Proposal to deal with the Complexity of i* Models with Aspects

Fernanda Alencar¹, Ana Moreira², João Araújo³, Jaelson Castro¹,³, Ricardo Ramos¹ and Carla Silva¹

Abstract— The i* applicability of organizational modeling has been compromised by the complexity of the resulting models. The focus of this paper is to propose a systematic process to simplify the i* models, by using the main aspect-oriented concepts. The basic idea is (i) to identify and modularize crosscutting concerns (model elements that affect several other elements in the same model), and (ii) to define composition rules for these crosscutting elements (or aspectual elements) that allow us to recover the original model. Thus, these rules work as transformations in model-driven engineering.

Index Terms— Agent-Oriented Model, Aspect-Oriented Requirements Engineering, Complexity, Early-Aspect.

I. INTRODUCTION

In the last few years, new approaches and methodologies for developing large scale software and modeling organizational processes have been appeared [10]. Among them we can highlight the TROPOS framework [17], which relies on the i* framework [14] to identify requirements. i* has been used both in the academy and industry [8], [9] and provides a graphical description to specify dependencies between actors in a system, emphasizing their intentions, dependencies, responsibilities and vulnerabilities. Since we are dealing with complex systems, the i* models can become huge, failing in legibility and comprehension. On the other hand, the recent appearing of the aspect oriented programming [19], [18] has motivated the interest in identifying crosscutting concerns in earlier phases, such as requirements [3], aiming at improving system design. From this point of view, we observe that the complexity of the i* model can be caused by improperly dealt crosscutting concerns. It results in models with many dependencies and, therefore, more complexity.

Aspect oriented requirements engineering allows modularizing and composing crosscutting concerns which cannot be encapsulated using artefacts provided by both traditional approaches (e.g., use cases and view points) [12] and i* models in its conventional fashion. Thus, we are investigating the principles of aspect orientation to incorporate them in the development of i* models to reduce their complexity and improve the development of later phases of Tropos (architectural design, detailed design and implementation).

The purpose of this work is presenting a proposal to identify candidate aspects and deal with them according to aspect orientation, aiming at reducing the graphical complexity of i* models traditionally developed. To achieve this, we propose a systematic process to identify crosscutting concerns present in the i* models and compose, through rules separately defined, these concerns with the remaining elements in the model. These composition rules allow us recover, by transformation, the original i* models. The results are simpler and easier to understand, maintain and evolve models.

This paper is organized as follows. Section 2 presents an overview on aspect oriented main concepts and the i* framework. Section 3 presents an approach to identify crosscutting concerns in i* models. Section 4 defines the composition rules used in the proposed approach. Section 5 illustrates the use of our approach in an example. A Section 6 discusses some related work. Finally, Section 7 presents the conclusion and future work.

II. BACKGROUND

A. Aspect-Orientation

The Aspect Oriented Software Development’s (AOSD) purpose is to develop techniques and mechanisms to support the identification, modularization, specification and of crosscutting concerns⁴. Crosscutting concerns means software properties which cannot be modularized effectively through traditional software development techniques, such as object oriented methods. Notice that “crosscutting” is the relationship between several concerns. When we say that a concern is crosscutting we are implicitly acknowledging some dominant decomposition that offers the base over which the crosscutting concern cuts across. This relationship depends on the chosen representation to model, specify or implement the concerns. It means that what is crosscutting in a object oriented representation cannot be crosscut-

¹ Universidade Federal de Pernambuco, Brazil, (e-mail: fmra@ufpe.br; jbc@cin.ufpe.br; raz2@cin.ufpe.br; ctils@cin.ufpe.br).
² CITI/FCT, Universidade Nova de Lisboa, Portugal, (e-mail: amm@di.fct.unl.pt; ja@di.fct.unl.pt).
³ Feder Istituto Trentino di Cultura – ITC, Istituto per la Ricerca Scientifica e Tecnologica – IRST, Trento-Povo, Italy, (e-mail: jaelson@ite.it).
* Currently on leave of absence at IRST, Trento, Italy

⁴ The interested reader can consult, for example, (aosd.net) and [4].
ting in a functional representation and vice-versa. This is the reason by which we consider throughout the text the expression candidate to crosscutting concerns. Typical examples of crosscutting concerns are the non-functional requirements, such as, security, fail tolerance and persistence. However, crosscutting concerns can be functional requirements, such as, auditorship, validation, monitoring or bidding.

Crosscutting concerns are encapsulated in separate modules, the aspects, to promote localization. Composition mechanisms are later used to observe the tangling of these modules with other modules in compilation time, in loading time or in run time. However, both aspects and its compositions have an important role still before implementation. In general, aspectualization results in a better support to modularization, which can reduce development, maintainance and evolution costs of the intended software system.

B. The i* Framework

The purpose of the i* framework is to make easier the earlier comprehension of the intended system in the development process of agent oriented software emphasizing social factors. The social nature of i* is represented through actors and relationships between them. This approach provides a graphical view of both system’s actors (agents, roles and positions) and their intentions, dependencies, responsibilities and vulnerabilities. i* proposes developing two models: the Strategic Dependency (SD) model and the Strategic Rationale (SR) model.

SD model includes a set of nodes and connector links between these nodes. Nodes represent actors (depender – the depending actor – and, dependee – the actor who is depended upon). Dependency among actors is called dependum. For example, in Fig. 1, actor A (dependee) is related with actor A (dependee) through dependum D. A dependum can be a goal, a task, a resource or a softgoal.

SR model incorporates two new types of relationships, besides SD’s four types of dependencies: means-end and task-decomposition. Means-end links suggest that a model element (T) can be provided as a means to achieve the other model element (G). In general, the means element is always a task, while the end element can be a dependum (goal, task, resource or softgoal). Task decomposition links describe what can be made to perform a certain task. The task being decomposed (T) can possess any type of dependum as sub-element (dependum tree). Fig. 2 presents a view of the SR model for the expanded A actor (dependee). To ensure the consistency between the SD and SR models, all the dependencies present in the SD model are preserved in the SR model. In the right side of Fig. 2, the element Dependum tree represents any type of dependum or another model sub-graph.

III. IDENTIFYING CROSSCUTTING CONCERNS IN THE i* MODELS

The purpose is to identify the probable crosscutting requirements, functional or non-functional. In [1] and [2] we have proposed a set of rule to identify candidate aspects (or candidate to crosscutting concerns). The general approach is illustrated in Fig. 3. This approach receives as entrance data the i* models and is based on four phases: (1) identification and representation of candidate aspects; (2) creation of the relationship matrix between the identified candidates;
His paper focuses on activities 2 and 3 which deal with the construction of the relationships matrix and the composition rules. The first activity of the approach, identification and representation of crosscutting concerns, has been discussed in [1, 2]. The trade-offs analysis will be explored in future work, having as basis our previous results [12].

A. Identification and Representation of Aspects Candidates

In [1], [2] we propose three guidelines to detect crosscutting concerns in the SD model and in the SR model, as well as to eliminate redundancies, respectively.

Guideline 1 (in the SD model): when the same dependum is required by at least two dependees (actors), its operationalization, assumed as the same for all the actors, is a candidate to crosscutting concern.

Guideline 2 (in the SR model): if a task is involved in a decomposition relationship, direct or indirectly related to an external dependency, is required by two or more tasks which are also related to another external dependency, then those task is a candidate to crosscutting concern. Notice that a task is indirectly related to an external dependency (dependency in the SD model) if, in the hierarchy it belongs to, at least one of its parents is connected to an external dependency.

Guideline 3 (remove redundancies): the crosscutting concerns identified by the guideline 2, which already are the operationalizations of the crosscutting concerns identified by the guideline 2, need to be grouped in just one concern. The final concerns are the ones which correspond to operationalizations.

After identifying the concerns, the SR model is analysed, by trying to prove if the operationalizations are the same in the several actors (dependee) where the concerns crosscut. In the affirmative case, these concerns can be aspectualized. Thus, following the AOSD principles, the tasks (operationalization/concerns) are externalized and modularized. The candidate aspect is represented by a star (Handle $D_i$, in Fig. 4), which when expanded, contains the aspectual task $T_i$ (candidate to crosscutting concern and externalized in the $i^*$ model) and a set of Composition Rules for $T_i$.

Thus, from this representation, and applying the composition rules, the simplified SD model is obtained and illustrated in Fig. 5. All the candidate aspects are represented as stars. He crosscutting relationship between each aspect and the other model elements are represented as arches with black triangles using the rotule “Crosscut”. The direction indicated by the triangles suggests the composition direction, meaning that the behaviour of the source element needs to be repassed to the behaviour of the target elements.

B. Relationship Matrix between Crosscutting Concerns and Actors

From both $i^*$ models (Fig. 1 and Fig. 2) and the proposed identification guidelines (Section 3.1) we related in a table (Table I) the candidate to crosscutting concerns (columns) with the respective actors (dependee) which are responsible for operationalize them.

Each cell in Table I contains information about which dependum is going to be operationalized by the task (candidate to crosscutting concern) and which are the actors dependent (dependee) on this dependum operationalization ($D_i$), here represented by a set of dependee actors $\{A_1, ..., A_n\}$. In such way, this matrix makes easier to identify the concerns scattering through several actors (dependee) and its dependet actors, becoming useful for defining composi-
tion rules. For example, the dependum $D_1$: (i) will be operationalized by the actors $A_n$ and $A_i$; (ii) will have as actor which depend on $A_n$ $\{A_i, A_n\}$. Thus, in the set of composition rules for the concern $T_1$, will exist a rule that indicates that actors need these concerns and which actors dependee will be operationalizing.

### TABLE I

<table>
<thead>
<tr>
<th>Actor (Dependee)</th>
<th>Crosscutting concerns candidates</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_1$</td>
<td>$A_n, A_{n+1}$</td>
</tr>
<tr>
<td>$T_2$</td>
<td>$A_{n+1}/D_2$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>$T_k$</td>
<td>$A_{n+1}/D_k$</td>
</tr>
</tbody>
</table>

#### C. Composition

The composition rules define “where” and “how” a crosscutting concern affects other concerns in a system. Hence, before performing a composition, it is necessary to identify, in the SR model, which elements are related with the crosscutting concerns and, in the SD model, which actors depend on the operationalization of these concerns. As we highlight in the identification rules, the crosscutting concerns are tasks in the SR model. In this model, tasks either (i) are going to be decomposed into sub-elements (which can be any type of dependum, including another task); or (ii) are going to be the mean to achieve an end (means-end link, where we consider that the most common situation is that the end is a goal or a softgoal. We call parent (task) the task that is being decomposed (decomposition task link in the SR model). Hence, in Table II, for each of the identified actors (line), it is informed which elements are related to these crosscutting concerns. In the case of the crosscutting concern is the parent task itself and then we associate to it just the dependee actor name. To fill in the Table II, we use as reference the model depicted in Fig. 1 and Fig. 2 and Table I.

### TABLE II

<table>
<thead>
<tr>
<th>Actor (Dependee)</th>
<th>Crosscutting Concerns Candidate</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_1$</td>
<td>$A_n, A_{n+1}$</td>
</tr>
<tr>
<td>$T_2$</td>
<td>$A_{n+1}/D_2$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>$T_k$</td>
<td>$A_{n+1}/D_k$</td>
</tr>
</tbody>
</table>

To completely fill the Table II, it is necessary to expand all the actors (dependee) of our interest through the SR model. In this case, for the generic example in Fig. 2, we just expand the actor $A_n$ (reason by which Table II has only one line). The information presented in Table II will help us to define composition rules that can be instantiated by the candidate aspects represented in the i* models.

#### IV. DEFINITION OF COMPOSITION RULES FOR ASPECTUALIZED i* MODELS

The possible set of rules is defined based both on the i* models features and the information stored in Tables I and II.

In the SD model, what is important to preserve is the dependum type that is involved with a dependency relationship (goal, task, softgoal or resource), as well as the actors who depend on the dependency relationship (dependee actors). Hence, for these situations, respectively, we propose: (i) an operator **is the type of**: and (ii) an operator **has dependee ... with dependee**. Table III presents a general form of the rules that preserve the links in the SD model.

### TABLE III

<table>
<thead>
<tr>
<th>Aspect&lt;name&gt;</th>
<th>is the type of</th>
<th>&lt;dependee name&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_n$</td>
<td>$D_1$</td>
<td>$A_i$</td>
</tr>
</tbody>
</table>

In which “$\times$” works as a separator between rules variation. Elements in a list are separated using “,”.

The first rule shows the dependum type involved in the crosscutting relationship, while the second one define, in relation to the dependee actor, who are the actors that depend (dependee) on the object associated to the operationalized concern in the dependee actor. Thus, for each actor which operationalize the concern we have to ve an actor or a list of associated dependent actors (composed with the operator “,”). Considering the crosscutting concern **Handle $D_i$** (Fig. 5 and Table I) and assuming that $D_i$ is a resource,

Aspect **Handle $D_i$** is the type of resource

Aspect Handle $D_i$ has depend $A_{n-1}$ with dependee $A_i$

Aspect **Handle $D_i$** has depend $(A_i, A_{n-1})$ with dependee $A_n$

These rules must be part of the set of rules in all the representation of the crosscutting concerns externalized as aspects in the i* models.

The second step focuses on the SR model. When we externalize a crosscutting concern, all the links from this element with others in the SR model must be kept in the new aspectual model (e.g., in a task decomposition, keep all the links with its sub-elements). Let’s consider **Handle $D_i$** a candidate aspect. After externalizing this element for the SD model, Fig. 6 is obtained. When we compare this model with the original model (Fig. 2), we need to define rules to keep the decomposition link between tasks $T_1$ and $T_2$, as well as the link with the end goal ($G_i$), since $T_1$ represents the means in a link means-end. To achieve this, we define the rules for a task decomposition, in which the aspectual task incorporated by the aspect is a sub-element of a decomposition. We propose an operator **is sub-task of** to indicate the parent task which possesses the aspectual task. If this task belong to more than one decomposition tree, therefore has more than one parent element, then a list of parent tasks is established. The actor where the task is being operationalized must be informed. Hence we define an operator **in**
which informs the dependee actor.

Hence, generically, we have the rule for those situation expressed in its general form illustrated in Table IV.

<table>
<thead>
<tr>
<th>Aspect &lt;name&gt;, &lt;aspectual task&gt;</th>
<th>is a sub-task of</th>
<th>in &lt;dependee name&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;task name&gt;</td>
<td>&lt;list of tasks name&gt;</td>
<td>Aspect.&lt;task name&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Aspect.&lt;list of tasks name&gt;</td>
</tr>
</tbody>
</table>

In the case of analysing the D₁ element in Fig. 6, we have:

Aspect Handle D₁.T₁ is a sub-task of Tₖ in Aₙ
This means that the externalized task T₁ is a sub-task of task Tₖ which belongs to the dependee actor Aₙ. However, when observing Table II, we see that Tₖ is also a candidate aspect, thus we must change the rule to:

Aspect Handle D₁.T₁ is a sub-task of (Aspect, T₁) in Aₙ

A second case to be analysed is related to a means-end link. In this case. We propose the operator is the mean of which defines from who the aspectual task is a means. The general rule is defined in Table V.

<table>
<thead>
<tr>
<th>Aspect &lt;name&gt;, &lt;aspectual task&gt;</th>
<th>is the mean of</th>
<th>in &lt;dependee name&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;end name&gt;</td>
<td>&lt;list of end name&gt;</td>
<td>Aspect.&lt;end name&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Aspect.&lt;list of end name&gt;</td>
</tr>
</tbody>
</table>

In this case we are considering that the end element can be a goal or a softgoal, although it can be any type of dependend. Thus, for the concern Handle D₁, we have:

Aspect Handle D₁.T₂ is the mean of G₁ in Aₙ

The third rule type is related to the links which must be recovered when the aspectual task is the root element in a task decomposition link. In this case, all the links with its sons must be reestablished through the composition rule. In Table VI we present the general rules.

| Table VI |
| RULES FOR THE ROOT OF A TASK DECOMPOSITION LINK |
|-----------------------|-----------------------|-----------------------|
| Aspect <name>, <aspectual task> | is the root of | in <dependee name> |
| <list of sub-element name> | Aspect.<sub-element name> | <list of Aspect.<sub-element name> |
| | | <list of Aspect.clist of sub-element name> |

The second rule expresses the situation in which he aspectual task is the root of a whole decomposition tree in the dependee actor under analysis.

Aspect Handle D₂, T₂ is the root of (Tₖ and G₁ and (Dependum tree)) in Aₙ

Aspect Handle D₂, T₂ is root in Aₙ

If D₂ is a goal the aspectualization of Handle D₂, taking into account Table I and II, we have the following rules related to the dependee actor Aₙ:

Aspect Handle D₂ is the type of goal
Aspect Handle D₂ has depender Aₙ₋₁ with dependee Aₙ
Aspect Handle D₂ has depender Aₙ with dependee Aₙ

To illustrate this set of rules, let’s consider the aspectualization of the Handle D₂ element, whose result can be found in Fig. 7.

In Table VII we present in summary all the operators proposed in the decomposition rules.

| Table VII |
| OPERATORS USED IN THE COMPOSITION RULES |
|-----------------------|-----------------------|-----------------------|
| Operator | Description |
|-----------------------|-----------------------|-----------------------|
| with dependee | It defines the the role which operationalizes the cross-cutting concern. |
| is the type of | It defines the type of dependence that was included by the aspect. |
| has depender | It restores the link among the dependence element included by the aspect and the dependee actor. |
| Aspect | It defines the object as an aspect. |
| <name> | It represents a name. |
| in <dependee name> | It informs in which dependee actor the action is being executed. |
| is root | It defines that the aspectual task is the root of the whole decomposition tree. |
| is sub-task of | It defines the decomposition link between the aspectual task and its “father” task. |
| is the means of | It restores the means-end link between the aspectual task and its end. |
| is the root of | It informs that the aspectual task is being decomposed and restores the link with its parts. |
| <list of…> | It represents a group of elements separated by “,” |
The chosen case study is based on the YKeyK System (Your Key Knows) [13]. YKeyK (Your Key Knows) is a special car key equipped with a special system that helps drivers to find their cars in a multi-storey car parking. The key guides the driver through the shortest way to the parking space where his/her car is parked. This system works in collaboration with two other systems: the car location system which is installed in the car and the parking system which is installed in the car park. The parking system has the information about the topology of the park, the pricelist (cost per hour) as well as the car entry time (hour and minute). The YkeyK can be in two modes, (i) the monitoring mode and (ii) the car searching mode (changing modes is achieved by pressing a button). In the monitoring mode the YkeyK has no signal connection to the “car location system”; it shows the current time and date on the screen and gives some relevant information to the driver (e.g., how long the car has been in the park). In Fig. 8, we see the SD model for the YkeyK System.

Fig. 8. The SD model of the YKeyK system.

YKeyK is composed of four parts: Calculator, Display, Searcher and Monitor. To exemplify, we have chosen the Searcher, which is responsible for sending, receiving and processing the signals needed to find the vehicle, and the Localization System, external to the YkeyK System. These actors are expanded, in the SR model in Fig. 9, highlighting only the task decomposition links.

Identifying and Representing Aspect in i*

Applying the Guideline 1 to the SD model in Fig. 8, we obtain the following candidates to crosscutting concerns: Enviar Signal que operacionaliza o dependum Signal; Handle Correctness, Handle Response time and Handle Availability which operationalizes, respectively, the Correctness, Response time and Availability dependums.

Applying the Guideline 2, the candidate to crosscutting concerns are the tasks: Process signal, Handle Correctness, Handle Response time and Handle Availability.

Guideline 3 deals with the possible duplications obtained from the application of the Guidelines 1 and 2. Hence, the final list of the candidate to crosscutting concerns is: Enviar
Sinal, Process signal, Handle Correctness, Handle Response time and Handle Availability.

Using the representation of candidate aspect present in Fig.4, we obtain the SR model depicted in Fig.10.

Fig. 9. The partial SR model of the YKeyK system
A. Relationship Matrix of Crosscutting Concerns

The crosscutting concerns are stored in Table VIII.

**TABLE VIII. CROSSCUTTING CONCERNS AND ITS DEPENDENCY TO THE YKEYK SYSTEM**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Handle Response Time</td>
<td>Handle Correctness</td>
<td>Handle Availability</td>
<td>Send Signal</td>
<td>Signal be processed</td>
<td>Searcher {LS}, RT</td>
<td>Localization System {LS}, RT</td>
</tr>
<tr>
<td>Searcher [LS], RT</td>
<td>[LS], C</td>
<td>[LS], A</td>
<td>[SL], S</td>
<td>[LS], SP</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

B. Applying Composition Rules

From both Table VIII and the analysis of the SR model for the YKeyK system (Fig. 9), we present in Table IX, the point-cuts for which we are going to instantiate the composition rules proposed for the two actors: Searcher and Localization System.

**TABLE IX. IDENTIFICATION OF THE YKeyK SYSTEM POINTCUTS**

T2: Recognize Press Button; T3: Generate Direction Signal; T4: Receive Signal; T5: Transfer Signal; T6: Send Signal; T7: Get Direction Signal; T8: Support the YKeyK

<table>
<thead>
<tr>
<th>Actor (Dependee)</th>
<th>Crosscutting Concerns Candidates</th>
<th>Handle Response Time</th>
<th>Handle Correctness</th>
<th>Handle Availability</th>
<th>Send Signal</th>
<th>Process Signal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Searcher</td>
<td>T6, T5, T4</td>
<td>T6, T5, T4</td>
<td>T6, T5</td>
<td>T3, T2</td>
<td>T1</td>
<td>T4, T3</td>
</tr>
<tr>
<td>Localization System</td>
<td>T7</td>
<td>T7</td>
<td>T7</td>
<td>T7</td>
<td>T7</td>
<td>T7</td>
</tr>
</tbody>
</table>

Let’s consider only the crosscutting concern Handle Response Time (HRT) and let’s apply the composition rules proposed in Tables 3-7 in Section 4. Hence, through the rule related to dependency links in the SD model (Table III), we have:

**Aspect HRT is the type of softgoal;**

**Aspect HRT has dependee** Localization System with dependee Searcher;

**Aspect HRT has dependee** Searcher with dependee Localization System;

Through the SR model, one can see that the crosscutting concern is related to a task decomposition link, thus we instantiate the rules present in Table IV. Observing Table 9, we see that none of the parent tasks of this concern will be an operationalization of another crosscutting concern. Thus:

**Aspect HRT is a sub-task of** \( \{T_4, T_5, T_6\} \) **in Searcher;**

**Aspect HRT is a sub-task of** \( T_7 \) **in Localization System.**

The model present in Fig. 10 and the set of composition rules associated to the aspectualized concerns are achieve in a similar way using the remained concerns in Table 8.

C. Discussion

Externalizing and Modularizing the crosscutting concerns, we could reduce the graphical complexity of the i* models. It represents an evident advantage, especially in large complex systems.

On the other hand, for simpler systems, we can loose the graphical view, similarly, needing to use the composition rules. We highlight that through the proposed composition rules, all atomic links between element in the i* models are preserved.

Our approach presents two advantages: (1) it is a process simpler and faster than others [15, 16], since it allows identifying candidate aspects directly from i* models; (2) it allows us to develop less complex i* models, keeping the same information of the original models.

VI. RELATED WORK

In [13] we identify aspects in the requirements engineering level. This work has been made for the approach presented in [6], while this new approach in more generic.

In [16] the authors points that the satisfaction of the OR decomposition of sub-goal in the KAOS models [7], tipically, leads to tangled implementations. Moreover, agents responsible for multiple refined goals into an OR decomposition can be implemented in the same way that happens in the aspect orientation. Instead proposing requirements engineering handling mapped directly to aspects, they examine well established requirements engineering models and identify patterns in these models which could be better designed and implemented using aspect orientation. From the code and through aspect mining techniques, they identify crosscutting features in the application. In another step, they run a consolidated requirements model in terms of KAOS concepts. Then, they compare the goal decomposition and the code decomposition, trying to identify connection patterns in the goal decomposition graph that probably will result in aspects.

In [15] the author show that aspects can be discovered during goal oriented requirements analysis. The proposal includes a systematic process to identify aspects from relationships between functional and non-functional goals. The process presents a systematic way to refine a graph, called V-graph. During each step of the process, the analysis tool is used to detect conflicts and loosees. The process finishes when all the root goals and softgoals are satisfied.

In [12] is also proposed an approach to modularize and compose concerns, which in the requirements level, crosscuts other concerns. These crosscutting concerns are aspectual responsible, having as basis the separation and the specification of the aspectual requirements. The implementation of these approaches is based on the system modeling using view points and using XML (eXtensible Markup Language) and is supported by a tool called ARCaDe. In [5] one have and extension of this work, proposing a new process to compose the concerns in which it is introduced the notion of match point, dominat...
Our approach identifies the crosscutting concerns directly from the organizational model and does not need to use auxiliary requirements engineering techniques to achieve the purpose. The proposed composition rules ensure that all the links between the elements in the original models are reestablished.

VII. CONCLUSION AND FUTURE WORK

This work describes an approach that use aspect orientation to simplify i* models. Thus, we present guidelines to help identifying candidate to crosscutting concerns directly from the information available in the i* models, we define a means to represent candidate aspects in the models and we propose composition rules to allow us generating extended i* models, in any time.

As future work we intend to: apply our approach to more case studies; implement aspectal concepts in i* modeling support tools (e.g. OME); developing a metamodel for our approach.

ACKNOWLEDGMENT

This work was supported by several research grants (CNPq Proc. 304982/2002-4, Proc. 142248/2004-5, CAPES Proc. BEX 1775/2005-7, Proc. BEX 3014/05-3 & CAPES/ GRICES Proc. 129/05).

REFERENCES