Aspect-Oriented Analysis for Software Product Lines Requirements Engineering

Patrícia Varela¹, João Araújo¹, Isabel Brito², Ana Moreira¹

¹ CITI, Departamento de Informática, Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa, 2829-516, Caparica, Portugal
patixinha@gmail.com, {ja, amm}@di.fct.unl.pt
² Instituto Politécnico de Beja, 7800-050, Beja, Portugal
issb@estig.ipbeja.pt

Abstract. When building feature models for Software Product Lines at requirements level other models are needed to help identifying, describing, and specifying features. Traditional approaches usually perform this manually and, in general, the identification and modularization of crosscutting features, which is crucial to obtain a more modularized model, is ignored, or not handled systematically. This hinders requirements change. We propose an aspect-oriented approach for SPL enriched to automatically derive feature models where crosscutting features are identified and modularized using aspect-oriented concepts and techniques. This is achieved by extending and adapting the AORA (Aspect-Oriented Requirements Analysis) approach. AORA provides templates to specify and organize requirements based on concerns and responsibilities. A set of heuristics is defined to help identifying features and their dependencies in a product line from AORA templates. A tool was developed to automatically generate the feature model from AORA templates.

Keywords: Aspect-Oriented Requirements Analysis, Software Product Lines.

1 Introduction

Requirements Engineering includes the identification, analysis, documentation, validation and management of requirements [15], where requirements describe functionalities, constraints or quality attributes in software systems. Our focus is on creating a synergy between Software Product Lines Engineering (SPLE) and Requirements Engineering benefiting from their concepts, techniques and tools. Whilst Requirements Engineering techniques can be used to elicit and specify domain and application requirements, SPLE captures commonalities and variabilities of product families promoting reuse [16]. Thus, in a medium term, productivity can be increased.

One of the most used Software Product Lines (SPL) techniques to specify requirements and handle their commonalities and variations is feature modeling [8][14]. A common limitation is that feature models do not provide enough information on each feature, such as their behavior and a rationale for their
dependencies, needing other models to provide that information. Another difficulty is dealing with the crosscutting nature of (parts of) some features. Aspect-oriented (AO) techniques [2] have been used successfully to model crosscutting concerns. In the context of this paper a concern refers to any matter of interest of one or more stakeholders [11], and when referring to SPL a feature can be seen as a concern too. Thirdly, a final limitation is the lack of tools to automatically derive feature models from requirements specifications be AO or not.

The application of requirements engineering techniques, such as use cases [12], viewpoints [9] and goals [6], has improved SPLE specifications [3][7][10][13][18][20]. This resulted in documents that provide models expressing different perspectives of the requirements [16], therefore complementing and giving a more complete view of the requirements specifications. However, requirements elicitation and analysis for SPL could be enhanced if the modularization of crosscutting concerns were addressed using aspect-oriented techniques [2]. Moreover, all the information gathered in AO requirements specifications for SPL should contain enough information to be able to derive automatically the corresponding feature models.

Aspect-Oriented Requirements Engineering (AORE)\(^1\) appeared to address this problem by identifying, representing, specifying and composing crosscutting concerns; these are encapsulated in separate modules (known as aspects) [2][17]. One of the pioneering AORE approaches is AORA (Aspect-Oriented Requirements Analysis) [5]. AORA offers some advantages with respect to other existing ones, in particular: a detailed template specification for all types of concerns (be them functional, non-functional or crosscutting); a set of concepts and techniques rigorously defined in a metamodel; a set of composition operators to study the impact of a set of concerns over the base ones; an efficient and rigorous conflict resolution method; and, finally, a supporting tool developed based on the defined metamodel.

We extend and adapt AORA to support SPL development both at domain and application engineering levels. The result is the PLAORA approach (Product Lines for Aspect-Oriented Requirements Analysis). PLAORA provides a sound set of heuristics to derive feature models taking into consideration the identification of crosscutting concerns at domain and application engineering levels, offers a tool to systematically and automatically identify and generate automatically common and variable features, and uses Analytic Hierarchy Process (AHP) [19], multi-criteria based method, to identify and resolve conflicts at a more abstract level.

In summary, the aim of this paper is twofold: (i) to enrich the development of SPL with the capabilities and advantages of AORA, where the specification of concerns facilitates the automatic derivation of a feature model and (ii) to make AORA, and AORE in general, also useful for the development of SPLs.

This paper is structured as follows. Section 2 summarizes AORA main concepts. Section 3 describes PLAORA, the new approach, and its development process. Section 4 applies PLAORA to a case study. Section 5 discusses the evaluation of the approach, in terms of both an industrial case study it has been applied to and an experiment performed with a group of ten master’s students. Section 6 presents some

\(^1\) The interested reader can refer to http://www.aosd-europe.net/deliverables/d11.pdf for a survey on AORE approaches, or to the Early Aspects portal (www.early-aspects.net).
of the related work, discusses the added value of our contribution and uses a set of criteria to compare PLAORA with other existing approaches in the field. Finally, Section 7 presents conclusions and future work.

2 Background

AORA [5] is composed of three main tasks: identify concerns, specify concerns and compose concerns. These tasks are accomplished iteratively and incrementally. Identify concerns aims at discovering the concerns of a system, where a concern is as a set of coherent requirements defining a property that the future system must provide. This is performed by analyzing the initial requirements, transcripts of stakeholders’ interviews, etc.. Good sources for concern identification are the existing catalogues, such as the NFR catalogue [6]. Specify concerns provides textual and visual representations of concerns and their relationships. All the useful information about a concern is collected in a template (Tables 1 and 2 are examples). Finally, Compose concerns offers the possibility to compose a set of concerns, incrementally, until the whole system is obtained (if we need that) and, at the same time, identify the impact of a set of concerns on a given base. A composition occurs in a match point which lists the crosscutting concerns that should be composed with each (set of) base concern, forming a composition rule. A composition rule is formed of concerns and pre-defined operators.

3 The PLAORA Approach

Being in tune with SPL, PLAORA also distinguishes between Domain Engineering and Application Engineering. The product line is created at the Domain Engineering level (according to the process depicted in the top part of Fig.1), and then a product is configured at the Application Engineering level (bottom part of Fig.1).

The Domain Engineering part of the process initially consists of the identification and specification of system concerns and the conflict identification and resolution (step 1 in Fig. 1). The specification of concerns can lead us to identify new concerns and refine others previously identified. Next, the specification templates are described and then the composition of concerns is realized. Here conflict identification and resolution is carried out using a multicriteria method as described in [4][5] where concerns are ranked according to their importance to different stakeholders. We will not focus on this, as this is not the contribution of this paper.

Having all the concerns specified we can identify the features of the SPL whose result is an initial feature model (step 2 in Fig. 1). These are extracted from AORA templates with the help of a set of heuristics. So features are derived from concern templates, with their lists of responsibilities. The resulting feature model is then refined with a complementary set of heuristics to modularize the feature model and identify dependencies between features (step 3 in Fig. 1). That is, once the extraction of possible features is completed, we identify the variability of the SPL and the
different kinds of dependencies between features taking into account crosscutting concerns. The heuristics are described and exemplified in Section 4.

Validation must be performed in parallel with the process just described. In particular, stakeholders need to help validating (i) the identified concerns and respective specifications, as well as (2) guaranteeing that the feature model meets their needs.

![Diagram](image)

**Fig. 1.** PLAORA Domain and Application Engineering process.

At the end of domain engineering process, we have concern templates and a feature model representing common and variable requirements in the SPL. These templates and feature models are analyzed and configured for a particular product of the SPL by the Application Engineering process. Conflicts particular to a specific configuration should be resolved here. Again we do not discuss this here as it is out of scope of this paper.

### 4 Applying PLAORA and describing the developed heuristics

PLAORA has been applied to two case studies, the Mobile Phone and the Smart Home case studies. The Smart Home case study was developed in the AMPLE project [1] and is not described here due to space constraints.

2 The complete specification of the application of PLAORA to the Mobile Phone case study can be found in [link](http://ctp.di.fct.unl.pt/~ja/MobilePhone_CaseStudy.pdf).

3 The complete specification of the application of PLAORA to the Smart Home case study can be found in [link](http://ctp.di.fct.unl.pt/~ja/SmartHome_CaseStudy.pdf).
The Mobile Phone case study is used to illustrate our approach. The example’s aim is to develop software components to make and answer calls, put phone calls on hold, insert contacts in a contacts list, send and receive e-mails, SMS and MMS, take pictures, set alarm and transfer data between two mobile phones. Payments can be performed by ATMs or banks’ websites.

4.1 Domain Engineering

For the activities of Domain Engineering process presented in Fig.1, we will focus on the major contribution of our approach: building the initial and final feature model of an SPL. Due to lack of space, let us assume that the modeling system’s concerns was already performed and a list of concern templates provided. The functional concerns for our problem are: Make call, Answer call, Put phone call on hold, Enable voice mail, Receive MMS/SMS/E-mail and Send MMS/SMS/E-mail. Also the non-functional concerns are: Response Time, Usability, Correction, Confidentiality, Availability, Integrity and Security. Two AORA template examples, one functional and another non-functional, are shown in Table 1 and Table 2, respectively.

<table>
<thead>
<tr>
<th>Table 1. Template for “Make call” concern.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong></td>
</tr>
<tr>
<td><strong>Sources</strong></td>
</tr>
<tr>
<td><strong>Stakeholders</strong></td>
</tr>
<tr>
<td><strong>Description</strong></td>
</tr>
<tr>
<td><strong>Decomposition</strong></td>
</tr>
<tr>
<td><strong>Classification</strong></td>
</tr>
</tbody>
</table>
| **List of responsibilities** | 1. The call is redirected by Mobile Phone Operator  
2. Play signal (at least one of the alternatives, sound or vibration)  
3. The screen displays the phone number  
4. Push button (only one of the alternatives: accept or reject)  
5. The call duration appears on the display  
6. Choose loud voice mode, if desired  
7. The call is disconnected, after finishing the conversation |
| **List of contributions** | <None> |
| **List of priorities** | 1. Usability |
| **List of required concerns** | |

<table>
<thead>
<tr>
<th>Table 2. Template for “Response Time” concern.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong></td>
</tr>
<tr>
<td><strong>Sources</strong></td>
</tr>
<tr>
<td><strong>Stakeholders</strong></td>
</tr>
<tr>
<td><strong>Description</strong></td>
</tr>
<tr>
<td><strong>Decomposition</strong></td>
</tr>
<tr>
<td><strong>Classification</strong></td>
</tr>
<tr>
<td><strong>List of responsibilities</strong></td>
</tr>
</tbody>
</table>
1. The system reacts in time to establish the call
2. The system reacts in time to check if the time of holding the call reached the limit
3. The system reacts in time to capture images
4. The system reacts in time to alert if SMS/MMS/e-mail were successfully sent
5. The system reacts in time to alert if SMS/MMS/e-mail were received
6. The system reacts in time to search for devices within a range of the phone
7. The system reacts to enable the voice mail

<table>
<thead>
<tr>
<th>List of contributions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Availability contributes negatively (-) to Response Time</td>
</tr>
<tr>
<td>2. Correction contributes negatively (-) to Response Time</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>List of priorities</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. User: Very Important</td>
</tr>
<tr>
<td>2. Mobile Phone Operator: Very Important</td>
</tr>
<tr>
<td>3. Banking System: Very Important</td>
</tr>
<tr>
<td>4. Sender/Receiver: Very Important</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>List of required concerns</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;None&gt;</td>
</tr>
</tbody>
</table>

Heuristics H1-H6 identify the features of the system from the AORA templates. Initially, we assume that all features are mandatory. Heuristics H7-H12 produce a feature model and identify variability.

**H1. Identify the root feature based on “sources” entry:** Analyze the “Sources” line to get the root feature of the feature model. For example, based on source “Knowledge of the mobile phones system”, we get the feature “Mobile Phones”. Basically, the root name will be the name of the system.

**H2. Identify features based on the concerns’ name:** Analyze the “Name” line and obtain the system features through these names, i.e., each concern originates a feature. For example, “Make call”, “Answer call”, “Response Time”, “Security” originate features with the same name. To improve readability of the model we proposed a change in the notation on the feature model: features resulting from non-functional concerns are represented by a rectangle with rounded corners, while those from functional concerns are represented by rectangles.

**H3. Identify features that can be grouped based on concerns’ names:** Analyze the “Name” line and make two types of feature groups. (1) Concerns beginning with the same verb, but different objects define a group where the parent feature is composed of the common verb in features plus a generic noun (Verb + Noun). This noun should be generic in order to specify the information common to the sub-feature that you get. For example, considering features “Send SMS”, “Send E-mail”, “Send MMS”, we obtain a parent feature named “Send data”, as data is a more generic noun that can be specialized as SMS, MMS or E-mail. As sub-features we have the original “Send SMS”, “Send Email” and “Send MMS” features. (2) The same object refers to different verbs. In this case we define a generic verb and use the object that occurs repeatedly originating the parent feature. For example, features “Make call”, “Answer Call” and “Put phone call on hold” share the word “call”, originating a group where we can define “Processing call” as a parent feature.

**H4. Identify features based on concerns’ “decomposition”:** Analyze the “Decomposition” line and, in the case of refinement, the refined concerns are sub-features and the concern that was refined is the parent feature. For example, features
“Integrity” and “Confidentiality” are subfeatures of “Security”. This refinement is based on the catalogue for security offered in [6].

**H5. Identify features based on the “list of responsibilities”:** Analyze “List of Responsibilities” and choose those starting with a “Verb + Noun” and which play an important role in the system; these may originate new features (or sub-features). For example, “Make call” has sub-features “Dial number”, “Push the call button”, “Choose a loud voice”, taken from responsibilities “Dial number desired”, “Push the call button to start call” and “Choose loud voice to the call if desired”. Features extracted using this heuristic requires this heuristic requires the user intervention to interact with the system. Also, using “List of Responsibilities” check for additional information to be defined as features to represent in the model, like types of information or ways to achieve functionality. For example, “Answer call” has the responsibility “Push button (only one of the alternatives, accept call or reject call)” gives us the features “accept call”, “reject call”.

**H6. Identify features based on the NFR catalog:** Using existing catalogues, such as [9], we can identify new features for the non-functional requirements (NFRs). These features will be added to the feature model.

**H7. Identify variability from concern’s description:** The “Description” line identifies an optional feature if a modal verb expresses non-obligatory, such as if “can” or “may” appear in the description. This variability is related to the features extracted from H2. For example, “Put phone call on hold” has the description “The user can place a particular call waiting”, which includes “can”. Hence, “Put phone call on hold” is optional.

**H8. Identify variability for other features of the model:** Analyze in “List of Responsibilities” if responsibilities therein have expressions such as “if desired”, “if wanted”, “if possible”, for example; these are optional. The concern “Make call” has the responsibility “Choose loud voice to the call mode, if desired”; therefore, the feature “Choose loud voice” obtained by H5 is classified as optional.

**H9. Identify xor alternatives:** Analyze “List of Responsibilities” using expressions like “only one of the alternatives”; these are xor alternatives. For example, the concern “Answer call” has a responsibility “Push button (only one of the alternatives, accept call or reject call)”. Therefore, the features “Accept call” and “Reject call” are sub-features of the feature “Push button” providing a xor alternative in the model.

**H10. Identify or alternatives:** Analyze in “List of Responsibilities” expressions such as “at least one of the alternatives”; these are identified as or alternatives. For example, in “Answer call” template, the responsibility “Play signal (at least one of the alternatives, sound or vibration) on your phone”, allows the identification of the features “Sound” and “Vibration”. These are alternative sub-features of the feature “Play signal” providing or alternative in the feature model.

**H11. Identify requires dependencies relationships in feature model:** “List of required concerns” in the template originates requires dependencies relationships, represented in the model by dashed arrows. For example, in the “Answer call” template, the required concern “Usability” originates a require dependency relationship in the feature model. One feature that has more than one arrow, pointing to itself, is identified as a crosscutting feature.
H12. Identify excludes dependencies relationships in feature model: Those responsibilities in “List of Responsibilities” that include expressions like “excluding the possibility of X” originate an excludes dependency relationship, where X identifies the other feature present in the link. For example, the responsibility “The service is active, excluding the possibility of putting phone call on hold” in “Enable voice mail” template originates an excludes dependency relationship in the model between the features “Enable voice mail” and “Put phone call on hold”.

The 12 heuristics applied to our case study originated the feature model in Fig.2, where variability is identified. For simplification purposes we represented only requires dependencies relationships for the features “Make Call” and “Answer call” as an example of H11. To reduce the complexity of the feature model with respect to the requires dependencies relationships, we added a small rectangle labeled “requires” under the features (Fig.3) that require others. The list of numbers after “requires” corresponds to the numbers of the required features. This numbering is done from left to right on the model, numbering only the features that were extracted from the names of concerns (H2), as these are required by other concerns.

Fig. 2. Feature model: a version derived from the application of the 12 heuristics presented.

H11 can identify crosscutting features, those that are required by at least another feature. Once we have specified and modularized the SPL features following AO principles, the crosscutting features emerge automatically: they are those represented more than once over the rectangles with a label “requires”. Fig.3 identifies “Usability”, “Response Time” and “Availability” as crosscutting features (these have a black triangle placed at the bottom right hand corner of features, as shown in Fig.3). An example of a crosscutting functional feature is “Mobile phone payment” specified in the URL provided.

4.2 Application Engineering

AORA and the heuristics were used to capture the domain engineering features. Now we can choose different configurations, each one representing a different product of the family. Both, the feature model and the AORA templates, are configured for a
specific product. First it is configured the feature model and then the AORA templates. As an example we want a mobile phone application with the following functionalities: make/answer calls and send/receive SMS and MMS. Fig. 4 illustrates the feature model of the configured application. Response Time and Usability are crosscutting features, recognized with a black triangle placed at the bottom right hand corner of the features.

Fig. 3. Feature model - final version.

Fig. 4. Feature model to the application.

The changes in the templates are done at the level of responsibilities and description of the concerns since these entries in templates are those used to obtain the system’s variability. A concrete application does not include “optional” features, or “alternatives”, “or” and “xor”. Due to lack of space the configuration of the AORA templates are not presented, but they can be found in the URL provided².

4.3 Tool support

The AORA tool specifies and composes concerns, keeping a repository with all the identified elements and relationships. This tool was extended to generate the feature model. This extension implements the 12 heuristics offered by PLAORA. A snapshot of the tool is presented in Fig. 5, show a feature model. By clicking on the yellow button (marked with circle “1” in the red rectangle on the left hand side of figure), the feature model is generated automatically (window on the right).
Tray diamond, marked with circle “2” in Fig. 5, represents the root of the model, and the middle of the image shows the mobile phone terms. The features of the system are thus connected by the links “optional”, “mandatory”, “or”, “xor” and “excludes”. By selecting the concern and clicking on the button marked with a black triangle, the user can visualize the list of concerns that the selected concern cuts across, as shown in small window in Fig. 5 marked with circle “3”. The different colors for elements in this diagram indicate the crosscutting features: red elements represent crosscutting features and blue elements represent non-crosscutting features.

![Fig. 5. Snapshot of the tool with the feature model.](image)

A parser was implemented for the heuristics to collect information from the repository, which is necessary to extract features of the system. For example, at the responsibilities level, if the first two words that compose the responsibility are a “verb + noun”, we obtain the features according to H5. Also, to analyze if the responsibility presents additional information, we need to verify if that information (included between brackets) is useful. The variability analysis was performed by analyzing if the responsibility sentences contained reserved words, such as “if desired/if wanted/if possible”, “only one of the alternatives”, “at least one of the alternatives” and “excluding the possibility of”, which, respectively, originate optional features, xor alternative, or alternative and excludes dependencies relationships (H8-H10 and H12) in the feature model.

NFRs are addressed by H6 (for example, “Response Time”). The tool automatically adds the existing information in the corresponding NFR catalogue. For instance, the feature “Performance” is added and, consequently, its sub-features “Time” and “Space”. Therefore, “Response Time” is a sub-feature of “Time”. Also, if we have the concern “Security”, it will always have as sub-features “Integrity”, “Confidentiality” and “Availability”.
A second parser was implemented to identify optional features (H7) taking into consideration if the description of the concerns included words like “can” or “may”. Obviously, the parsers can be extended to accept other expressions that will help to derive the features, their kinds, and their relationships.

5 Evaluation

PLAORA was evaluated in three ways: (1) based both on case studies, in particular, Smart Home\(^3\) and Health Watcher\(^4\); (2) based on a questionnaire\(^5\) answered by 10 MSc students; (3) comparison to other approaches (Section 6.2).

The Smart Home case study helped us find several situations needing improvement. For example, H8 was extended to consider as variability the information provided in the list of responsibilities, which is contained in brackets without reserved expressions. These reserved expressions are “if desired” (in the original H8), “only one of the alternatives” (H9), and “at least one of the alternatives” (H10), corresponding features will be derived in the model. The new heuristic is illustrated in the “Configure security control system” concern\(^3\), where we have the responsibility “Simulate presence (define rooms, insert date, set initial time, set end time, set duration, set frequency)” where the features “define room”, “insert date”, “set initial time”, “set end time”, “set duration” and “set frequency” are defined as mandatory and also the sub-features of “Simulate presence”.

Another issue is the list of numbers after “requires”, as shown in Fig. 4. In the case of requirements change, all the numbering must be redone. This problem can be solved by making the tool capable of reflecting the impact of the change in the model.

The Health Watcher was also used to validate the approach, where a PLAORA specification was given to a set of 10 Master’s students with knowledge on SPL and AORE. The students were asked to build a feature model based on given specification and then to compare their feature model with the one generated by the tool. At the end they were asked to answer a questionnaire whose questions involved the identification of features, the contribution of templates to identify features and to create the feature model, comparison between the model generated and the model drawn by hand, the advantage of the implemented tool views, the advantage of representing aspects, and the advantage of representing the requires dependency relationships modularly.

The results obtained have the following positive points: (i) existence of the functional and non-functional views on the feature model; (ii) ability to expand and collapse features, reducing the complexity of the models; (iii) identify crosscutting features and “requires” dependencies relationships modularization. The negative points are: (i) representation of the syntax of the model to be unabbreviated; and (ii) lack of a legend to help the perception of the various features represented in the model.

Some suggestions for improvement were presented: (i) transform the abstract syntax of models into a standard one; (ii) add in the models a different symbology for those features that have the ability to add/remove sub-features; (iii) add a descriptive

---

\(^3\) The Health Watcher case study can be found in http://ctp.di.fc.unl.pt/~ja/HW_CaseStudy.pdf.

\(^4\) The questionnaire can be found in http://ctp.di.fc.unl.pt/~ja/Questionnaire.pdf.
label in the buttons to add and remove sub-features of a given feature previously selected; (iv) provide a legend to facilitate the understanding of the models. These suggestions were later implemented to improve the tool functionality.

6 Related work

SPL and the feature model have been widely used and integrated to RE approaches, giving rise to new approaches [18][13][3]. Next we present some of these aspect-oriented approaches integrated with SPL.

6.1 Related aspect-oriented approaches integrated with SPL

Silva et al. present an approach [18] to show that i* extended with aspects can support variability for SPL. The selection of specific features and its decomposition to an individual product is facilitated by the use of aspects in this approach. Heuristics are presented to map the aspectual i* model to the feature model. However, the approach needs to be improved to manage models’ scalability. The approach considers that each feature, optional or alternative is mapped into one aspect and this is not always the case. With limited capacity to manage the complexity of the model, the representation of all the variability is difficult to understand and there is a lack of mechanisms for modularization and composition. Our proposal tries to cover these limitations and also offers tool support.

Jayaraman et al. present an approach [13] aiming at maintaining the separation of features during the modeling of systems based on UML models. It also detects unwanted structural interactions between the different types of features. Also, the basic features are expressed in terms of class diagrams, sequence diagrams and state diagrams in UML, while variable features are specified in UMLT (UML Transformation), which is a UML representation of transformations of graphs. This description corresponds to the MATA technique [21], where the features are built automatically, using a mechanism for rewriting graphs, where a critical pair analysis is used to detect interactions of structural features. The user can choose a subset of features available and automatically generate a product for this set of features. Our proposal differs from theirs, since we provide a set of heuristics to derive a feature model and they do not specify a separate feature model.

Bonifácio and Borba present an approach [3] whose main objective is to characterize the management of variability, as a crosscutting concern. The specification of concerns variability is done separately. It suggests a framework for modeling the process of composition of variability in scenarios. This framework provides a basis to describe variability as aspects mechanisms, differently from existing approaches, since it considers the contribution of different input languages. It presents the specification of three forms of variability for use case scenarios, such as, variability in function, variability in data and variability in flow control. The approach improves the separation of concerns between variability management and scenario specifications, dealing with the variability of scenarios as the composition of different artifacts. Variability management is based on weaving processes that have as input the
mentioned artifacts that crosscut each other, with respect to the result of a specific product that uses use case models. Our proposal differs from theirs as we offer a set of heuristics to identify features, create feature models and help identifying variability.

6.2 A comparative study

The aspectual SPL approaches described in Section 5.1 are now compared with PLAORA. The set of comparison criteria is taken from [1]: Conflict resolution (conflicts are inevitable and can arise between the requirements, functional or non-functional); Heuristics (this is a set of steps aimed at facilitating access to new theoretical developments or discoveries, in our case to discover features); Tool support (the approach presents a support tool, for requirements management in support of its architecture, traceability, or its evolution); Modeling requirements (activities to capture the functional requirements, of a product line and their dependencies on each other); Modeling features (consist of activities to identify, study and describe the features relevant to a given domain); Modeling scenarios (include not only the functionality of systems and their interactions with users, but also aspects); Composition (analyzes the composition in the approach) and Feature interaction (occurs when the integration of two features would modify the behavior of one or both). Table 3 summarizes the results of the comparison.

Table 3. Comparison between AO approaches integrated in SPL.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Conflict resolution</td>
<td>Offers rigorous decision support system to identify (using contribution and priorities information) and solve conflicts (using AHP) at a more abstract level.</td>
<td>Can be extended to support the negative contribution relationship supported by [6].</td>
<td>Uses pair-analysis for identifying conflicts in more detailed analysis models (e.g. sequence diagrams).</td>
<td>No</td>
</tr>
<tr>
<td>Heuristics</td>
<td>For domain and application engineering, as well as for identifying crosscutting concerns.</td>
<td>Only used to reduce the model complexity &amp; identification of crosscutting.</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Tool support</td>
<td>Models composition, variabilities with feature models, product config., config. knowledge and conflict detection. It maintains a repository of elements &amp; relationships, where all the information is kept according to AORA templates.</td>
<td>No</td>
<td>Allows automated composition of UML models of features and detection of some kinds of feature interactions.</td>
<td>Models composition of scenario variabilities with feature models, product configuration, and configuration knowledge.</td>
</tr>
</tbody>
</table>
Captures commonality & variability.

Captures commonality & variability.

Captures commonality & variability.

Captures commonality & variability.

Uses UML sequence diagrams and use cases (from original AORA).

No

Uses UML sequence diagrams.

Provides use cases.

Identify requires and excludes dependencies relationships in feature model

No

Feature interactions can be verified for consistency with the relations captured in the feature dependency diagram

No

Composition is built from simpler rules using brackets, "\(\)" and "\[\]" for allocating priorities to the operators: Enable \(\Rightarrow\) Disable \([\Rightarrow\) Parallel \("\)" and Choice "\[\)"

Allows composition trying to reduce the complexity of the models \(\ast\).

Supports composition for UML class, sequence and state diagrams using graph transformations (all composition mechanisms are from original MATA).

Deals with scenario variability as a composition of different artifacts: SPL use case \&, feature models, product configuration, \& configuration knowledge.

In summary, according to Table 3 our approach has the following advantages: it provides a sound set of heuristics to derive a feature model that takes into account the identification of crosscutting concerns at domain and application engineering level; a tool that offers a systematic and automatic way to identify common and variable features and a multi-criteria based method (AHP) to identify and resolve conflicts at a more abstract level.

7 Conclusion and future work

PLAORA is an aspect-oriented approach that supports elicitation and analysis of requirements for SPL at domain and application engineering levels. It offers a set of heuristics to automatically derive feature model from aspect-oriented requirements descriptions. This is done automatically by the extension performed on the AORA tool. Aspect-orientation mechanisms were very useful in the definition of PLAORA to identify crosscutting features and consequently obtain a more modularized feature model. It brings to the community several advantages, as the comparison Table 3 shows.

As future work we need to work on the scalability of the model. We are planning to implement two different views of the system (functional and non-functional) to partially achieve this. Our final goal is to use lexical analysis and text mining to ultimately interpret the text offered by the AORA templates to extract initial the initial feature model. The resulting approach needs to be then applied to real case studies.
Acknowledgements

This work was partially funded by the European AMPLE project and FCT MCTES.

References