A Semi-Automatic Strategy to Identify Crosscutting Concerns in PL-AOVgraph Requirement Models

Maíra Medeiros[1], Lyrene Silva[1], Ana Luisa Medeiros[1]

[1] DIMAp – Departamento de Informática e Matemática Aplicada
UFRN – Universidade Federal do Rio Grande do Norte
Campus Universitário Lagoa Nova, Natal, RN, Brasil.

{mairafbmedeiros, analuisafdm}@gmail.com,
lyrene@dimap.ufrn.br

Abstract. The Requirements Engineering area faces problems because the requirements are often ambiguous, incomplete or confusing. These points are commonly obscured by the natural language, which abstracts the complexity of interactions among requirements. However, these interactions need to be analyzed because they influence on how the software development life-cycle activities can be modularized from a well formulated and concise requirements description. In this context, PL-AOVgraph is an aspect-oriented requirement modeling language, which offers support to represent relationships among concerns and provides separation of crosscutting concerns. However, in order to identify these crosscutting concerns in PL-AOVgraph, there are only some heuristics, which help analysts to perform this activity manually. Therefore, this paper proposes a semi-automatic strategy to identify crosscutting concerns in PL-AOVgraph models. This strategy is based on analysis of an adjacency matrix composed of the relationships among requirements. In order to evaluate this strategy, a case study is applied.

Keywords: Crosscutting concerns, crosscutting concern identification, PL-AOVgraph, ReqSys-MDD tool.

1 Introduction

The requirements engineering activity is responsible for discovering, documenting and maintaining software requirements [17]. Some of the artifacts produced during these activities are written in natural language. Hence, the requirements engineering faces problems due to the fact that the requirements are often ambiguous, incomplete and confusing. These problems happen because there are many ways to describe the same requirement using natural language, and also, it abstracts the complexity of interactions among requirements.

One way to deal with these problems is by using the separation of concerns. It decomposes software systems into smaller modular units, each one is related to one concern. In this context, the Aspect-Oriented Software Development (AOSD) [7] is particularly interesting because the modularization is achieved by encapsulation of crosscutting concerns through specific abstractions of language used. Crosscutting concerns are parts of a system which are strongly related, scattered or tangled, influencing or restringing each other, making the system complex and difficult to analyze.

The aspect-oriented requirements engineering area proposes methods and techniques to identify, to separate and to compose crosscutting concerns focusing on requirement artifacts.

In the context of the aspect-oriented requirements engineering, there are some approaches of identification, modeling and analysis of crosscutting concerns. Among these, PL-AOVgraph (Product Line – Aspect-Oriented Vgraph) is a goal-oriented requirement modeling language which supports the description, composition and visualization of requirements. Regarding PL-AOVgraph models, the crosscutting concern identification is performed in an ad hoc way, based on some general heuristics, such as quantity of input or output relationships (fan-in or fan-out), or the possibility of reusing a particular concern [16], leaving to the analyst the subjective decision to model some requirements as crosscutting concern or not. Additionally, even deciding to model a particular concern as crosscutting, the analyst may do it incompletely, not modularizing such concern in the most appropriate way.

Therefore, the objective of this paper is to present a semi-automatic strategy to identify crosscutting concerns in PL-AOVgraph models and to report a case study which evaluates if this strategy is efficient. This strategy was inserted in ReqSys-MDD tool, which is an Eclipse plug-in to validate requirement specifications (in PL-AOVgraph) and execute automatic bi-direction transformations between PL-AOVgraph models and Features Models [14][15]. The motivation for attaching our strategy to ReqSys-MDD is the reuse of some ReqSys-MDD functionalities, such as the validation of PL-AOVgraph documents and the parsers Xtext and Acceleo, which transform text to model and model to text, respectively. Furthermore, our aim is to integrate into ReqSys-MDD tool all functionalities about PL-AOVgraph.

This paper is organized as follows. Section 2 presents PL-AOVgraph and its features; Section 3 defines our semi-automatic strategy to identify crosscutting concerns and to write the crosscutting relationships. Section 4 specifies how this strategy was inserted in PL-AOVgraph tool. Section 5 describes the results achieved using our approach with Crisis Management System (CMS) case study [9]. Section 6 summarizes some related works. Section 7 presents our final remarks and future works.

2 PL-AOVgraph

PL-AOVgraph [14] is an extension to the AOV-Graph goals model [16] with support to the variability. In other words, PL-AOVgraph is a requirement modeling language which inherits all the AOV-Graph features. These models consist of oriented graphs composed by the following element types: (i) task – functional requirement; (ii) softgoal – non-functional requirement; and, (iii) goal – organizational goal. The relationships among the PL-AOVgraph elements may be one of three types: contribution; correlation; and, crosscutting [12]. Although these models are graphics, we have worked in their textual representation, which is more easily manipulated.

Correlation relationships indicate the influence, whether positive or negative, from a goal to a softgoal (one-to-one relation). This influence is labeled as follows: Make, Break, Help, Hurt or Unknown.

Contribution relationships are represented by the hierarchical relationships (one-to-one) between child and parent elements, respectively, source and target of the relationship. These contributions may be labeled as one of the following types: And, Or, Xor, or Inc-or.

In opposition to contributions and correlations, which are one-to-one relationships, crosscutting relationships represent many-to-one relations. These relationships can modularize many interactions in one relationship, decreasing the quantity of contributions. The amount of contributions must be specified and also it must be determined which concerns are crosscutting each other [14]. The description of the crosscutting relationship is based on elements of Aspect-Oriented Software Development (AOSD) proposed by AspectJ language, so, it is composed of: (i) Source – which is the origin of the relationship, i.e., what concern influences other requirements; (ii) Pointcut – which is the target set of the relationship, i.e., the requirements which are affected by the source; (iii) Advice – which specifies what requirements (child of source) are scattered or tangled on the pointcuts; (iv) Intertype Declaration – which defines new instances (if it is element type) or types of elements (if it is attribute type) to the model.

Additionally, PL-AOVgraph allows the insertion of new properties to a model by using “property” elements. There are six pre-defined properties to support the variability and the transformation between feature models and PL-AOVgraph [14]. However, these properties are not used in this work.

We use the Crisis Management System (CMS) [9] like a demonstrative example in next sections. This system was defined with purpose to create a common case study for aspect-oriented modeling community and it was presented like special issue in the Transactions on Aspect-Oriented Software Development journal. The CMS domain helps in the identification, evaluation to handle a crisis situation, allowing the communication among all parties involved.

Figures 1 and 2 illustrate a small part of the CMS PL-AOVgraph model. Figure 1 illustrates its graphical representation and Figure 2 is its textual representation. In this example, the task “Manage [Communication]” (line 5) contributes to two tasks (lines 5 and 10) and four softgoals (lines 15, 20, 25 and 30). These contributions are showed by the “and” pointers between the nodes shown in Figure 1, and included in the labels in parenthesis in Figure 2.

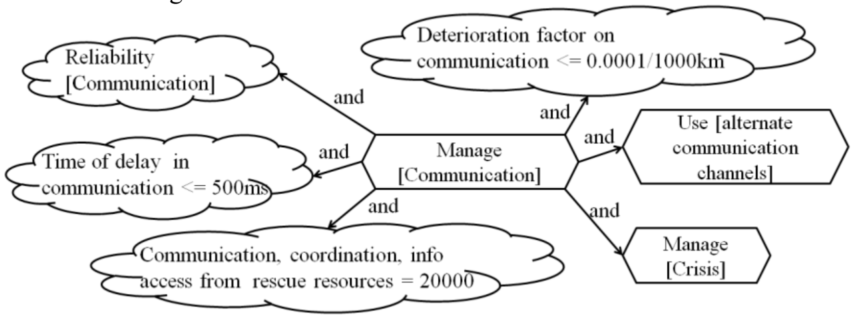

Reliability [Communication]
Deterioration factor on communication <= 0.0001/1000km
and
and
and
Use [alternate communication channels]
and
Manage [Communication]
Time of delay in communication <= 500ms
and
and
Communication, coordination, info access from rescue resources = 20000
Manage [Crisis]

Fig. 1. Contribution relationships in PL-AOVgraph graphic mode.

```
01 aspect_oriented_model {
02   goal_model "Crisis Management" {
03     goal "Crisis resolved" (and) {
04       task "Manage [Crisis]" (and) {...
05         task "Manage [Communication]" (and) {}
06       }
07     }
08     softgoal "Security" (and) {...
09       task "Use [alternate communication channels]" (and) {
10         task_ref "Manage [Communication]" (and) {}
11       }
12     }
13     softgoal "Reliability" (and) {...
14       softgoal "Reliability [Communication]" (and) {
15         task_ref "Manage [Communication]" (and) {}
16       }
17     }
18     softgoal "Multi-Access" (and) {...
19       softgoal "Communication, coordination, info access from rescue resources = 20000" (and) {
20         task_ref "Manage [Communication]" (and) {}
21       }
22     }
23     softgoal "Real-time" (and) {...
24       softgoal "Time of delay in communication <= 500ms" (and) {
25         task_ref "Manage [Communication]" (and) {}
26       }
27     }
28     softgoal "Accuracy" (and) {...
29       softgoal "Deterioration factor on communication <= 0.0001/1000km" (and) {
30         task_ref "Manage [Communication]" (and) {}
31       }
32     }
33   }
34 }
```

Fig. 2. Contribution relationships in PL-AOVgraph textual mode.

As requirement models are usually extensive, insofar as the model is evolving and growing it becomes difficult to maintain its readability as well as its understanding. Therefore, the crosscutting relationships are a strategy to separate concerns, reduce the number of contribution relationships and tangled and scattered elements. Figure 3 shows a crosscutting relationship example, which replaces those relationships represented by task references (task_ref) showed in Figure 2(lines 10, 15, 20, 25 and 30).

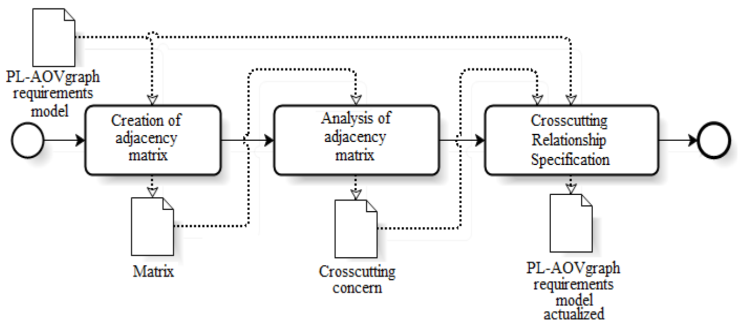
Figure 3, line 2, represents the source of this crosscutting relationship, it is the goal “Crisis resolved”, because it is the parent of the elements which are scattered or tangled. The pointcut block (lines 03 to 08) defines the elements which are affected by the advice: the two tasks, “Use [alternate communication channels]” and “Manage [Crisis]”; and four softgoals “Reliability [Communication]”, “Communication, coordination, info Access from rescue resources = 20000”, “Time of delay in communication <= 500ms” and “Deterioration factor on communication <= 0.0001/1000km”. The advice block (lines 09 a 11) defines the elements which are scattered or tangled with pointcut elements, referenced by “PC1”, in this case, the task “Manage [Communication]” (line 10).

```
crosscutting {
  source: goal_ref "Crisis resolved" {
  pointcut PC1: include  "Use [alternate communication channels]"
                and include "Manage [Crisis]"
                and include "Reliability [Communication]"
                and include "Communication, coordination, info access from rescue resources = 20000"
                and include "Time of delay in communication <= 500ms"
                and include "Deterioration factor on communication <= 0.0001/1000km"
  advice (around): PC1 {
    task_ref "Manage [Communication]" (and) {}
  }
}
```

Fig. 3. Crosscutting relationship in PL-AOVgraph textual model.

3 Crosscutting Concerns Identification in PL-AOVgraph

The identification strategy presented in this paper is based on the fan-out analysis of relationships among requirements. For this analysis, it is used an adjacency matrix to identify and to account these output relationships. In this context, the identification process is comprised of three major steps, as illustrated in Figure 4. A PL-AOVgraph requirements model is the input of this process. From it, a matrix with the relationships among requirements is created. After that, crosscutting concerns may be identified by accounting how many contribution relationships are sourced at each requirement. Hence, crosscutting concerns may be specified, considering the data of the matrix and the data of the input model. Finally, the PL-AOVgraph model is updated by substituting contributions by crosscutting relationships. This process is detailed and exemplified below.



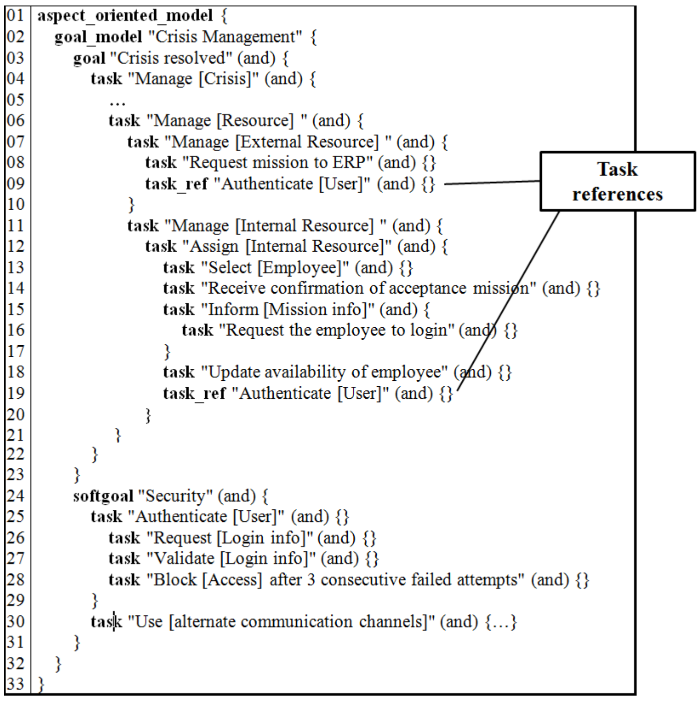
PL-AOVgraph requirements model
Creation of adjacency matrix
Analysis of adjacency matrix
Crosscutting Relationship Specification
Matrix
Crosscutting concern
PL-AOVgraph requirements model actualized

Fig. 4. Process to identify and specify crosscutting relationships.

The first stage is the creation of an adjacency matrix, considering that it shows the relationships among requirements. This activity takes as input a PL-AOVgraph requirement model from that it generates an adjacency matrix. This matrix is built from contribution relationships among the PL-AOVgraph elements. As soon as the PL-AOVgraph model is read by the tool, the matrix is fulfilled in a way that the horizontal lecture indicates the relationships which the element, refered by the matrix lines, originates.

In order to illustrate this activity, Figure 5 shows a small part of the requirement specification in PL-AOVgraph of CMS. In this Figure, there are two major concerns, represented by the goal “Crisis resolved” (line 03) and by the softgoal “Security” (line 24), as well as contribution relationships between them and other (sub) tasks (lines 04 to 22 and lines 25 to 31), “Authenticate [User]” is a task that contributes, besides to the softgoal “Security” (line 24), to the tasks “Manage [External Resource]” and “Manage [Internal Resource]” and it is represented by task_refs elements (lines 09 and 19).

Based on this PL-AOVgraph model, the adjacency matrix is fulfilled, see Table 1, where its horizontal lecture indicates the direction of the contribution relationships, for instance, “Select [Employee]”, “Receive confirmation of acceptance mission” and “Inform [Mission info]” (lines 13 to 15 in Figure 5) contribute to “Assign [Internal Resource]” (line 12 in Figure 5).



```
01 aspect_oriented_model {
02   goal_model "Crisis Management" {
03     goal "Crisis resolved" (and) {
04       task "Manage [Crisis]" (and) {
05         …
06         task "Manage [Resource] " (and) {
07           task "Manage [External Resource] " (and) {
08             task "Request mission to ERP" (and) {}
09             task_ref "Authenticate [User]" (and) {}
10           }
11           task "Manage [Internal Resource] " (and) {
12             task "Assign [Internal Resource]" (and) {
13               task "Select [Employee]" (and) {}
14               task "Receive confirmation of acceptance mission" (and) {}
15               task "Inform [Mission info]" (and) {
16                 task "Request the employee to login" (and) {}
17               }
18               task "Update availability of employee" (and) {}
19               task_ref "Authenticate [User]" (and) {}
20             }
21           }
22         }
23       }
24     softgoal "Security" (and) {
25       task "Authenticate [User]" (and) {}
26         task "Request [Login info]" (and) {}
27         task "Validate [Login info]" (and) {}
28         task "Block [Access] after 3 consecutive failed attempts" (and) {}
29       }
30       task "Use [alternate communication channels]" (and) {…}
31     }
32   }
33 }
```

Fig. 5. Example of PL-AOVgraph textual model

Once the relationships between the requirements are identified and added to the matrix, the second step of the process takes place, which consists on analyzing this matrix to identify crosscutting concerns.

This analysis consists on evaluating the scattered and tangled phenomenon. This phenomenon is identified when a requirement affects (or is affected by) several other requirements. Therefore, how much more a requirement contributes to other, more scattered and tangled it is. Hence, the crosscutting concern identification can be achieved by counting these relationships. For example, using our adjacency matrix (Table 1), we can quickly visualize that “Authenticate [User]” contributes to three other requirements while the others only contributes to one (highlighted in Table 1). It is important to mention that some cells, in which the element did not generate output relationships, were removed to improve the visualization.

It is important to remark that we do not determine a minimum value to the amount of contributions that a requirement must have to be considered a crosscutting concern. We prefer to make this value configurable, once this quantity depends on the context and it varies from one system to another, so, the requirements engineer must set it. Because of this, our strategy is not completely automatic. For example, considering the example of Table 1, if the requirements engineer set this value with three (it can be any number greater than two – the minimum defined by literature), then it can be inferred that the task “Authenticate [User]” is a crosscutting concern, whereas if the requirements engineer set this value with four (or more) then it would not be considered a crosscutting concern in this model, because there is not any element which generates this quantity of relationships.

Table 1. Example of adjacency matrix

$\longrightarrow$	Crise resolved	Manage [Crisis]	Manage [Resource]	Manage [External Resource]	Request mission to ERS	Manage [Internal Resource]	Assign [Internal Resource]	Inform [Mission info]	Security	Authenticate [User]
Manage [Crisis]	x									
Manage [Resource]		x								
Manage [External Resource]			x							
Request mission to ERS				x						
Manage [Internal Resource]			x							
Assign [Internal Resource]					x					
Select [Employee]							x			
Receive confirmation of acceptance mission							x			
Inform [Mission info]							x			
Request the employee to login								x		
Update availability of employee					x					
Authenticate [User]				x		x			x	
Request [Login info]										x
Validate [Login info]										x
Block [Access] after 3 consecutives failed attempts										x
Use [alternate communication channels										x

Once the crosscutting concerns are identified, it is necessary to represent them in crosscutting relationships, it is the third and last step of the process. Therefore, to specify crosscutting relationships, it is indispensable to define the elements that form this kind of relationship: source, pointcut, advice and/or intertype declaration.

It is important to emphasize that if there are crosscutting relationships defined in the input specification, then they need to be updated with other pointcuts and advices.

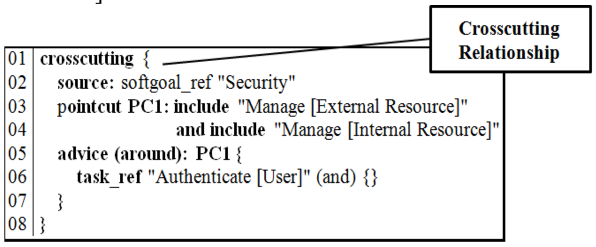
Furthermore, there must never be relationships with the same source otherwise, there will not modularization.

The source is the crosscutting relationship origin that is represented by the parent requirement of the crosscutting concern. The Table 1 illustrates that the task "Authenticate [User]" was identified as crosscutting concern by analysis of matrix. Therefore, to define the source of the relationship, it is necessary to analyze the specification (Figure 5) to identify its parent requirement. Analyzing this specification, we can notice that the softgoal "Security" is the parent requirement, consequently, it is the source of the crosscutting relationship (line 25).

Pointcuts indicate the requirements which are affected by crosscutting concern, then in our example, the tasks "Manage [External Resource]" and "Manage [Internal Resource]" compose the elements of the pointcut. As explained in Section 2, advice and intertype declaration indicate requirements, which affect other requirements. However, the pieces of advice are stated in the model while intertype declarations are not stated in the model. So, this strategy cannot identity intertype declarations. Therefore, in our example, the advice is composed of the task "Authenticate [User]".

It is worth highlighting that since each crosscutting relationship accommodates many contributions, then these contributions are replaced by crosscutting relationships. This reduction and modularization of the relationships aids traceability and consistency management, because each part involves only one feature, thus it makes it easier to locate changes and to deal with one important issue at a time.

Finally, concluding these stages, a new requirement specification is created which removes some contributions and adds crosscutting relationships. The Figure 6 shows the crosscutting relationship (lines 01 a 08) created to represent that "Authenticate [User]" is a crosscutting concern and it crosses to "Manage [External Resource]" and "Manage [Internal Resource]".


Crosscutting Relationship

```
01 crosscutting {
02    source: softgoal_ref "Security"
03    pointcut PC1: include "Manage [External Resource]"
04                  and include "Manage [Internal Resource]"
05    advice (around): PC1 {
06       task_ref "Authenticate [User]" (and) {}
07    }
08 }
```

Fig. 6. Example of Crosscutting Relationship

4 Extending ReqSys-MDD Tool to Identify Crosscutting Concerns

ReqSys-MDD [15] implements a bi-directional mapping between PL-AOVgraph and Features Model, using MDD approach. This tool was coded at Eclipse environment, which offers the Plug-in Developer Environment (PDE) and the Eclipse modeling framework (EMF), both needed for the elaboration of metamodels, and the ATL (ATLAS Transformation Language) Development Tool, needed for implementation of mapping rules.

Furthermore, Xtext and Acceleo plug-ins are utilized and integrated to the EMF in order to transform text into model and model into text, respectively. In ReqSys-MDD, Xtext is responsible for transforming a PL-AOVgraph textual specification (or a Feature Model described in XML), in a XMI model, which is the input to ATL transformation rules. The Acceleo, on the other hand, is responsible for doing the inverse process, that is to transform a XMI model, produced by ATL transformation, in a PL-AOVgraph textual specification (or in a Feature Model) described in XML.

ReqSys-MDD also helps to edit PL-AOVgraph specifications, offering keyword coloring and lexical and syntactic analysis, certifying that input models are conformed to their metamodel.

In this context, the method proposed in this paper was implemented as an additional module to the ReqSys-MDD plug-in. This module was coded using the Java programming language, this choice occurred because ATL did not present support to strategy implementation proposed in this paper.

Figure 7 shows the crosscutting concern identification flow in an additional module of ReqSys-MDD tool: (i) the input PL-AOVgraph textual specification is analyzed by the Xtext module and thus transformed in a XMI model; (ii) from this XMI model, Java objects are created and through analysis of adjacency matrix the crosscutting relationships are identified and written, generating a new XMI model; (iii) this XMI is transformed, through Acceleo, in a PL-AOVgraph textual specification.

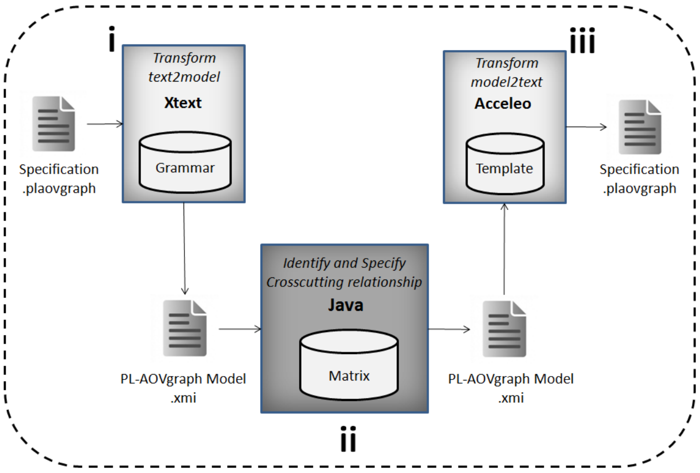


Fig. 7. The crosscutting concern identification flow in ReqSys-MDD.

5 Case Study

We have used in this case study the same system used like a demonstrative example in this paper – the Crisis Management System (CMS). This system was selected because it has many of the PL-AOVgraph elements which are fundamental for an accurate analysis.

As mentioned in this paper, the CMS system aims to help in identifying, assessing, and handling a crisis situation by orchestrating the communication between all parties involved, by allocating and managing resources, and by providing access to relevant crisis-related information for authorized users [9].

The case study reported here aimed to compare the crosscutting concerns obtained from the manual technique and the results from the strategy described in this paper. It was consisted of 3 stages:

1. **Manual crosscutting concerns identification and specification** – this stage is responsible to model a PL-AOVgraph model before and after the composition process, based on [9]. It is important to remind that composition process generates a specification whose crosscutting relationships are disunited in contribution relationships. So, firstly, it is modeled a specification using heuristics to identify crosscutting relationships (model 1), defined in [16]. After that, a version without crosscutting relationships of this model was generated (model 2) (by the composition process explained in [16]) in order to be used as input to the ReqSys-MDD. It is important to remark that these models were not created by the authors of this strategy. However these models were created by a person that was expert in PL-AOVgraph because of this we assumed that these models are correct;
2. **Crosscutting concerns identification using ReqSys-MDD** – the second model cited on stage 1 was used as input in ReqSys-MDD in order to identify crosscutting concerns. It was set that a requirement is scattered or tangled if it affects 3 or more elements. ReqSys-MDD generate another model with crosscutting relationships (model 3); and
3. **Results comparison** – the output generated in the stage 2 (model 3) was analyzed and compared with the first model created in the stage 1. Therefore, we compare the results presented by the manually case study with the result of applying the semi-automatic strategy proposed in this paper.

Table 2. Statistics about CMS Case Study

	Manual case study		Semi-automatic case study
Elements	Quantity of elements - Model 1, stage 1	Quantity of elements - Model 2, stage 1	Quantity of elements – Model 3, stage 2
Goals	6	6	6
Softgoals	27	27	27
Tasks	79	79	79
Contribution relationships	108	156	106
Correlation relationships	20	20	20
Crosscutting relationships	6	0	4
Crosscutting concerns	17	0	12

Table 2, summarizes the results achieved; columns 2 and 3 present the quantity of elements comprised in models created in stage 1, while the last column present the quantity of elements created in the stage 2. The CMS PL-AOVgraph model is comprised of 27 softgoals, 6 goals, 79 tasks and 20 correlations. These elements are the same in all models. On the other hand, the amount of contributions and crosscutting relationships are different in each model: in model 1, there are 108 contributions and 6 crosscutting relationships; in model 2, there are 156 contributions and zero crosscutting relationships; and in model 3, there are 106 contributions and 4 crosscutting relationships.

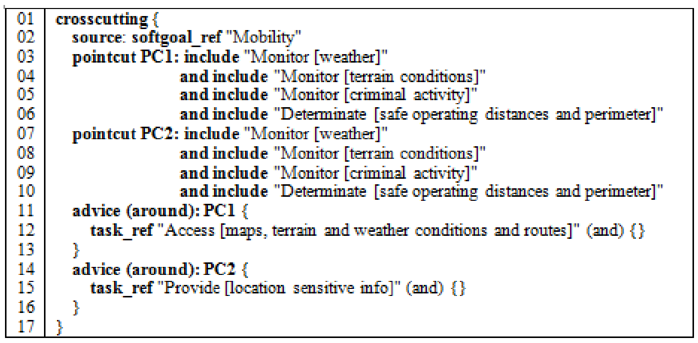
The elements showed like crosscutting concern in Table 2 are the advice elements, i. e., the elements which are repeated along the model.

Comparing the results presented by manual case study and semi-automatic case study (see Table 2), the main difference is the quantity of crosscutting relationships. In the semi-automatic case study were identified 4 crosscutting relationships, while in the manual case study were showed 6 crosscutting relationships; and the quantity of crosscutting concerns, in the semi-automatic case study were identified 12 crosscutting concerns, while in manual case study were identified 17 crosscutting concerns.

Analyzing these relationships, it was noticed that: 5 of the 6 crosscutting relationships manually identified were covered by semi-automatic case study. It could be observed due the level of hierarchy used in both case studies. Hereinafter, it will be explained clearer.

Thus, 2 of them were written in the same manner. However, in one of them, the manual case study presented one additional element of advice than the semi-automatic case study. Figure 8 shows the crosscutting relationship identified by ReqSys-MDD that was equivalent in both case studies, i.e., the source, pointcut and advice elements were equals in both case studies.

In this perspective, 3 of these 5 elements were not written in the same manner, because the semi-automatic strategy defines the source of this relationship with the element which is at the top of the hierarchy and, the manual case study, defines the second level of hierarchy, but they are equivalents. Additionally, these 3 relationships represented 2 relationships in semi-automatic case study, and also the semi-automatic case study identified one more advice element than the manual case study. This element was not identified by manual case study because it was not used only the quantity of relationships, but also the possibility of reuse.



```
crosscutting {
  source: softgoal_ref "Mobility"
  pointcut PC1: include "Monitor [weather]"
                and include "Monitor [terrain conditions]"
                and include "Monitor [criminal activity]"
                and include "Determinate [safe operating distances and perimeter]"
  pointcut PC2: include "Monitor [weather]"
                and include "Monitor [terrain conditions]"
                and include "Monitor [criminal activity]"
                and include "Determinate [safe operating distances and perimeter]"
  advice (around): PC1 {
    task_ref "Access [maps, terrain and weather conditions and routes]" (and) {}
  }
  advice (around): PC2 {
    task_ref "Provide [location sensitive info]" (and) {}
  }
}
```

Fig. 8. Example of crosscutting relationship identified

Finally, 1 of the 6 crosscutting relationships manually identified was not covered by semi-automatic case study. The reason for this is that the amount of output relationship was not equal to or greater than 3. This relationship grouped 4 elements in advice, i.e., 4 crosscutting concerns.

Therefore, in general, we consider which the crosscutting concerns identified by ReqSys-MDD are correct, but the insights of the requirement engineer can identify other elements, which the semi-automatic strategy could not do.

Thus, this strategy facilitated the crosscutting concerns identification since it identified almost all crosscutting relationships. In this perspective, semi-automatic strategy proposed to help efficiently in the crosscutting concerns identification. More details for this case study can be found in [10].

6 Related Work

In order to respond to the necessity of identifying crosscutting concerns early in the software development lifecycle, some methods were created to systematize and make the execution of this activity easier. Therefore, we made a literature review, which aimed to seek methods and techniques for identifying crosscutting concerns in requirements documents to guide the development of the approach presented in this paper. Then, we could describe two major groups of identification approaches: (i) those which process textual requirements documents; and (ii) those which deal with a specific type of model, for instance, Use Cases and I* models.

Among the methods which process textual documents, we can highlight CCCINPL [1], Theme/Doc [6], DISCERN [11] and Early-AIM [13]. The first difference among them is the identification technique: Theme/Doc performs lexical analysis and the other three perform semantic analysis. Although these three approaches perform the same identification technique, each one is performed in a different way, such as using natural language processing or aspect mining. Other difference between them is the requirement type that is identified as crosscutting concern: Theme/Doc and CCCINPL identify crosscutting concerns in functional or non-functional requirements, but DISCERN and Early-AIM only identify in non-functional requirements.

Among the method which process specific models, we can highlight the approaches which extend UML [4][5] and the approaches which use I*[2][3]. They perform the crosscutting concern identification by different techniques: those which extend UML use semantic analysis and those which work with I* use rules (considering the intrinsic elements of those languages). Furthermore, other difference between them is the requirement type that they identify as crosscutting concern. Only those which work with I* identify crosscutting concerns in both requirement types, the other identify only in non-functional requirements.

This strategy proposed by this paper is in this second group, because it process PL-AOVgraph model and like those process I* models, this strategy identifies crosscutting concern in both requirements types – functional or non-functional requirements. Furthermore, this strategy uses the fan-out analysis as identification technique.

Above all, some of these works, or parts of them, were used as a base to define the strategy proposed here, among them CCCINPL, Theme/Doc and those which works with I*. The first, by the use of a relationship matrix to identify the verbs that influence each requirement; and the others by the heuristics used to identify crosscutting concerns.

Therefore, the contribution of the strategy and tool presented in this paper is to make semi-automatic some heuristics to identify and define crosscutting concerns in PL-AOVgraph models. This application makes a good use of its natural characteristics, such as, the modeling of both functional and non-functional requirements, the ability to make explicit the relationships between the requirements, and then, help to analyze them.

Table 3 presents a summary of the related works, showing the identification technique used by the approach and in what type of requirement it identifies crosscutting concern.

Table 3. Summary of related works

Approaches that process textual requirements documents		
Approach	**Identification technique**	**Requirement types**
Theme/Doc	Lexical analysis	Functional Requirements Non-Functional Requirements
DISCERN	Semantic analysis	Non-Functional Requirements
Early-AIM	Semantic analysis	Non-Functional Requirements
CCCINPL	Semantic analysis	Functional Requirements Non-Functional Requirements
Approaches that process specific models		
Approach	**Identification technique**	**Requirement types**
Crosscutting concern identification with UML	Semantic analysis	Non-Functional Requirements
Identifying crosscutting concern with I*	Rules	Functional Requirements Non-Functional Requirements

7 Final Remarks

This paper presents, briefly, a semi-automatic strategy for crosscutting concerns identification in PL-AOVgraph models. This strategy aims to help the requirement engineer to identify crosscutting concerns and to write the crosscutting relationships properly early in the software development process. This strategy is supported by a tool, named ReqSys-MDD. This support makes the PL-AOVgraph models better modularized and then, it makes them more easily analyzed and mapped to other stages of software development.

The Crisis Management System (CMS) [9] was used as a case study in this paper in order to evaluate the heuristics defined to identify crosscutting concerns and their implementation. This strategy facilitated the crosscutting concerns identification since it identified almost all crosscutting relationships. This result can be due the crosscutting concern identification in manual way is not only by output relationships but also by requirements engineer's insights.

As future works, it is suggested to carry out other studies case to evaluate the efficiency of the tool and, consequently, of the method proposed here, in different contexts. Furthermore, we intend to perform controlled experiments to compare the results obtained by this method and the results obtained by other crosscutting concerns identification approaches. And also, we aim to make our strategy more generic, allowing its use with others goal-based languages.

Acknowledgements. This research was performed with support from The Brazilian National Council for Scientific and Technological Development – CNPq.

References

1. Ali, B.S., Kasirun, Z. M. Crosscutting concern identification at requirement level. In: Malaysian Journal of Computer Science, vol. 21(2), pp.78-87 (2008).
2. Alencar, Fernanda; et al. "Identifying Candidate Aspects with I-star Approach". In: International Conference on Aspect-Oriented Software Development. (2006).
3. Alencar, Fernanda; et al. "Using Aspects to Simplify i* Models". In: International Conference on Requirements Engineering. (2006).
4. Araújo, J.; Moreira, A. An Aspectual Use Case Driven Approach. In: VIII Jornadas de Ingeniería de Software y Bases de Datos (JISBD 2003), Alicante, Spain. (2003).
5. Araújo, J.; et al. Aspect-Oriented Requirements with UML. In: Workshop on Aspect-Oriented Modelling with UML (UML 2002). Dresden, Germany. (2002).
6. Baniassad, E., Clarke, S. Finding Aspects in Requirements with Theme/Doc. In: Proceedings of Early Aspects 2004 (AOSD 2004). Lancaster, United Kingdom, (2004).
7. Filman, Robert E., et al. Aspect-Oriented Software Development. Boston: Pearson Addison-Wesley (2005).
8. Kiczales, G.; et al. Aspect-Oriented Programming. In: ECOOP 1997 - Proceedings of the 11th European Conference on Object-Oriented Programming, p.p. 220–242. Finland (1997).
9. Kienzle, Jörg; Guelfi, Nicolas; Mustafiz, Sadaf. Crisis Management Systems: A Case Study for Aspect-Oriented Modeling. Technical Report. McGill University, Montreal, Canada (2009).
10. Medeiros, Maíra de Faria Barros. Identificando Interesses Transversais em Modelos de Requisitos PL-AOVgraph. Master Dissertation – Federal University of Rio Grande do Norte, Natal, RN (2013).
11. Rosenhainer, L. The DISCERN Method: Dealing Separately with Crosscutting Concerns. In Proceedings of OOPSLA Early Aspects 2005. San Diego, USA (2005).
12. Sampaio, Américo; et al. EA-Miner: a tool for automating aspect-oriented requirements identification. In: International Conference on Automated Software Engineering (2005).
13. Sampaio, Américo; Rashid, Awais; Rayson, Paul. Early-AIM: An Approach for Identifying Aspects in Requirements. In: International Conference on Requirements Engineering (2005).
14. Santos, Lidiane Oliveira dos. PL-AOVgraph: uma extensão de AOV-Graph para linha de produto de software. 84 f. Monograph in Computer Science – State of University of Rio Grande do Norte, Natal (2010).
15. Santos, Lidiane Oliveira dos. ReqSys-MDD: Uma ferramenta para mapeamento entre modelos de features e requisitos em linhas de produto de software. 113 f. Master Dissertation – Federal University of Rio Grande do Norte, Natal, RN (2012).
16. Silva, Lyrene Fernandes da. Uma estratégia orientada a aspectos para modelagem de requisitos. 222 f. PhD Thesis – Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro (2006).
17. Sommerville, Ian. Engenharia de Software. 8. ed. São Paulo: Pearson Addison-Wesley, 552 p. (2007).