analyses

Paradigm:	Propositional Logic		
FM notation:		Extended FM:	
Formalization:	No		

Operations

Operation	Alternative name	Support

Internal statistics

Self-citations:	benavides05-caise
FaMa ext.:	Implement a FaMa configurator
Ideas:	

Table A.68: Janota 2008 ASPL

Paper title:	Semantic Annotations of Feature Models for Dynamic Product Configuration in Ubiquitous Environments		
Authors:	N. Kaviani, B. Mohabbati, D. Gasevic and M. Finke		
Publication:	SWESE	Year:	2008
Acronym:	kaviani08-swese	Pages:	15
DOI/URL:	Workshop Web site		

Summary

Koviani *et al.* [] propose using ontologies to annotate feature models with non-functional requirements. This ontologies are used to check the consistency between the capabilities provided by external services and the requirements of the product line. The authors suggest using description logic reasoners to automate the process. The analysis of feature model is not addressed in the paper. Rather, the analyses focus on configuration issues and realizability checkings.

Analyses

Paradigm:	Description Logic		
FM notation:	Basic FMs	Extended FM:	Yes
Formalization:	Yes		

Operations

Operation	Alternative name	Support

Internal statistics

Self-citations:	benavides05-caise
FaMa ext.:	
Ideas:	

Table A.69: Kaviani et al. 2008 SWESE

Paper title:	Decision-making coordination in collaborative product configuration		
Authors:	M. Mendonça, T.T. Bartolomei and D. Cowan		
Publication:	SAC	Year:	2008
Acronym:	mendonca08-sac	Pages:	6
DOI/URL:	10.1145/1363686.1363715		

Summary

Mendonça *et al.* [36] present an approach to collaborative product configuration supporting teamwork decision-making in the context of product configuration. Some hypergraph-baed reasoning techniques are used to deal with dependency analysis during the configuration process. The analysis of feature models is not addressed.

Analyses

Paradigm:			
FM notation:	Cardinality-based FMs	Extended FM:	No
Formalization:	No		

Operations

Operation	Alternative name	Support

Internal statistics

Self-citations:

FaMa ext.:

Ideas:

Table A.70: Mendonca et al. 08 SAC

Paper title:	An OWL- Based Approach for Integration in Collaborative Feature Modelling		
Authors:	L.A. Zaid, G. Houben, O. Troyer and F. Kleinermann		
Publication:	SWESE	Year:	2008
Acronym:	swese08-swese	Pages:	8
DOI/URL:	Workshop Web site		

Summary

Zaid *et al.* [73] propose an OWL-based approach for enabling the merging of feature models in the context of collaboration work. A mechanism to resolve merge conflict automatically is also provided. The anlaysis of feature models is out of the scope of the approach.

Analyses

Paradigm:	Description Logic (OWL)		
FM notation:	Basic FMs	Extended FM:	Yes
Formalization:	No		

Operations

Operation	Alternative name	Support

Internal statistics

Self-citations:	benavides05-caise, batory06-sacm
FaMa ext.:	
Ideas:	

Table A.71: Zaid et al. 2008 SWESE

Paper title:	Towards Tool Support for the Configuration of Non-Functional Properties in SPLs		
Authors:	j. Sincero and W. Schröder-Preikschat and O. Spinczyk		
Publication:	HICSS	Year:	2007
Acronym:	sincero09-hicss	Pages:	7
DOI/URL:	10.1109/HICSS.2009.472		

Summary

In [28], Sincero *et al.* present an approach to consider non–functional properties in feature model configurations. They use the Linux Kernel Configurator tool and claim the usage of BDD–based support for analysis. We discard this work from the survey because they do not make explicit reference to any analysis operation on their tool.

Analyses

Paradigm:	Propositional Logic		
FM notation:	Basic FMs	Extended FM:	Yes
Formalization:	No		

Operations

Operation	Alternative name	Support

Internal statistics

Self-citations:	benavides05-caise,benavides06-splc,benavides07-vamos
FaMa ext.:	
Ideas:	Using the kernel configurator to model feature models

Table A.72: Sincero et al. 2009 HICSS

Bibliography

[1] L. Abo, F. Kleinermann, and O. De Troyer. Applying semantic web technology to feature modeling. In *SAC*, pages 1252–1256, 2009.

[2] V. Alves, R. Gheyi, T. Massoni, U. Kulesza, P. Borba, and C. Lucena. Refactoring product lines. In *GPCE '06: Proceedings of the 5th international conference on Generative programming and component engineering*, pages 201–210, New York, NY, USA, 2006. ACM Press.

[3] R. Bachmeyer and H. Delugach. A conceptual graph approach to feature modeling. In *ICCS*, pages 179–191, 2007.

[4] D. Batory. Feature models, grammars, and propositional formulas. In *Software Product Lines Conference*, volume 3714 of *Lecture Notes in Computer Sciences*, pages 7–20. Springer–Verlag, 2005.

[5] D. Batory, D. Benavides, and A. Ruiz-Cortés. Automated analysis of feature models: Challenges ahead. *Communications of the ACM*, December:45–47, 2006.

[6] D. Benavides, A. Ruiz-Cortés, and P. Trinidad. Automated reasoning on feature models. In *Advanced Information Systems Engineering: 17th International Conference, CAiSE 2005*, volume 3520 of *Lecture Notes in Computer Sciences*, pages 491–503. Springer–Verlag, 2005.

[7] D. Benavides, A. Ruiz-Cortés, and P. Trinidad. Using constraint programming to reason on feature models. In *The Seventeenth International Conference on Software Engineering and Knowledge Engineering, SEKE 2005*, pages 677–682, 2005.

[8] D. Benavides, A. Ruiz-Cortés, P. Trinidad, and S. Segura. A survey on the automated analyses of feture models. In *Jornadas de Ingeniería del Software y Bases de Datos (JISBD)*, 2006.

[9] D. Benavides, S. Segura, P. Trinidad, and A. Ruiz-Cortés. A first step towards a framework for the automated analysis of feature models. In *Managing Variability for Software Product Lines: Working With Variability Mechanisms*, 2006.

[10] D. Benavides, S. Segura, P. Trinidad, and A. Ruiz-Cortés. Using java csp solvers in the automated analyses of feature models. *Lecture Notes in Computer Science*, 4143:389–398, 2006.

[11] D. Benavides, S. Segura, P. Trinidad, and A. Ruiz-Cortés. FAMA: Tooling a framework for the automated analysis of feature models. In *Proceeding of the First International Workshop on Variability Modelling of Software-intensive Systems (VAMOS)*, pages 129–134, 2007.

[12] F. Cao, B. Bryant, C. Burt, Z. Huang, R. Raje, A. Olson, and M. Auguston. Automating feature-oriented domain analysis. In *International Conference on Software Engineering Research and Practice (SERP'03)*, pages 944–949, June 2003.

[13] K. Czarnecki and U.W. Eisenecker. *Generative Programming: Methods, Techniques, and Applications*. Addison–Wesley, may 2000. ISBN 0–201–30977–7.

[14] K. Czarnecki, S. Helsen, and U. Eisenecker. Staged configuration using feature models. In *Proceedings of the Third Software Product Line Conference 2004*, volume 3154 of *Lecture Notes in Computer Sciences*, pages 266–282. Springer–Verlag, 2004.

[15] K. Czarnecki, Simon Helsen, and Ulrich W. Eisenecker. Formalizing cardinality-based feature models and their specialization. *Software Process: Improvement and Practice*, 10(1):7–29, 2005.

[16] K. Czarnecki and P. Kim. Cardinality-based feature modeling and constraints: A progress report. In *Proceedings of the International Workshop on Software Factories At OOPSLA 2005*, 2005.

[17] K. Czarnecki, S. She, and A. Wasowski. Sample spaces and feature models: There and back again. In *proceedings of the Software Product Line Conference (SPLC)*, pages 22–31, 2008.

[18] Krzysztof Czarnecki and Andrzej Wasowski. Feature diagrams and logics: There and back again. In *Software Product Lines, 11th International Conference, SPLC, Proceedings*, pages 23–34. IEEE Computer Society, 2007.

[19] M. Dean and G. Schreiber. OWL web ontology language reference. W3C recommendation, W3C, February 2004.

[20] Various developers. SMV system , www.cs.cmu.edu/∼modelcheck. published on line.

[21] O. Djebbi, C. Salinesi, and D. Diaz. Deriving product line requirements: the red-pl guidance approach. *Asia-Pacific Software Engineering Conference*, 0:494–501, 2007.

[22] S. Fan and N. Zhang. Feature model based on description logics. In *Knowledge-Based Intelligent Information and Engineering Systems*, 2006.

[23] D. Fernandez-Amoros, R. Heradio, and J. Cerrada. Inferring information from feature diagrams to product line economic models. In *Proceedings of the Sofware Product Line Conference*, 2009.

[24] R. Gheyi, T. Massoni, and P. Borba. A theory for feature models in alloy. In *Proceedings of the ACM SIGSOFY First Alloy Workshop*, pages 71–80, Portland, United States, nov 2006.

[25] R. Gheyi, T. Massoni, and P. Borba. Algebraic laws for feature models. *Journal of Universal Computer Science*, 14(21):3573–3591, 2008.

[26] Adithya Hemakumar. Finding contradictions in feature models. In *First International Workshop on Analyses of Software Product Lines (ASPL'08)*, pages 183–190, 2008.

[27] P. Heymans, P.Y. Schobbens, J.C. Trigaux, Y. Bontemps, R. Matulevicius, and A. Classen. Evaluating formal properties of feature diagram languages. *Software IET*, 2(3):281–302, 2008.

[28] j. Sincero, W. Schröder-Preikschat, and O. Spinczyk. Towards tool support for the configuration of non-functional properties in spls. In *HICSS*, pages 1–7, 2009.

[29] M. Janota. Do sat solvers make good configurators? In *First International Workshop on Analyses of Software Product Lines (ASPL'08)*, pages 191–195, 2008.

[30] Mikolás Janota and Joseph Kiniry. Reasoning about feature models in higher-order logic. In *Software Product Lines, 11th International Conference, SPLC, Proceedings*, pages 13–22, 2007.

[31] K. Kang, S. Cohen, J. Hess, W. Novak, and S. Peterson. Feature–Oriented Domain Analysis (FODA) Feasibility Study. Technical Report CMU/SEI-90-TR-21, Software Engineering Institute, Carnegie Mellon University, November 1990.

[32] P. Klint. A meta-environment for generating programming environments. *ACM Trans. Softw. Eng. Methodol.*, 2(2):176–201, April 1993.

[33] M. Mannion. Using first-order logic for product line model validation. In *Proceedings of the Second Software Product Line Conference (SPLC2)*, LNCS 2379, pages 176–187, San Diego, CA, 2002. Springer.

[34] M. Mannion and J. Camara. Theorem proving for product line model verification. In *Software Product-Family Engineering (PFE)*, volume 3014 of *Lecture Notes in Computer Science*, pages 211–224. Springer Berlin / Heidelberg, 2003.

[35] M. Mendonca, D. Cowan, and T. Oliveira. A process-centric approach for coordinating product configuration decisions. In *Proceedings of the 40th Annual Hawaii International Conference on System Sciences (HICSS '07)*, page 283a, Washington, DC, USA, 2007. IEEE Computer Society.

[36] M. Mendonça, T.T. Bartolomei, and D. Cowan. Decision-making coordination in collaborative product configuration. In *Proceedings of the 2008 ACM symposium on Applied computing (SAC '08)*, pages 108–113, New York, NY, USA, 2008. ACM.

[37] M. Mendonça, D.D. Cowan, W. Malyk, and T. Oliveira. Collaborative product configuration: Formalization and efficient algorithms for dependency analysis. *Journal of Software*, 3(2):69–82, 2008.

[38] M. Mendonça, A. Wasowski, and K. Czarnecki. SAT–based analysis of feature models is easy. In *Proceedings of the Sofware Product Line Conference*, 2009.

[39] Marcílio Mendonça, Andrzej Wasowski, Krzysztof Czarnecki, and Donald D. Cowan. Efficient compilation techniques for large scale feature models. In *Generative Programming and Component Engineering, 7th International Conference, GPCE , Proceedings*, pages 13–22, 2008.

[40] A. Metzger, K. Pohl, P. Heymans, P. Schobbens, and G. Saval. Disambiguating the documentation of variability in software product lines: A separation of concerns, formalization and automated analysis. In *Requirements Engineering Conference, 2007. RE '07. 15th IEEE International*, pages 243–253, 2007.

[41] A. Osman, S. Phon-Amnuaisuk, and C.K. Ho. Knowledge based method to validate feature models. In *First International Workshop on Analyses of Software Product Lines*, pages 217–225, 2008.

[42] A. Osman, S. Phon-Amnuaisuk, and C.K. Ho. Using first order logic to validate feature model. In *Third International Workshop on Variability Modelling in Software-intensive Systems (VaMoS)*, pages 169–172, 2009.

[43] X. Peng, W. Zhao, Y. Xue, and Y. Wu. Ontology-based feature modeling and application-oriented tailoring. In *ICSR*, pages 87–100, 2006.

[44] A. Pieczyriski, S. Robak, and A. Walaszek-Babiszewska. Features with fuzzy probability. In *Proceedings of the 11th IEEE Engineering of Computer-Based Systems*, pages 323–328, May 2004.

[45] R Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1):57–95, 1987.

[46] S. Robak and A. Pieczynski. Employing fuzzy logic in feature diagrams to model variability in software product-lines. In *10th IEEE International Conference and Workshop on the Engineering of Computer-Based Systems*, volume 0, page 305, Los Alamitos, CA, USA, 2003. IEEE Computer Society.

[47] S. Robak and A. Pieczyski. Application of fuzzy weighted feature diagrams to model variability in software families. In *Artificial Intelligence and Soft Computing - ICAISC 2004*, volume 3070 of *Lecture Notes in Computer Science*, pages 370–375. Springer Berlin / Heidelberg, 2004.

[48] C. Salinesi, C. Rolland, and R. Mazo. Vmware: Tool support for automatic verification of structural and semantic correctness in product line models. In *Third International Workshop on Variability Modelling of Software-intensive Systems*, pages 173–176, 2009.

[49] P. Schobbens, P. Heymans, J. Trigaux, and Y. Bontemps. Feature Diagrams: A Survey and A Formal Semantics. In *Proceedings of the 14th IEEE International Requirements Engineering Conference (RE'06)*, Minneapolis, Minnesota, USA, September 2006.

[50] P. Schobbens, J.C. Trigaux P. Heymans, and Y. Bontemps. Generic semantics of feature diagrams. *Computer Networks*, 51(2):456–479, Feb 2007.

[51] S. Segura. Automated analysis of feature models using atomic sets. In *First Workshop on Analyses of Software Product Lines (ASPL 2008). SPLC'08*, pages 201–207, Limerick, Ireland, September 2008.

[52] J. Sun, H. Zhang, Y.F. Li, and H. Wang. Formal semantics and verification for feature modeling. In *Proceedings of the 10th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS)*, 2005.

[53] T. Thüm, D. Batory, and C. Kästner. Reasoning about edits to feature models. In *International Conference on Software Engineering*, pages 254–264, 2009.

[54] P. Trinidad, D. Benavides, A. Durán, A. Ruiz-Cortés, and M. Toro. Automated error analysis for the agilization of feature modeling. *Journal of Systems and Software*, 81(6):883–896, 2008.

[55] P. Trinidad, D. Benavides, A. Durán, A. Ruiz-Cortés, and M. Toro. Automated error analysis for the agilization of feature modeling. *Journal of Systems and Software*, 81(6):883–896, 2008.

[56] P. Trinidad, D. Benavides, and A. Ruiz-Cortés. A first step detecting inconsistencies in feature models. In *CAiSE Short Paper Proceedings*, 2006.

[57] Pablo Trinidad and Antonio Ruiz Cortés. Abductive reasoning and automated analysis of feature models: How are they connected? In *Third International Workshop on Variability Modelling of Software-Intensive Systems. Proceedings*, pages 145–153, 2009.

[58] P. van den Broek and I. Galvao. Analysis of feature models using generalised feature trees. In *Third International Workshop on Variability Modelling of Software-intensive Systems*, number 29 in ICB-Research Report, pages 29–35, Essen, Germany, January 2009. Universität Duisburg-Essen.

[59] T. van der Storm. Variability and component composition. In *Software Reuse: Methods, Techniques and Tools: 8th International Conference, ICSR 2004. Proceedings*, volume 3107 of *Lecutre Notes in Computer Sciences*, pages 157–166. Springer, July 2004.

[60] Tijs van der Storm. Generic feature-based software composition. In *Software Composition*, volume 4829 of *LNCS*, pages 66–80. Springer, 2007.

[61] A. van Deursen and P. Klint. Domain–specific language design requires feature descriptions. *Journal of Computing and Information Technology*, 10(1):1–17, 2002.

[62] T. von der Massen and H. Lichter. Requiline: A requirements engineering tool for software product lines. In F. van der Linden, editor, *Proceedings of the Fifth International Workshop on Product Family Engineering (PFE)*, LNCS 3014, Siena, Italy, 2003. Springer Verlag.

[63] T. von der Massen and H. Lichter. Deficiencies in feature models. In Tomi Mannisto and Jan Bosch, editors, *Workshop on Software Variability Management for Product Derivation - Towards Tool Support*, 2004.

[64] T. von der Massen and H. Litcher. Determining the variation degree of feature models. In *Software Product Lines Conference, LNCS 3714*, pages 82–88, 2005.

[65] H. Wang, Y. Li, J. Sun, H. Zhang, and J. Pan. A semantic web approach to feature modeling and verification. In *Workshop on Semantic Web Enabled Software Engineering (SWESE'05)*, November 2005.

[66] H. Wang, Y.F. Li, J. un, H. Zhang, and J. Pan. Verifying Feature Models using OWL. *Journal of Web Semantics*, 5:117–129, June 2007.

[67] J. White, B. Doughtery, and D. Schmidt. Selecting highly optimal architectural feature sets with filtered cartesian flattening. *Journal of Systems and Software*, 82(8):1268–1284, 2009.

[68] J. White, B. Doughtery, D. Schmidt, and D. Benavides. Automated reasoning for multi-step software product-line configuration problems. In *Proceedings of the Sofware Product Line Conference*, pages 11–20, 2009.

[69] J. White and D. Schmidt. Filtered cartesian flattening: An approximation technique for optimally selecting features while adhering to resource constraints. In *First International Workshop on Analyses of Software Product Lines (ASPL)*, pages 209–216, 2008.

[70] J. White, D. Schmidt, D. Benavides P. Trinidad, and A. Ruiz-Cortes. Automated diagnosis of product-line configuration errors in feature models. In *Proceedings of the Sofware Product Line Conference*, 2008.

[71] J. Wookcock and J. Davies. *Using Z: Specification, Refinement, and Proof.* Prentice–Hal, 1996.

[72] H. Yan, W. Zhang, H. Zhao, and H. Mei. An optimization strategy to feature models' verification by eliminating verification-irrelevant features and constraints. In *ICSR*, pages 65–75, 2009.

[73] L.A. Zaid, G. Houben, O. Troyer , and F. Kleinermann. An owl- based approach for integration in collaborative feature modelling. In *4th International Workshop on Semantic Web Enabled Software Engineering, Proceedings*, 2008.

[74] W. Zhang, H. Mei, and H. Zhao. Feature-driven requirement dependency analysis and high-level software design. *Requirements Engineering*, 11(3):205–220, June 2006.

[75] W. Zhang, H. Zhao, and H. Mei. A propositional logic-based method for verification of feature models. In J. Davies, editor, *ICFEM 2004*, volume 3308, pages 115–130. Springer–Verlag, 2004.

[76] Wei Zhang, Hua Yan, Haiyan Zhao, and Zhi Jin. A bdd–based approach to verifying clone-enabled feature models' constraints and customization. In *High Confidence Software Reuse in Large Systems, 10th International Conference on Software Reuse, ICSR, Proceedings*, LNCS, pages 186–199. Springer, 2008.

[77] W. Zhao, B. Bryant, F. Cao, R. Raje, M. Auguston, C. Burt, and A. Olson. Grammatically interpreting feature compositions. In *Proceedings of the Sixteenth International Conference on Software Engineering & Knowledge Engineering (SEKE'2004)*, 2004.