Current situation facing the needs of the scenarios from the deliverables I2.1.1 and I2.2.1

Daniel Romero, Mireille Blay-Fornarino, Philippe Collet, Laurence Duchien, Philippe Renevier, Simon Urli

Current situation facing the needs of the scenarios from the deliverables I2.1.1 and I2.2.1

Rapport explicitant l'état des lieux confronté aux besoins des scénarios issus des livrables I2.1.1 et I2.2.1

Livrable D-3.1.1

Team MODALIS & RAINBOW (Université de Nice-Sophia, I3S)
Team ADAM (Université de Lille, LIFL)
Daniel Romero, Mireille Blay-Fornarino, Philippe Collet, Laurence Duchien, Philippe Renevier, Simon Urli

juin 2012

Table of Contents

1. Introduction

In this document we present the main issues that we have to face in order to define a Software Product Line (SPL) for Broadcasting Systems. These issues were identified through requirement analysis and refactoring of SEDUITE[1] which are described in two internal deliverables:

a) D.2.2.1: Introduces the requirements (functional and non-functional) of a Broadcasting System by using a case study based on large gatherings (e.g., concerts, competitions, parties, etc.).
b) D.2.1.1: Explains the definition of SEDUITE as a SPL by identifying the different assets and products that make part of it.

In particular, from each deliverable different questions were raised. We use these questions to identify the issues that we need to face and to guide the redaction of this document. We classify the questions according to three main topics: (i) user assistance (cf. Section 2), (ii) building and evolution of the SPL (cf. Section 3) and (iii) kinds of variability (cf. Section 4) The questions from the D.2.2.1 deliverable are identified with *I.x* and those from D.2.1.1 with *Q.x*. In both cases, the '*x*' represents the number of the question in the deliverable. Additionally, we include the results of two questionnaires intended for consumers of information (i.e., professor and students) from broadcasting system in academic institutions.

2. User assistance

In this section we include all the topics related to functionalities of the product line that aim to assist final users in the usage of the SPL itself or the derived products. We choose to model variability using feature models [KKL+98]: a feature is a "characteristic of a concept that is relevant to some stakeholder of the concept"[Cza02].

2.1 Product descriptions (Q-1)

In order to help managers to define configurations, it would be useful to have descriptions related to the different products that make part of the SPL. Such descriptions are related to variability and additional information about products (when possible). The issue here is how to define such descriptions and link them with the specific products. In particular, in the source case two solutions are proposed:

a) *Deduction of descriptions*: The constraints in the feature model can be used to give managers information

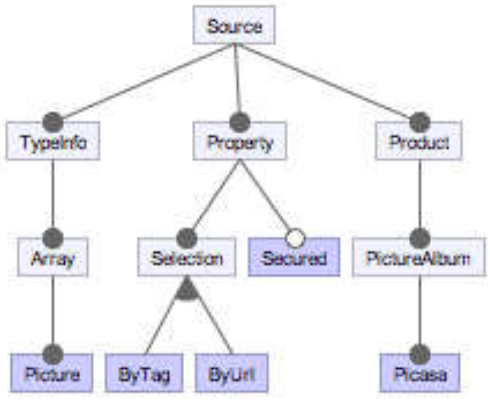

Figure 1. FM of the Picasa Source

[1] http://www.jseduite.org/doku/

about the product capabilities [PBL05, BJS09]. For example, from the FM depicted in Figure 1 we can deduce "Picasa is a Picture Album that supports the secured property"

b) *Descriptions as metadata*: The language to define FM from Familiar[2] can be extended to support the addition of metadata containing the product description. Following the Picasa example, we can extend its Familiar description in this way

```
Product: Picasa //Service supported by Google…// Secured
//property indicating that the service requires
authentication…//
```

where the italics between // represent the metadata.

2.2 Default Values (Q-9)

Another important issue in the user assistance is the management of default values. In particular, it is necessary to determine a suitable way of defining them in the SPL. In some cases, such as the specification of a default behavior, the task is simple. However, in situations where the domain values and types are complex, the definition of default values becomes harder. For instance, in the source case different kinds of parameters (simple and complex) can be found and it is necessary not only to specify the default value but also how to present the domain values to users. Indeed, this information can help the user in her choices (e.g., choose a weather service taking as parameter names of French cities or GPS coordinates) (cf. Section 4).
In [CHE05], authors introduce the concept of multilevel configuration and extend the feature model formalism. In this work, features can have optional attributes of a certain type, and those attributes can have an optional value.

2.3 Design of the Configurator Interface (I2)

This issue derives from the default value selection and the need for a configuration tool adaptable according to FM evolutions. In particular, because of the variability of sources (cf. Section 4) in terms of parameter types, the design of the Configurator interface becomes complex. We deal with different types of information (e.g., lists, dates and GPS coordinates) but also with the requirement of providing an ergonomic interface that helps managers in the definition of configurations. Deriving ergonomic GUI for adapted configuration tools is still an open research problem, with first steps being recently made [BPH12].

[2] https://nyx.unice.fr/projects/familiar/

3. Building and Evolution of the Software Product Line

This section presents all the issues related to the building and evolution of the SPL.

3.1 Addition of new assets (Q-2, Q-5, I3)

The definition of different FMs related to SEDUITE enables us to identify two main issues related to the incorporation of new assets into the SPL: (i) *lack of a unify vocabulary* and (ii) *semantic of features with the same name*. The former refers to the fact that two features with different names can express the same thing. For example, the types "Picture Album" and "Picture Collection" express the same type of product. The acceptation of this kind of redundancy makes the evolution of SPL difficult. On the other hand, the semantic of features is associated with presence of features with the same name in different sub trees of the FM. This is the case of the "Date" feature in policies, which is used to trigger alarms but also to filter the information.
Using ontologies could be a way to deal with these issues at the vocabulary level [CKK06, FFA+10]. To support consistency maintenance during FMs evolution, a basic solution can be to apply a set of atomic operations on FMs [GW10], but other solutions can also be envisaged by using reasoning and composition tools [ACLF12], coupled with differentiation techniques over the feature models [AHC+12].

3.2 Relationships between features (Q-6)

The definition of the FMs of policies allows us to detect an issue in dependency relationships between features. In general, when there is a dependency toward a variation point (or general feature) and one of its variants (or specialized feature) is selected, the depending feature is always deselected. An example will clarify this issue. Figure 2-a depicts the FM of the `ShuffleInfo` policy. This FM specifies that the policy can be applied on any information of `Array` type. Figure 2-b shows a fragment of the FM resulting of the fusion of different policies via Familiar. In this FM, the selection of one of the variants of `Array` (i.e., Pictures, News or Info) automatically produces the exclusion of the `ShuffleInfo` policy. The only way to choose this policy is to do it explicitly or selecting the `Array` type. This limitation can lead to a Configurator with an inconsistent behavior that will confuse users.

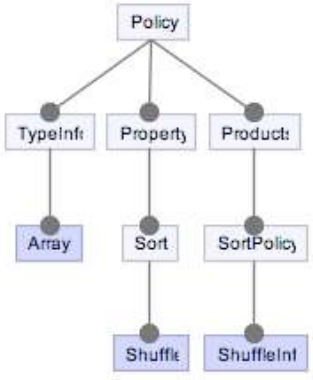

Figure 2. a) FM of the Shuffle Info Policy

The cause for these problems is the evolution of a family of products. Each integration of a new product in the FM implies to verify compatibility with all affected products [BH01]. In [BLP04], the variability model differentiates variation points and variants. "The variation point captures the location and reason of variation. Variants capture the different possibilities of a variation." This distinction could help to detect error due to evolution when a "variation" becomes a variation point.

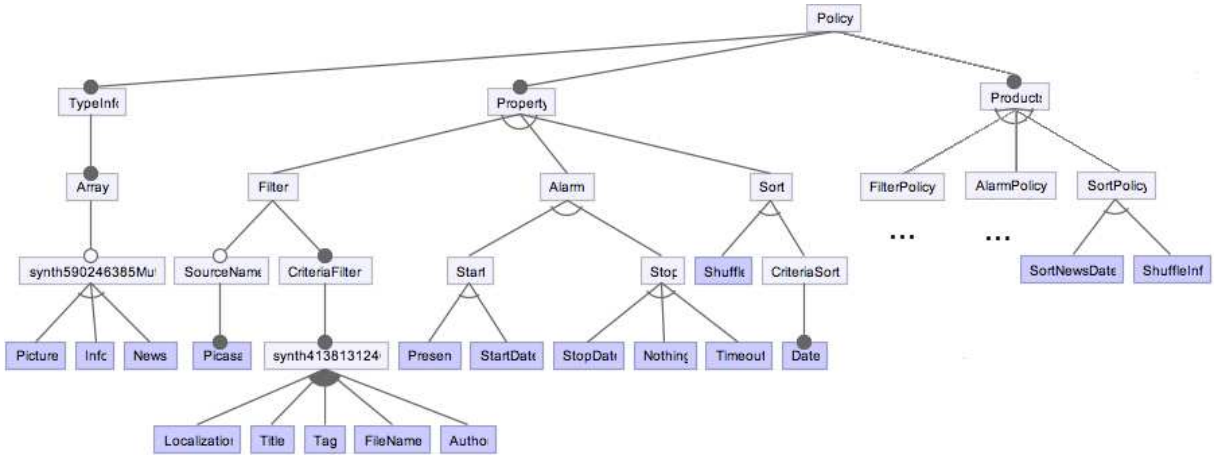

Figure 2. b) FM of Policies

In [BCW10], authors propose Clafer, a concise notation for meta-models, feature models, mixtures of meta- and feature models. It supports specialization, which seems necessary here to express some of the relationships between features.

3.3 Management of Information

A recurring issue in the building of the SPL is associated with the management of the information required (parameters) and produced by sources. We analyze them in the following subsections.

3.3.1 Presentation/Selection of Hierarchies of Information (I1)

This issue is related to the parameterization of sources via hierarchies of information. In the case of the "Item List" source, it can be necessary to display a list of people, of times, resources, etc. How to present and enable the selection of such information? This problem reaches the one presented in 3.2. Dealing with the structure seems to be a way to consider new information types and the evolution of the line[BCW10]. But, unlike works such as [KMPY06], we also focus in a change of abstraction level: a skilled person with no knowledge of software design will use the product line. The difficulty is to express the variability of types of information (structure) but at the same time in a level that everyone (domain expert and designer) can understand.

3.3.2 Validity of the Information (I6)

In the context of sources, some information are valid during a specific period of time. For example, a calendar source just provides the event of a given date and a news service only gives news of the day. A first assumption is therefore that the source just sends valid information at this time. However, if the source does not do it, we need to use a behavior policy to enabling its filtering. Thus, we need to indicate the filtering or not of information from sources to assure the integrity of the SPL, which makes in some cases filtering policies mandatory (cf. Section 4.2).

3.4 Management of Interruptions (Emergency) (I5)

In the SPL SEDUITE there are policies for dealing with the display time of sources and interruption policies to present specific information in short periods of time. However, the usage of both can be confused to users. A clear presentation of both at the interface

level (of the Configurator) is required while supporting their composition at implementation level. We could explore for that the ability to express explicitly the relationship between feature models and architecture models as supported by VML[ZSS+10].

In conclusion, we have shown that a broadcasting system is described by hundreds of features that are valid for different criteria-sets (e.g., a source must be displayed using a specific renderer on a given layout). We may agree on the fact that it will be a great challenge to determine this information from a feature tree. Describing this variability in one FM is consequently difficult, tedious and error-prone. This makes evolution management and design of an appropriate configuration tool very challenging activities. With the needs to integrate third-party components and services, a single feature model is clearly insufficient to represent the feature variability of some product lines [KMPY06, ACLF12]. Our broadcasting system product line is a typical example of this issue. We have to deal with a composition of SPLs and therefore to tackle challenges originating in multi-instance composition of SPLs [SS09, Bos10] including the composition of inter-dependent software artifacts [PCBD10][P11].

4. Kinds of variability

The variability in a SPL represents the opportunities to derive new products by allowing the selection between different options. In the case of the SPL of SEDUITE we identify two main kinds of variability: (i) ***Customizable Variability*** and (ii) ***Traditional Variability***. A third kind of variability that does not influence the derivation of new products is the ***Data Type Variability***. They are summarized in Table 1 and derived from Q3 and Q8 questions.

Table 1. Kinds of Variability in the SEDUITE SPL

Kind of Variability	Description	Associated FMs	Associated Feature	Example
Traditional Variability	Variability related to several products but that is not customizable (i.e., there is no values that can be defined by the manager).	Sources, Policies, Renderers	Information Type	- The WeatherBug and Weather2 sources have "Weather" as information type. - The Thumbails Picture, Full Screen Picture and Mosaic Album renderers have "Picture" as information type.
		Sources, Policy, Layouts	Property	- The Sort News Date and Shuffle Info Policies have a "sort" property. - The WeatherBug and Twitter sources can have a "secured" property. - All the layouts have a target and a usage.
		Layout	Design	- All the layouts have a design.
		Layout	Zones	- All the layouts have a "big square centre" zone.
				- The full screen
		Renderer, Zones, Behaviors	Elements	picture and mosaic album renderers have a "graphic" element. - All the zones have a "content" element. - All the behaviors

		Behaviors	Animation	- All the scrolling behaviors have a "scrolling" animation.
		Behaviors	Numbers	- The simple repeat behavior and the basic behavior have "all" as number.
Customizable Variability	Options that can be not only used to select a product but also for its customization.	Sources, Policies	Parameters	- The WeatherBug source has a city code as parameter. The Weather2 source requires the latitude and longitude of the city. - In the case of policies, the parameterization depends on the associated source.
Data type Variability	Variability related to several types of parameter	N/A	N/A	- Dates, GPS coordinates, item lists

The *Traditional Variability* represents the possibilities to derive products in the SPL but that do not required a customization from the manager after the generation of the Broadcasting System. The different FMs identified in the D.2.1.1 deliverable (i.e. Source, Policies, Renderers, Layouts and Behaviors) present different features related to this kind of variability. For example WeatherBug and Weather2 sources have "Weather" as information type, which represents a criteria to select sources. However, it is not possible to provide a customization for such a value.
On the other hand, the *Customizable Variability* refers to the options that can be customized by managers. In particular, it is related to parameters of sources and policies that can be used as criteria to select a product of both categories but at the same time their values can vary once the Broadcasting System is generated. For example, two weather sources, WeatherBug and Weather2, need different parameters in order to provide the predictions. The former requires a city code (predefined by the same source) and the latter one the longitude and the latitude. In the policy case, the parameterization depends on the sources on which the policy is applied.

Finally, the *Data Type Variability* refers to the different types of data that we need to deal with. Different sources and policies bring with them richness in terms of data types

associated to their configuration parameters. For instance, a display policy uses dates to define the start and end of displaying source information and a weather source can use the longitude and latitude to retrieve the predictions. Even if this kind of variability does not lead to the derivation or selection, we need to consider it because of the "Configurator Interface". As a matter of fact, this variability imposes an additional complexity on the design of such interface, which has to be as friendly as possible to make the manager works easier.

4.1 Issues related to the Variability (Q-4, Q-8, I1)

The issue from the variability, and more specifically from the customizable variability, comes from different parameterizations for sources and policies with the same type of information. The different parameters have domain values specified by the sources (e.g., code of cities) or restricted by the type of the parameter itself (e.g., the longitude and the latitude). In a similar way, the domain values can be simple (e.g., a threshold in a policy) or complex (e.g. an enumeration, list of values or GPS coordinates). It is necessary to determine a suitable way to provide default values and provide the domain values to managers.

4.2 Implicit Constraints (Q7, I7)

The analysis of SEDUITE also allowed us to discover implicit constraints between products of the SPL [CHE05a]. In particular, the identified dependencies are related to policies and the urgency source. Below we present such dependencies.

a) Policy Constraints: The behavioral policies are applied on sources. However, it cannot be performed in any way. Depending on the kind of provided information, sources just accept specific policies. For instance, a "filter picture" policy cannot be used on sources providing news. On the other hand, there are also constraints regarding the cardinality of policies on sources. In fact, in some cases a policy can be applied only once (e.g., a threshold policy). Multiple policies can be combined and applied on the same source and others are mandatory (e.g., a source providing information that does not change a lot requires a caching policy). The issue here is to specify all these constraints at the SPL.
b) Emergency Source Constraints: Constraints from urgency source are related to configurations, layouts and zones. All valid configurations require an urgency source. Therefore, by default each configuration has one and it has to be defined as a criteria to consider a configuration as valid. Additionally, a layout exposes one and only one "zone" of emergency, which has to be connected to emergency source. Again here, the issues is the specification all these constraints at the SPL.

To deal with feature interaction issues, we will refer to [MPDBF12] considering interactions as a variation point in the configuration process.

5. Summary of the spread questionnaires

In order to obtain feedback from the users of information in academic institutions (i.e., teachers, students and administrative staff), we elaborate a questionnaire that was spread in the University Lille 1 and the University of Nice-Sophia-Antipolis. The former

uses an industrial broadcasting system, VisioSense[3]. The latter promotes the usage of SEDUITE. The questionnaire enabled us to get feedback related to the relevance of the broadcasted information, the identification of new sources and the style of the presentation of the information (design, colors, typefaces). We use these three categories to present the results

5.1 Results University Lille 1

In Lille 96 people answer the questionnaire. From these people, 15 people (16%) have never seen the screen content. The main reasons are the lack of time (9 people) and the position of the screen that does not help the visualization (3 people). Figure 3 summarizes these results.

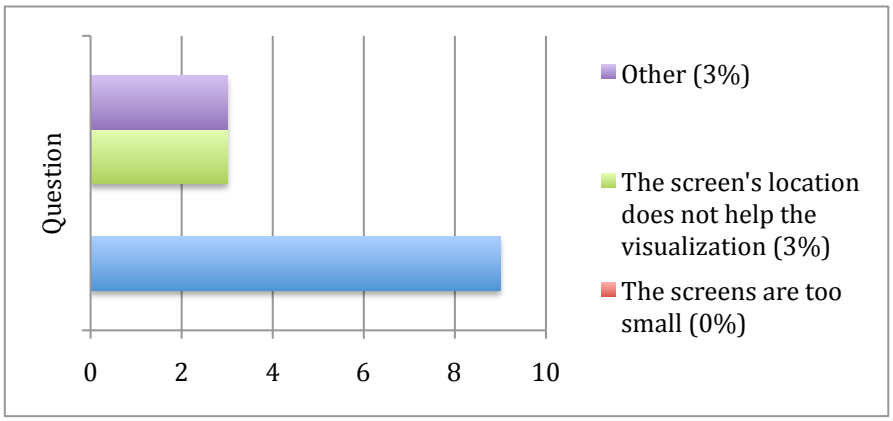

Figure 3. Main reasons people do not see the screen content

5.1.1 Relevance of the information

In Lille 1, 53 people (55%) consider that the displayed information on the screen is suitable, 9 (9%) think is unsuitable and 19 (20%) that it is not enough. Furthermore, 63 (66%) people consider that information is changed with an acceptable frequency and 18 (19%) that is not. People answer "no" say that information should be changed every day or at least 3 times by week.

5.1.2 Identification of sources

People saying the information is unsuitable would like to have information related to the practical information (e.g., menu restaurant), news concerning the university and specifically the computer science (e.g., new laws related to informatics, job offers, new technologies, new courses), real time information from sources such as Twitter, (indicating, e.g., absence of teachers, changes in classroom or schedules). People also suggest that it would be interesting to have access to displayed information via Internet.

[3] http://www.visiosense.fr/

5.1.3 Presentation of information

Regarding the style of presentation, 39 people (41%) consider that it is suitable, 33 (34%) that it is simple and 9 (9%) unsuitable. This latter group comments on the typefaces and the combination of colors. Furthermore, 49 people (51%) express that the amount of information displayed on the screens is enough, 29 (30%) that it is not enough, and 3 (3%) that is a lot. On the other hand, 56 people (58%) think that the display time (for each information) is suitable, 9 (9%) that it is too short and 16 (17%) that it is too long.

5.2 Results University Nice- Sophia

In Nice 98 people answered the questionnaire. All the participants have already watched on the screens.

5.2.1 Relevance of the information

In Nice, 41 people (42%) find that the displayed information is suitable, 11 (11%) that is suitable but not enough, 36 (37%) that is not enough and 6 (6%) that is unsuitable. Regarding the update frequency of information, 33 (34%) consider that it is enough, 12 (12%) that it is not, and 51 (52%) do not rest enough time in front of the screen to need it.

5.2.2 Identification of sources

People that consider that displayed information is not enough suggest the inclusion of information about bus and train schedules, schedule of computer rooms, news related to the university (e.g., dates of academic council, deadline of Erasmus candidatures, VIP visits), classes schedule and location, technology news, and building news (e.g., exceptional closures, problems with the elevator).
Additionally, in the University Nice-Sophia-Antipolis version of the questionnaire, we ask for opportunities to interact with the broadcasting system. In general, people suggest the display of schedule for each student by using the student card, and the opportunity to select the information that the person did not have the time to read.

5.2.3 Presentation of information

Considering the presentation, 37 people (38%) find that it is suitable, 19 (19%) unsuitable and 42 (43%) simple. The principal remarks are related to cut phrases, typefaces, font colors and images. The display time is enough for 38 people (39%), it is not enough for 22 (22%), it is both for 34 (35%) and it is too long for 4 (4%).

5.3 Analysis and summary of results

The relevance of the information does not directly impact the building of the SPL but personalization (selection of sources and tuning of cache policies) made by managers. Comments about such relevance have to be shared with current clients of the broadcasting systems in order to improve their use. However, the relevance of the information represents an opportunity to identify new and more suitable sources

according to the current context. In particular we identify the following sources (that does not exist in the current SEDUITE version):

- *Bus schedule*: Source providing different schedules (of the current date) of bus stations near to the campus.
- *Train schedule*: Sources providing different schedules of train's arrivals/departs to the nearest train station to the campus.
- *Computer science news*: Source collecting news about new technologies and informatics laws.
- *Job offers*: Source providing job offers regarding the undergraduate programs.
- *Building news*: Source providing information about the specific building where the screen is located.
- *University Live*: Source providing information about relevant events in the university such as VIP visits and key dates.

In the case of the presentation, the personalization impacts the GUI. Depending on the broadcasting system, the managers can be helped in the task associated with the GUI design. VisioSense enables the integration of different kinds of files (PDF, PPT, videos). Managers are therefore completely responsible to find a suitable design. In the case of SEDUITE as a SPL, we help managers in this task by providing templates elaborated considering the remarks from the questionnaires.

6. Conclusion

In this document we have presented several issues that we need to tackle in order to build a SPL for Broadcasting Systems. Such issues are the result of the analysis and reflections about domain requirements and different products and concepts that make part of the SPL. The issues were classified in three categories: *User Assistance*, *Building and Evolution of the SPL* and *Kinds of Variability*.
In User Assistance we included different aspects that we need to consider in the SPL to help final users to employ derived products. Such aspects include descriptions of sources, ergonomics of graphical interfaces and the management of default values. Regarding the Building and Evolution of the SPL, we identified aspects that can lead to inconsistencies in the SPL (e.g., the generalization and specialization of features and the management of information related to sources) and that make the inclusion of new assets difficult. In the kinds of variability we identify the customizable variability as an important issue to deal with in order to build a suitable Configurator interface. All these identified issues will be solved, at least partially, throughout the project.

Finally, we also presented the collected results from a questionnaire answered by students and teachers of University Lille 1 and Nice-Sophia-Antipolis. The obtained feedback enables us to improve aspects of the SPL related to sources and layouts.

7. References

[ACLF12] Mathieu Acher, Philippe Collet, Philippe Lahire, Robert France. "Separation of Concerns in Feature Modeling: Support and Applications" in Proceedings of the Aspect-Oriented Software Development (AOSD'12), pages 1-12, ACM, mar 2012

[AHC+12] Mathieu Acher, Patrick Heymans, Philippe Collet, Clément Quinton, Philippe Lahire, Philippe Merle. "Feature Model Differences" in Proceedings of the 24th International Conference on Advanced Information Systems Engineering (CAiSE'12), Springer, 25-29 june 2012

[BCW10] K Bak, K Czarnecki, and A Wasowski. ***Feature and meta-models in Clafer: Mixed,specialized, and coupled***. Proceedings of the Third international conference on Software language engineering, pages 102-122, 2010.

[BH01] Jan Bosch and Mattias Högström. ***Product instantiation in software product lines: A case study***. Lecture Notes in Computer Science, 2177:147 -162, 2001.

[BPH12] Quentin Boucher, Gilles Perrouin, Patrick Heymans: Deriving configuration interfaces from feature models: a vision paper. VaMoS 2012 workshop: 37-44

[BLP04] Stan Bühne, Kim Lauenroth, and Klaus Pohl. ***Why is it not Sufficient to Model Requirements Variability with Feature Models***? Automotive Requirements Engineering, (September):5 -12, 2004.

[BJS09] Goetz Botterweck, M Janota, and Denny Schneeweiss. ***A design of a configurable feature model configurator***. VAMOS 2009, 2009.

[Bos10] Jan Bosch. ***Toward Compositional Software Product Lines***. IEEE Software, 27(3):29-34, 2010.

[CHE05] Krzysztof Czarnecki, Simon Helsen, and Ulrich Eisenecker. ***Staged configuration through specialization and multilevel configuration of feature models.*** Software Process: Improvement and Practice, 10(2):143 -169, 2005.

[CHE05a] Krzysztof Czarnecki, ,Simon Helsen, Ulrich W. Eisenecker. **Formalizing cardinality-based feature models and their specialization.** Software Process: Improvement and Practice, 10(1) :7-29,2005.

[CKK06] Krzysztof Czarnecki, Chang Hwan Peter Kim, and Karl Trygve Kalleberg. ***Feature Models are Views on Ontologies***. In 10th International Software Product Line Conference SPLC06, volume 1, pages 41 -51. IEEE Computer Society, 2006.

[Cza02] Krzysztof Czarnecki. ***Generative Programming: Methods, Techniques, and Applications Tutorial Abstract***, Software Reuse Methods Techniques and Tools2002.

[FFA+10] Martin Fagereng Johansen, Franck Fleurey, Mathieu Acher, Philippe Collet, and Philippe Lahire. ***Exploring the Synergies Between Feature Models and Ontologies***. In International Workshop on Model driven Approaches in Software Product Line Engineering MAPLE 2010/SPLC10 Volume 2, volume 2 of SPLC'10 (Volume 2), pages 163 -171. Lancester University, 2010.

[GW10] Jianmei Guo and Yinglin Wang. ***Towards consistent evolution of feature models***. In Proceedings of the 14th international conference on Software product lines going beyond, SPLC'10, pages 451-455. Springer-Verlag, 2010.

[KMPY06] Ronny Kolb, Dirk Muthig, Thomas Patzke, and Kazuyuki Yamauchi. ***Refactoring a legacy component for reuse in a software product line: a case study***. Journal of Software Maintenance and Evolution Research and Practice, 18(2):109 -132, 2006.

[KKL+98] K. 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(1): 143–168, 1998.

[MPDBF12] Sébastien Mosser, Carlos Parra, Laurence Duchien, and Mireille Blay-Fornarino. ***Using Domain Features to Handle Feature Interactions***. In Sixth International Workshop on Variability Modelling of Software-intensive Systems(VAMOS'12), workshop, Leipzig, Germany, 2012.
[P11] Carlos Parra, **Towards Dynamic Software Product Lines: Unifying Design and Runtime Adaptations.** Université des Sciences et Technologie de Lille - Lille I, Mar. 2011.
[PBL05] Klaus Paul, Gunter Bockle, and Franck J Linden. ***Software product Line Engineering : Foundation, Principles and Techniques***. Springer, hardcover edition, 2005.
[PCBD10] Carlos Parra, Anthony Cleve, Xavier Blanc, and Laurence Duchien. ***Feature-based Composition of Software Architectures***. Software Architecture, pages 230 -245, 2010.
[SS09] Horst Schirmeier and Olaf Spinczyk. ***Challenges in Software Product Line Composition***. Constraints, 0:1-7, 2009.
[ZSS+10] Ste_en Zschaler, Pablo Sanchez, Joao Santos, Mauricio Alferez, Awais Rashid, Lidia Fuentes, Ana Moreira, Joao Araujo, and Uira Kulesza. ***VML* - A Family of Languages for Variability Management in Software Product Lines.*** Engineering, pages 82 -102, 2010.