Improving the Modularity of *i** Models

Fernanda Alencar[1], Márcia Lucena[2,], Carla Silva[3],
Emanuel Santos[4], Jaelson Castro[4]

[1]Universidade Federal de Pernambuco - UFPE, Departamento de Eletrônica e Sistemas, Recife, Brazil,
fernandaalenc@gmail.com
[2]Universidade Federal do Rio Grande do Norte - UFRN, Departamento de Informática e Matemática Aplicada Natal, Brazil,
marciaj@dimap.ufrn.br
[3]Universidade Federal da Paraíba - UFPB, Centro de Ciências Aplicadas e Educação, Rio Tinto, Brazil
carla@dce.ufpb.br
[4] Universidade Federal de Pernambuco - UFPE, Centro de Informática, Recife, Brazil
{ebs,jbc}@cin.ufpe.br

Abstract. i* offers expressive models to capture social and intentional characteristics of a system organizational context, and explicitly captures stakeholders' motivations and rationale in a requirements model. Thus, the more detailed i* models are, the more complex they become. Hence, i* models can become unnecessarily hard to read, understand, maintain and reuse. In the past years we have been investigating how to tame the complexity of the models, with a view to improve their modularity. This paper presents two of our strategies. The first one relies on aspect-orientation principles whereas the second one is based on model transformations.

Keywords: i*, modularization, Aspects, Model Transformations.

1 Introduction

Modularity measures the degree to which the modeling language offers well-defined building blocks for building model. Although i* incorporates a decomposition mechanism based on strategic actors, which could be used to improve modularization of i* models, the way in which this mechanism is used is often not suitable to produce models that are easy to maintain and reuse. Current modeling methods represent the rationale of an actor in a monolithic way [3],[6]. Besides, sometimes several refinements are described in a scattered and tangled form (also known as crosscutting), making it hard to visualize the boundaries of sub-graphs related to specific domains. This poor modularity compromise the management of the complexity of the models, an important pre-requisite for the adoption of i* in industrial settings [4]. In order to reduce the complexity of i* models and increase their modularity we proposed two strategies; the use of aspect oriented principles [1] or the adoption of a model transformation strategy [8]. The paper is organised as follows. Section 2 describes a strategy to improve the modularity of i* models using

aspect oriented principles. Section 3 presents an approach which relies on the definition of transformation rules to re-structure the models. Section 4 discusses results obtained and Section 5 points out ongoing and future research.

2 Modularizing i* with Aspects

The modularity of i* models can be improved by removing tangled and scattered information into aspectual actors together with some weaving mechanisms [1]. Our aspectual approach consists of (i) a set of guidelines to identify crosscutting concerns in i* models; and (ii) an extension of the i* modeling language [11] by adding aspectual constructors to modularize crosscutting concerns and to allow its graphical composition with other system modules (Fig 1).

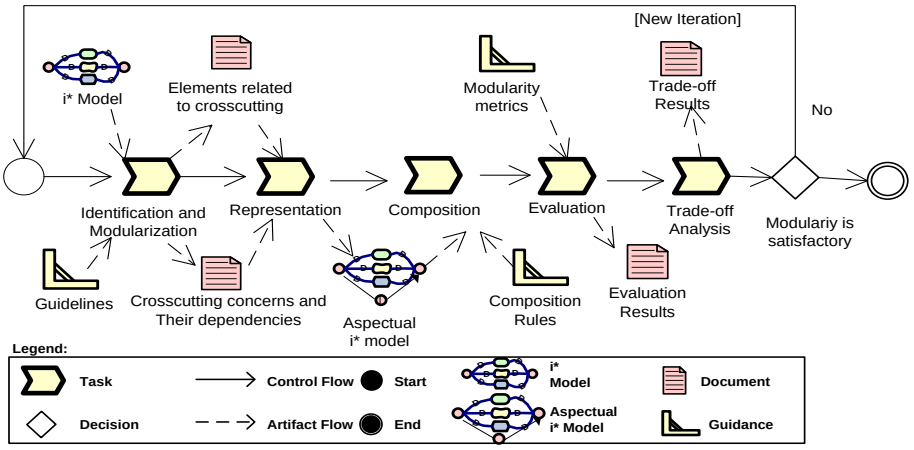

Fig. 1 – The modular i* with aspects strategy.

In this approach the crosscutting concerns are extracted into modules, called aspects, which are later composed back to the base model. Hence, we claim that Aspect-Oriented Software Development (AOSD) mechanisms [5] can contribute to increase the modularity of i* models. Four guidelines were proposed to deal with the identification, separation and modularity of the crosscutting concerns. Once identified, the crosscutting concerns are removed from the original actors, and placed in a new type of model element, the so called *Aspectual Element*. This element will have a specific graphical representation. Later it will be composed (weaved) with an actor or another aspectual element using a *Crosscut Relationship*. This relationship specifies how an i* element, located inside an aspectual element, is related to another i* element located inside an actor or another aspectual element. The composition step can be performed by graphical transformations. The evaluation of the resulting i* models is based on a suite of metrics adapted from the literature. Finally, a trade-off analysis will be performed and if the results are not appropriate (modularity is still poor) a new interaction may be executed.

3 Modularizing i* by means of Model Transformations

Another approach to improve the modularity of i* models is to restructure the models in order to extract the information that are not fully related to the application domain. To balance the responsibilities of the system actor, this information could be delegated to new system actors. Hence, we could transform the original model into a more modularized one.

Our model transformation approach consists of three activities (Fig 2): (i) Analyze Internal Elements, where Internal Elements can be factored out from software actor are identified; (ii) Apply Transformation Rules, which relies on model transformation rules to systematically move (delegate) the identified internal elements from software actor to new actors; (iii) Evaluate i* Models, used to evaluate the modularization of the models. The process is semi-automatic since the activities (ii) and (iii) can be automated, while the analysis of internal elements activity (i) depends on requirements engineers and domain experts. In this case, it is necessary to use: (i) heuristics to guide the decomposition of the software actor; (ii) a set of rules to transform i* models in modular i* models; (iii) metrics to evaluate the degree of modularity of both initial and final models. Further details can be found in [8].

Some measurement is required to check the improvement of the modularity. If the modularization still is inappropriate, new iterations may be necessary. These modular i* models are used as the starting point to generate architectural descriptions from requirements models [9].

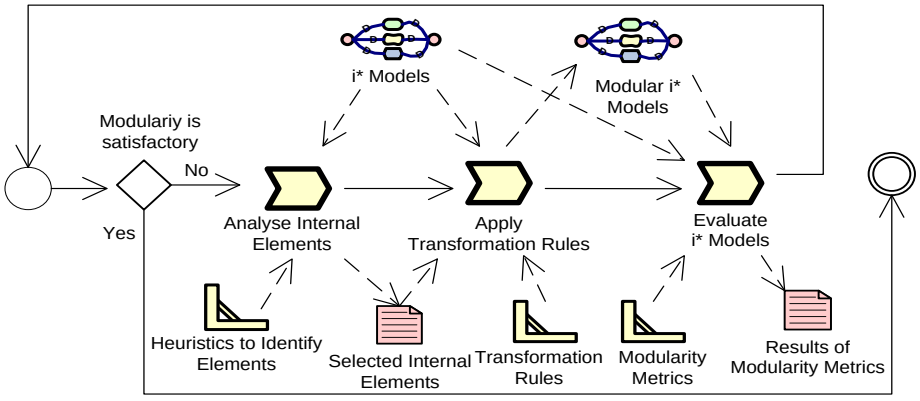

Fig. 2 – The modular i* with model transformation.

In order to illustrate the techniques used in this work, we review the *Media Shop* example [3]. *Media Shop* is a store that sells and ships different kinds of media items. To increase market share, *Media Shop* has decided to use the *Medi@* system, a business to customer retail sales front-end on the Internet.

Often i* models are overloaded with information capturing features of both the system organizational environment and the software system itself. However, the more detailed i* models are, the more complex they become (Fig. 3). This rich ontology aligned with the common misuse of the decomposition mechanisms provided by the i*, can head to models unnecessarily hard to read, understand, maintain and reuse.

The proposed approach allows delegation of different issues of a problem, initially concentrated into a single system actor, to new actors, which allows dealing with each actor separately. Details on an earlier version of this activity can be found in [8]. We have added a new rule, to deal with a special situation that may arise when independent sub-graphs, i.e., sub-graphs from different domains, have the same root goal. These sub-graphs are alternatives to satisfy this root goal. In this case, an actor is created for each alternative. Later, each of them will be considered as a different architectural solution.

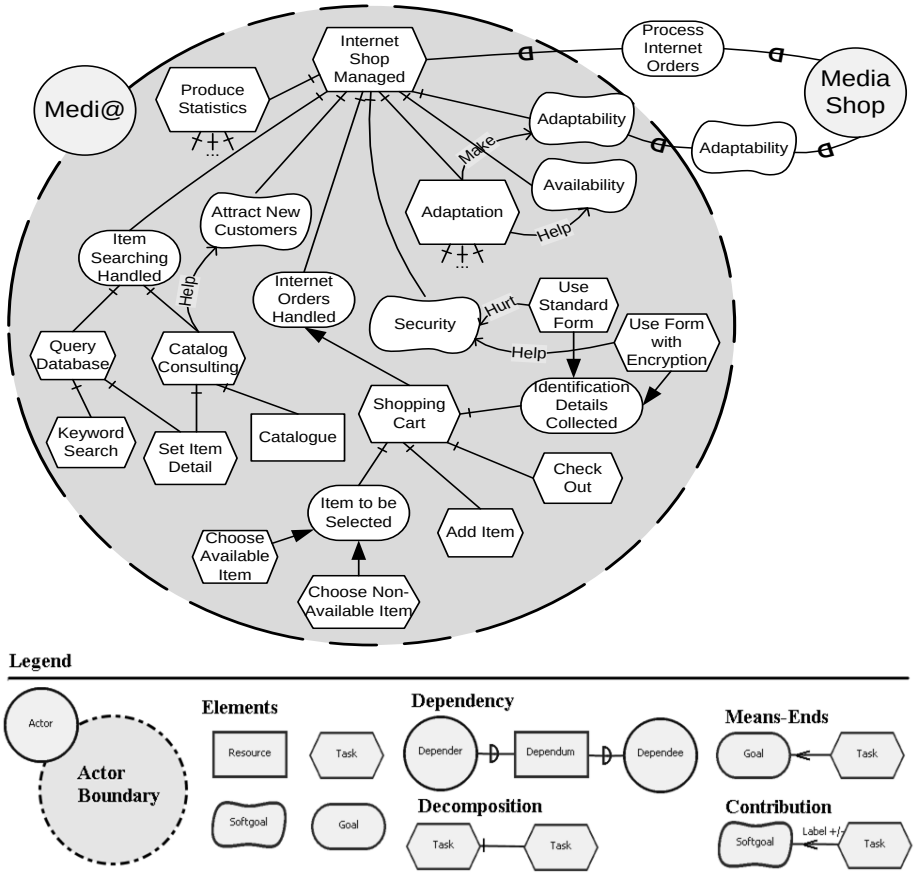

Fig. 3 – SR model for Medi@ system.

After carrying out the *Prepare Requirements Models* activity, the resulting model is decomposed into more modularized software actors (see Fig. 4). In our example there are two alternatives to achieve the *Identification Details Collected* goal (see Fig. 3). One relies on the use of standard forms, while a second alternative is to use encrypted forms. If we apply horizontal rules (those that transform an initial i* model into a more modular one [8]) each alternative previously identified is moved to a different actor (see A1 and A2 dependencies in Fig. 4). Thus, in our *Medi@* example, we will have two SR i* models representing different configurations of system and to be considered in the next activity. For the sake of space, here we present both

alternatives in the same model. In fact, different SR models should have been used to represent each alternative. But an interested reader can easily extract them.

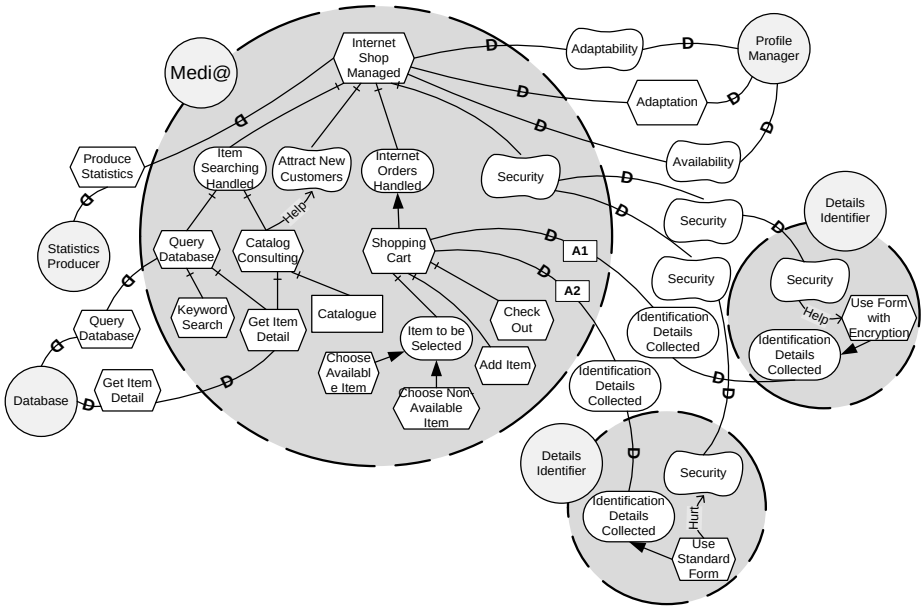

Fig. 4 – Modular i* model after model transformations.

4 Discussion

The aspectual approach contributes to increase modularity of i* models and, as demonstrated by the application of the metrics in [1], the number of concerns in a single module was reduced. Also, the models' visual complexity decreased, which may improve model understandability. This approach was applied to two case studies: the meeting scheduler problem [2] and a web-based information system [1].

However, the approach relies on aspect oriented principles. The big disadvantages of this strategy is the need to introduce new elements (namely aspects) in the original i* semantics. If the reader is familiarized with the aspect oriented principles this is not a cognitive burden. Otherwise some learning curve is required.

The second modularization approach relies on model transformations. The evaluation results demonstrated that it also promotes reduction of complexity in i* models. Besides, the proper definition of rules (for example in OCL, QVT or ATL) enables the semi-automatic derivation of modular i* specifications as well as can contribute to keep traceability among software artifacts. Note that since it does not introduce new elements to the i* syntax/semantics it is of easier adoption. This approach was applied to two case studies: a web-based recommendation system [10] and a web-based information system [9].

Both approaches can be used in a complementary way. The second approach could be used to decompose a system actor overloaded of responsibilities into several new

system actors, whereas the first approach could be used to identify the crosscutting concerns present in the i* models and separate them into aspectual elements.

5 Ongoing and Future Work

Currently we are evolving the Istar Tool [7] to support our modularity approaches. As future work, we intend to unify our approaches to decrease complexity, and to increase modularity and separation of concerns in i* models.

The identification of suitable metrics for goal modeling is also advancing, as other case studies are performed in an experimental setting. We also need to validate the metrics. We plan to define a trade-off analysis method to complement the aspectual i* process and to investigate the use of modularized i* models to support early architectural design. We aim at the decrease of coupling and improvement of separation of concerns, issues which are critical when dealing with large and complex projects. We also plan to evaluate and improve the quality of i* models [10].

References

1. Alencar, F., Castro, J., Lucena, M., Santos, E., Silva, C., Araújo, J., Moreira, A.: Towards Modular i* Models. Requirement Engineering Track at 25th ACM symposium on Applied Computing, SAC 2010. pp. 292-297 , Sierre, Switzerland (2010).
2. Alencar, F., Moreira, A., Araújo, J., Castro, J., Silva, C., Mylopoulos, J.: Towards an Approach to Integrate i* with Aspects. 8th Intl. Bi-Conference Ws. on Agent-Oriented Information Systems, AOIS'06 at CAISE'06. pp. 183-201, Luxembourg, June (2006).
3. Castro, J., Kolp, M. and Mylopoulos, J.: Towards Requirements-Driven Information Systems Engineering: The Tropos Project. Information Systems Journal, Elsevier, Vol 27: 365--89 (2002).
4. Estrada, H., Rebollar, A., Pastor, O. and Mylopoulos, J.: An Empirical Evaluation of the i* Framework in a Model-Based Software Generation Environment. In: CAiSE'06, LNCS 4001, Springer, pp. 513--527 (2006).
5. Filman, R., et al..: Aspect- Oriented Software Development. Addison-Wesley (2005)
6. Grau, G., Franch, X., Maiden, N. A. M.: PRiM: An i*-based process reengineering method for information systems specification. In: Information and Software Technology, v. 50, pp. 76-100 (2008).
7. IstarTool Project: A Model Driven Tool for Modeling i* models. Available at http://portal.cin.ufpe.br/ler/Projects/IstarTool.aspx, Mar (2010) .
8. Lucena, M., Silva, C., Santos, E., Alencar, F., Castro, J.: Applying Transformation Rules to Improve i* Models. In Software Engineering and Knowledge Engineering (SEKE 2009). pp. 43-48 , Boston, USA (2009).
9. Lucena, M., Castro, J., Silva, C., Alencar, F., Santos, E., and Pimentel, J.: A Model Transformation Approach to Derive Architectural Models from Goal-Oriented Requirements Models. 8th Int. Ws. On System/Software Architectures (IWSSA'09) LNCS. Berlin, Heidelberg: Springer, vol. 5872, pp. 370--380 (2009).
10. Ramos, R., et al.: AIRDoc-Approach to Improve Requirement Documents. In: XXII Brazilian Symposium on Software Engineering (SBES'08), pp. 1--16 (2008) .
11. Yu, E.: Modeling Strategic Relationships for Process Reengineering. Ph.D. thesis. Department of Computer Science, University of Toronto, Canada (1995).