IMPROVING THE UNDERSTANDABILITY OF I* MODELS

Fernanda Alencar
Departamento de Eletrônica e Sistemas, Universidade Federal de Pernambuco, R. Acad. Hélio Ramos S/N, Recife, Brazil

Carla Silva[1], Márcia Lucena[1,2], Jaelson Castro[1], Emanuel Santos[1], Ricardo Ramos[1]
[1]*Centro de Informática, Universidade Federal de Pernambuco, Av. Prof. Luiz Freire S/N, Recife, Brazil*

[2]*Departamento de Informática, Universidade Federal do Rio Grande do Norte, Campus Universitário, Natal, Brazil*

Keywords: Requirements Engineering, Goal-Oriented Framework, Modelling Language, Structuring Mechanisms.

Abstract: Requirements engineering (RE) has been considered a key activity in almost all software engineering process. i* is a goal-oriented approach widely adopted in the earlier phases of RE, as it offers a modelling language that describes the system and its environment in terms of actors and dependencies among them. However, often the models become cluttered even for small applications, compromising their understanding, evolution and scalability. In large and complex applications, this problem increases significantly. In this paper we investigate the use of structuring mechanisms to deal with the complexity which may arise when i* is used to model complex domains.

1 INTRODUCTION

Requirements specification should include not only software specifications but also business models and other kinds of information describing the context in which the intended system will operate. During the early stages of requirements engineering, it is necessary to identify and specify how the intended system meets organizational goals, why the system is needed, what alternatives were considered, what the implications of the alternatives are for the various stakeholders, and how the stakeholders' interests and concerns might be addressed. The i* framework provides expressive models to achieve these, wherein motivations and rationale are explicitly captured in a requirements model. Thus, the i* framework (Yu, 1995) is becoming widely used for organizational modelling, capturing social and intentional characteristics of the system organisation context (Giorgini et al., 2002).

In particular Tropos (Castro et al., 2002) (Bresciani et al., 2004), an agent-oriented software development framework, adopts i* to support the initial phases of software development lifecycle. Indeed, actors in i* can be autonomous, as advocated by agent-oriented software technologies (Yu, 2001), as well as intentional, since it has desires and beliefs. Nowadays, we are dealing with complex systems, such as agent-based ones and, consequently, the i* models can become very large, hard to read and understand.

The framework i* is a rich ontology that has many constructors and relationships. However, the amount of relationships may increase even for small examples (as show Figure 1). This occurs when the analyst goes deeper into context understanding analysis and applies this analysis introducing some new internal and external relationships in the model. Even with three or four actors, the understandability may be damaged since it is very sensible to the granularity of model details. This situation depends on the analyst and how far the model is detailed. The i* models are defined by the amount of internal relationships in the actor's boundary. Whenever the analysis of a complex domain is deepened, through the refinement of the several alternative solutions, more complex models will arise. Those are important points to deal with. Therefore, we need to investigate approaches to reduce the i* models complexity and to improve their reusability and understandability. Although there are several

* Currently in post-doc position at Univ. Nova de Lisboa, PT

variants of the original i* (Yu, 1995), tailored to support many Tropos versions (Bresciani et al., 2004), (Susi et al., 2005), (Bertolini et al., 2005), they are not concerned with the reduction of complexity in i* models. In this paper, we propose to use two new structuring mechanisms: one to help to hide details of alternatives (i.e. different means to achieve an end) present in the model; and the other to cope with the order of the operationalization of actors' intentions. Thus, we extend the i* metamodel by adding new constructs tailored to increase the structuring power of i* models. Our goal is to introduce a new visualization mechanism to improve the readability and understandability of i* models. In doing so, we may help to reduce the size and complexity of these diagrams, facilitating their analysis. To illustrate our proposal, we use the Health Care example presented in (Yu, 1995).

This paper is organized as follows. Section 2 overviews the i* framework. Section 3 presents our proposal for extending the i* modelling language. Section 4 discusses the benefits of using the extended i* models, while Section 5 describes some related works. Finally, Section 6 summarizes our work and points out open issues.

2 THE i* FRAMEWORK OVERVIEW

2.1 The Health Care Example

To illustrate our proposal throughout the paper, let us consider the health insurance domain already modelled using the i* notation in (Yu, 1995). In this domain, medical costs are covered by an insurance company in return for Premium payment. Treatment by a physician must be pre-approved for a physician to receive reimbursement. A claims manager issues approval by verifying that the patient's policy is applicable to the medical condition, and by confirming that the treatment plan is appropriate according to medical opinion. Patients pay insurance because they want their medical expenses to be covered in case of sickness or injury. Physicians submit treatment plans to insurance companies for approval because they want to be reimbursed for giving the treatment. Claims managers seek medical assessment of treatment plans because they want to prevent unnecessary treatments, in order to control costs. This kind of deeper understanding about the "whys" constitutes an important part of the knowledge about a domain and can be captured by using the concepts and models provided by the i* framework, since it defines a richer ontology that recognizes motivations, intentions and rationales beneath the surface features of a process.

2.2 Main Concepts and Models of i*

The i* framework (Yu, 1995) captures organizational requirements using the strategic relationships among actors. Systems and their environments are described in terms of intentional relationships among strategic actors. Actors are intentional since they have desires and needs, and are strategic since they are concerned about opportunities and vulnerabilities.

The i* framework offers two models: the Strategic Dependency (SD) model and Strategic Rationale (SR) model. The SD model includes a set of nodes and links connecting them, where nodes represent actors (depender and dependee) and each link indicates a dependency between two actors (dependum).

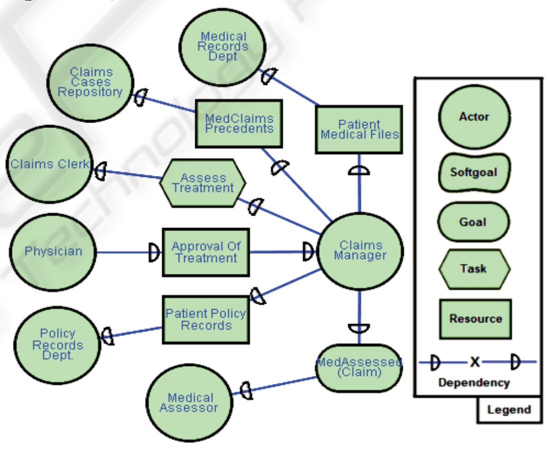

Figure 1: The Strategic Dependency Model for Claims Manager Actor.

An example of a SD model for the health care domain (Figure 1) focuses on the Claims Manager actor and its strategic dependencies in the insurance company. It shows (some of the) relationships among Claims Managers, Claims Clerks, Claims Repositories, Physicians, Departments and Medical Assessors. Physicians depend on Claims Managers to *Get Approval Of Treatment* (a task dependency), while Claims Managers, in turn, depend (i) on Medical Records Department to provide *Patient Medical Files* (a resource dependency), (ii) on Claims Clerk to *Assess Treatment* (a task dependency), (iii) on Claims Cases Repository to get *Medical Claims Precedents* (a resource dependency),

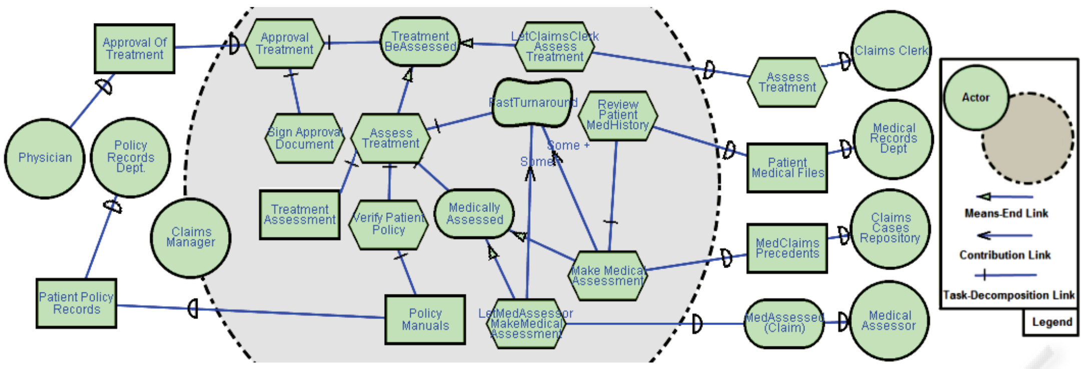

Figure 2: The Health Care Strategic Rationale Model.

(iv) on Policy Records Department to get *Patient Policy Records* (a resource dependency) and, (v) on Medical Assessors to have *Medical Claim Assessed* (a goal dependency)

In this example we do not have a softgoal dependency which can be associated to a non-functional requirement.

The SR model is used to expand the description of a given actor (e.g. Claims Manager actor in Figure 1). Apart from the previous four types of dependencies, three new types of relationships are incorporated in the SR model: (i) *task-decomposition* links describe what should be done to perform a certain task (e.g. *Approval Treatment* task); (ii) *means-end* links suggest that one model element (e.g. *Let Claims Clerk Assess Treatment* task) can be offered as a means to achieve another model element (*Treatment Be Assessed* goal); (iii) contributions links suggest how a task (e.g. *Make Medical Assessment*) can contribute to satisfy a softgoal (e.g. *Fast Turnaround*).

The SR model (Figure 2) captures some of the rationales involved in an insurance claims setting. A Physician must submit a treatment plan to the Insurance Company for prior approval, or else the treatment may not be reimbursed. The Insurance Company verifies that the type of treatment is covered by the policy, and that the proposed treatment is reasonable according to medical opinion.

The SR model shows that the Claims Manager is able to produce *Approval Of Treatment* (a resource) via the *Approval Treatment* task. This task is decomposed into of two components – the subgoal *Treatment Be Assessed*, and the subtask of *Sign Approval Document*. Originally, the task-decomposition link does not explicitly define the order in which its components are required. However, often this ordering information facilitates its understandability.

One way to have the treatment plan assessed is to let a claims clerk do it. Another way is for the Claims Manager to do it herself. This alternative requires the Claims Manager to verify the policy (that the medical condition and the treatment plan are covered under the patient's policy, and that the policy is in force) and also to have the treatment plan assessed for its medical appropriateness, producing the treatment assessment. Thus, the Medically Assessed goal can be achieved by relying on someone with special medical knowledge (a medical assessor) to do it, or by doing it herself, making use of case knowledge about previous claims, from a repository. These alternatives are explicated by the means-end link.

From this simple example, we can identify two problems which can compromise the readability and scalability of the model: the absence of ordering information in the operationalization of the tasks as well as the complexity which arises when several alternatives (with different extent of intentional elements) are taken in consideration. In fact, empirical evaluation has indicated that there is a lack of modularity in the i* framework (Estrada et al., 2006). Moreover, currently, only two views are supported, the Strategic Dependency (SD) Model and the Strategic Rational (SR) Model. Therefore, as systems models evolve and become larger, better information hiding and structuring mechanisms are needed.

Next section presents our proposal to extend the i* modelling language by adding new concepts to help us in the analysis of the models produced.

3 THE EXTENDED i* MODELLING LANGUAGE

Metamodels are models that describe the structure of models. They define the constructs of a modelling language and their relationships, as well as constraints and modelling rules. Although a version of the i* metamodel has been specified in (Yu, 1995) using Telos (Mylopoulos, 1990), we specify it now using the Meta-Object Facility (OMG, 2002). The idea is to incorporate new abstractions to hide detailed information (in general related to alternatives to reach a goal) and therefore, produce simplified model, as well as to order the operationalization of actors' intentions.

According to the metamodel introduced in Figure 3, i* models are composed of at least one Node, which can be a Dependum or a DependableNode. The DependableNode is specialized in an Actor and an InternalElement. This means that an Actor (or its internal InternalElement) can depend on an InternalElement inside of another Actor (or the Actor itself). An InternalElement can be specialized into an Alternative or an IntentionalElement, as well as can be related to IntentionalElements through a MeansEndLink, a ContributionLink or a TaskDecompositionLink.

Observe in the metamodel that the new i* language constructs are the InternalElement, the Alternative and two new attributes in the TaskDecompositionLink. An InternalElement is any element inside the boundaries of an Actor or Alternative. An Alternative is a metaclass added to allow the suppression of a subgragh composed of at least one InternalElement and in charge of the operationalization of an InternalElement through a MeansEndLink relationship. A sub-element related to a task through TaskDecompositionLink relationships can be ordered by using the optional *priority* attribute, which defines its order of execution. Elements with the same priority number can be executed in parallel. A redundant attribute *parallel* is used to explicitly indicate this parallelism. The other metaclasses in the metamodel belongs to the original i* framework (Yu, 1995).

It is important to highlight the distinction between abstract and concrete syntax of a language.

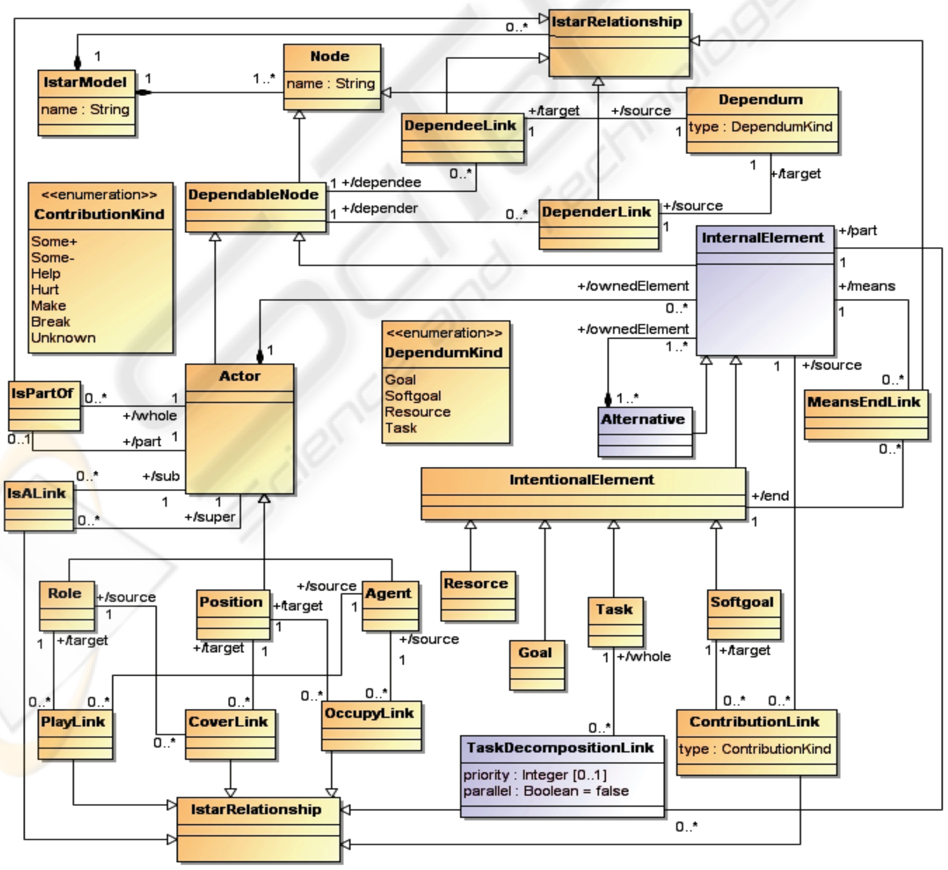

Figure 3: The Extended i* Metamodel.

The abstract syntax of a language is defined by its metamodel, while the concrete syntax of a language defines the concrete form of the textual or graphical representation of the constructs and is not defined in the metamodel. The concrete syntax of the language define by the extended metamodel is presented in Figure 4.

An Alternative is graphically represented by a diamond (Figure 4), which can be expanded or contracted to provide specific views of an actor's rationale and, consequently, produce simpler SR models. In the concrete syntax, the parallelism is represented by || symbol. We can have sibling elements with and without priority labels. Those without priority labels have the lowest priority. Some restrictions apply to priorities: (i) the number 1 (one) has the highest priority; (ii) the priority number must be continuous and sequential. (iii) softgoals do not have priority labels, because they are not directly operationalized. We emphasize which for each new task to be decomposed, the labels in the related task-decomposition links must be re-initialized.

Note that the elements *Treatment Be Assessed* goal and *Sign Approval Document* task have, respectively, priority labels 1 and 2. Thus we initiate the analysis by the left branch (priority 1). Going down at the graph analysis we can see that the elements *Verify Patient Policy* task, *Medically Assessed* goal and *Treatment Assessment* resource have, respectively, priority labels 1, 2|| and 2||. The last labels indicate that these elements are in parallel. Furthermore, as explained before, softgoals do not have priority labels.

The use of priority labels enables us to visualize and analyse, in a better way, the order of execution of the elements. This adds a temporal behaviour to elements in SR models facilitating the analysis of the execution flow of a task.

4 DISCUSSION

An important concept defined in the context of SR models is the notion of routine (Yu, 1995). A routine is an interconnected collection of intentional elements (subgoals, subtasks, resources and softgoals) serving to some purpose for an actor.

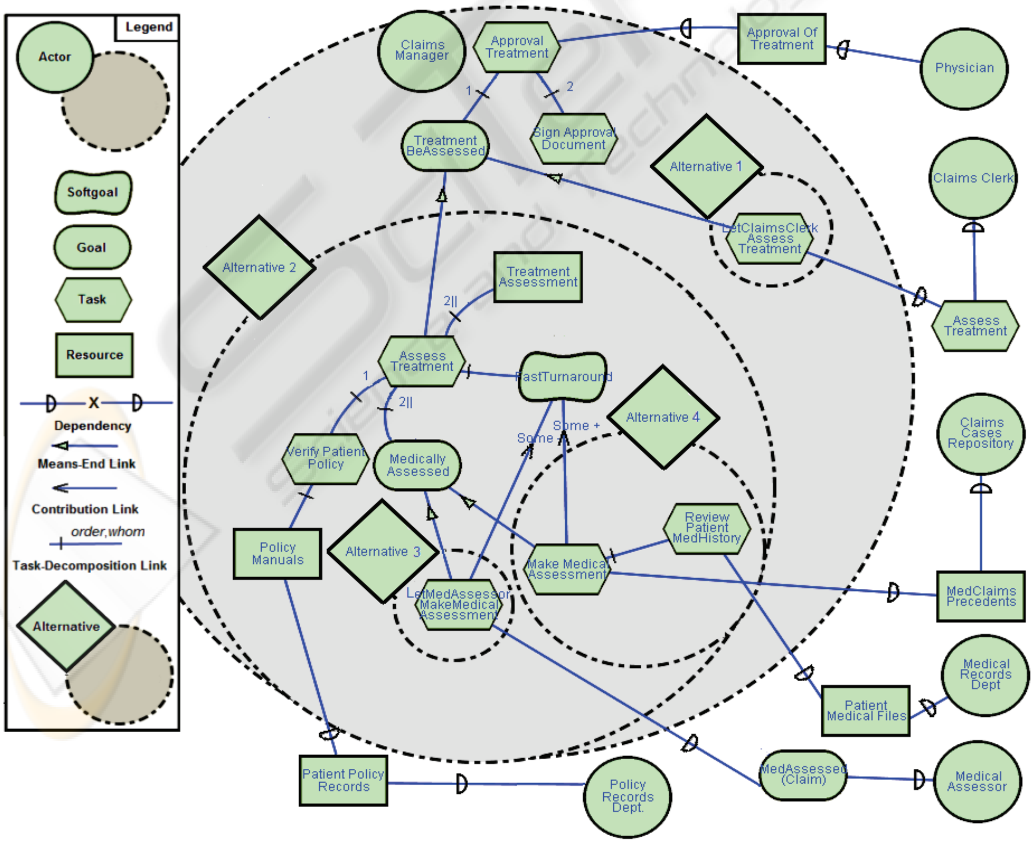

Figure 4: The Extended i* Modelling Language.

In other words, a routine is a subgraph in the SR model with a single link to a "means" node from each "end" node in a means-end relationship. Therefore, it represents one particular course of action among the multiple alternatives to accomplish an intentional element (Yu, 1995). However, in the case of softgoals, in which means-end relationships represent partial contributions of tasks or softgoals to achieve a specific softgoal, a routine will include multiple means-ends links contributing to softgoals.

The SR model explicitly enumerates an actor's set of routines. An actor often has more than one routine for accomplishing something, providing a convenient unit for analysis when evaluating alternatives. Thus, to simplify the visualization of i* models, now we are able to focus on one routine (to achieve a specific dependency) each time. A routine refers to one process and its rationales. One example of a routine is presented in Figure 5, in which the view of the SR model suppresses the Alternative 2 and focuses on Alternative 1 to achieve the *Treatment Be Assessed* goal. In this case, the routine to accomplish the *Approval Of Treatment* task dependency is the subgraph that includes *Let Claims Clerk Asses Treatment* and *Sign Approval Document* tasks.

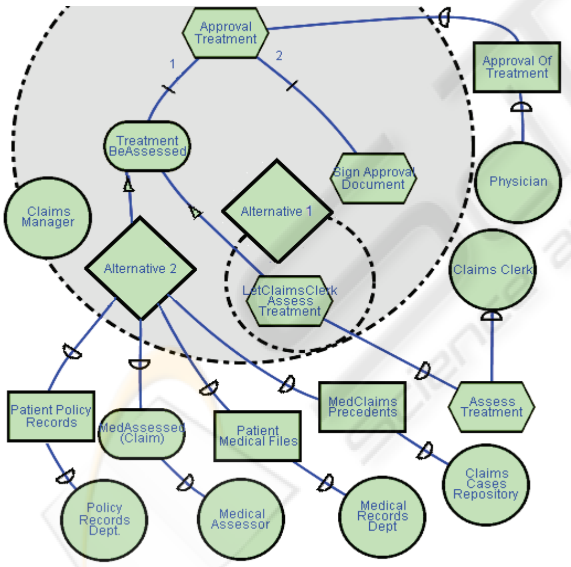

Figure 5: First view of the Claims Manager SR Model

Another routine to accomplish the *Approval Of Treatment* task dependency is presented in Figure 6 (a), in which Alternative 1 and Alternative 4 are suppressed and the attention is paid to Alternative 2 and Alternative 3. In this view, the routine considered is the subgraph including *VerifyPatientPolicy*, *ReviewPatientMedHistory* and *SignApprovalDocument* tasks.

The last possible routine to accomplish the *Approval Of Treatment* task dependency is shown in Figure 6 (b). In this view, the Alternative 1 and Alternative 3 are suppressed and the Alternative 2 and Alternative 4 are focused. Therefore, this routine involves the subgraph including *Verify Patient Policy*, *Let Medical Assessor Make Medical Assessment* and *Sign Approval Document* tasks.

Notice also that the extended modelling language now allows the ordering of the sequence of execution of sub-elements in task decompositions. An attribute in the task-decomposition link states the priority of the operationalization of a sub-element to accomplish the decomposed task. For example, in Figure 5, the *Treatment Be Assessed* goal must be operationalized before all sub-elements (priority 1), while the *Sign Approval Document* task must be executed latter (priority 2). If the *priority* property is empty, then the order of the sub-element operationalization does not matter. This is the case of *Fast Turnaround* softgoal in Figure 6 (a). Moreover, we can also specify sub-elements which must be operationalized in parallel, such as *Medically Assessed* goal and *Treatment Assessment* resource. The sub-elements' ordering in a task decomposition is essential in the analysis of the process being modelled to help decision making among alternatives.

5 RELATED WORK

The proposal presented in (Alencar et al., 2007) uses the principles of Aspect-Oriented Software Development to simplify i* models. They propose an approach to identify, modularise and compose crosscutting concerns in i* models. In doing so, they aim at reducing the graphical complexity of i* models, especially large i* descriptions. However, they do not deal with the different views of i* models, neither with the sub-elements' ordering in task decompositions.

The approach proposed in (You, 2004) introduces systematic methods to deal with scalability issues of i* models. To accomplish it, he used views (a projection over a model according to some criteria) as a way to break down one baseline model into self-contained segments in order to increase human understandability.

In the approach four types of views where introduced: Actor Class (AC), Strategic Dependency (SD), Strategic Rationale (SR) and Evaluation

Results (EVLR). The AC focus on various forms of actors and the associations among the different forms of each actor. The SD model focuses on inter-actor dependency relationships. SR view focus on “the rationales that actors have about adopting one configuration or another” (Yu, 1995). The EVLR helps the decision-making process over alternative system configurations. Each view is associated with a formally defined *selection rule* so that the projection of a specific view from a baseline model can be automated.

Although that approach increases the number of views produced in the i* framework, the complexity of the SR model, as well as the ordering of sub-elements in a task decomposition have not been addressed.

On the other hand, in our proposal, the issue of ordering the operationalization of the sub-elements of a task is addressed by extending the i* modelling language itself. In fact, we have extended the i* metamodel to allow the addiction of two properties in the task-decomposition link: one to represent the order in which the sub-element must be operationalized; and other to state if its operationalization can be in parallel with another sub-element.

In doing so, we did not need to use diagrams of other modelling languages, such as UML (OMG, 2005), to improve the understandability of the i*

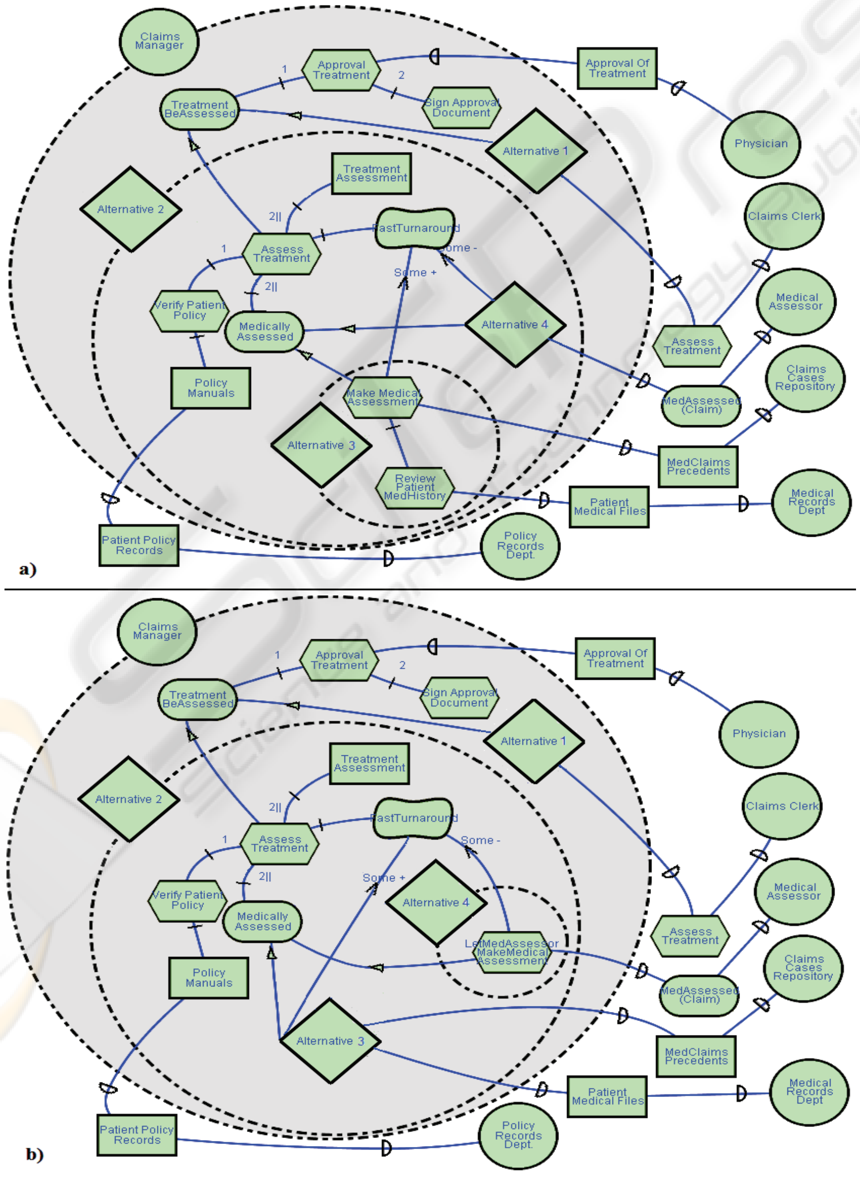

Figure 6: a) The Second View of Claims Manager SR Model; b) The Third View of the Claims Manager SR Model.

models. Moreover, the extended i* metamodel also defines a new abstraction (the alternative) to allow the generation of several different views of the same model. In particular, each view focuses in a particular routine to accomplish something, reducing the complexity of the model and, therefore, improving its readability and understandability.

6 CONCLUSIONS

In this paper we propose an extension of the i* modelling language by adding new constructs to the i* metamodel to improve the understandability of the i* models. By doing so, we have added two attributes to the task-decomposition link to define the order in which the components of a decomposed task are required. Moreover, a new abstraction has been added to the metamodel, called alternative, to suppress specific sub graphs in the SR model and, therefore, produce simpler views of the same SR model. In fact, each view focuses on a specific routine to achieve a dependency. These views may reduce the complexity of i* models, increasing their readability, maintainability and scalability. In fact, it is now necessary to carry out experiments with some available metrics (Ramos et al., 2008) (Franch, 2006) to estimate the degree of amount complexity reduction using our approach.

Currently, we are working on the inclusion of routines identification and their association with viewpoints (Sommerville et al., 1998) in i* models. Finally, we plan to evolve the current i* support tool (Yu and Yu, 2000) to include the new constructs of the extended metamodel. Moreover, the new constructs requires the formal specification of constraints to be considered when specifying i* models. Currently, these constraints are being specified in OCL (OMG, 2006) and will be presented in future work.

ACKNOWLEDGEMENTS

This work was supported by CAPES/GRICES Proc. 129/05.

REFERENCES

Alencar, F., Castro, J., Moreira, A., Araújo, J., Monteiro, C., Ramos, R., Mylopoulos, J., 2007. Simplifying i* Models. In *AOIS'07 in conjunction with CAiSE'07*. Tapir Academic Press, Norway. 635-649.

Bertolini, D., Perini, A., Susi, A., Mouratidis, H., 2005. The Tropos Visual Language. A MOF 1.4 Compliant Metamodel. *Agentlink III AOSE TFG 2*. Slovenia.

Bresciani, P., Giorgini, P., Giunchiglia, F., Mylopoulos, J., Perini, A., 2004. Tropos: An Agent-Oriented Software Development Methodology. *Journal of Autonomous Agents and Multi-Agent Systems*, 8(3): 203-236.

Castro, J., Kolp, M., Mylopoulos, J., 2002. Towards Requirements-Driven Information Systems Engineering: The Tropos Project. In *Information Systems News*, Elsevier, 27: 365-89.

Estrada, H., Rebollar, A. M., Pastor, O., Mylopoulos, J., 2006. An Empirical Evaluation of the i* Framework in a Model-Based Software Generation Environment. In *CAiSE'06*. LNCS 4001, Springer-Verlag , 513-527 .

Franch, X., 2006. On the Quantitative Analysis of Agent-Oriented Models. In *CAiSE'06,* LNCS 4001 Springer-Verlag: 495–509.

Giorgini, P., Mylopoulos, J., Nicchiarelli, E., Sebastiani, R., 2002. Reasoning with Goal Models. In *Proc. of the 21st Int. Conference on Conceptual Modelling*. LNCS 2503. Springer-Verlag, London, 167-181.

Mylopoulos J., Borgida A., Jarke M., Koubarakis M., 1990. Telos: Representing Knowledge About Information Systems. *ACM Transactions on Information Systems*, 8(4): 325–362

Object Management Group (OMG), 2002. Meta-Object Facility (MOF) Specification, 1.4. Available at: http://www.omg.org/cgi-bin/apps/doc?formal/02-04-3.pdf. Last access: 11/2007.

Object Management Group (OMG), 2005. Unified Modelling Language (UML) Superstructure, 2.0. Available at: http://www.omg.org/docs/formal/05-07-04.pdf. Last access: 09/2007.

Object Management Group (OMG), 2006. Object Constraint Language (OCL) Specification, 2.0. Available at: http://www.omg.org/cgi-bin/apps/doc?formal/06-05-01.pdf. Last access: 11/2007.

Ramos, R., Castro, J., Araujo, J., Moreira, A., Alencar, F., Penteado, 2008. R. Early Aspects Refactoring. In *Proc. of the XI IDEAS'08*. FASA: Recife-PE, 238-252.

Sommerville, I. Sawyer, P. Viller, S., 1998. Viewpoints for requirements elicitation: a practical approach. *Proc. of RE'98*, 74—81.

Susi, A., Perini, A., Mylopoulos, J., Giorgini, P., 2005. The Tropos Metamodel and its Use. *Informatica.* Slovenia, 29(4): 401-408.

You, Z., 2004. *Using Meta-Model-Driven Views to Address Scalability In i* Models*. MSc thesis, Department of Computer Science, University of Toronto, Canada.

Yu, E., 1995. *Modelling Strategic Relationships for Process Reengineering.* Ph.D. thesis. Department of Computer Science, University of Toronto, Canada.

Yu, E., 2001. Agent-Oriented Modelling: Software Versus the World. In *AOSE'01*, LNCS 2222, Canada, 206-225.

Yu, E., Yu, Y., 2000. Organization Modelling Environment. At: http://www.cs.toronto.edu/km/ome/ Last access: 12/2007.