JGOOSE: A REQUIREMENTS ENGINEERING TOOL TO INTEGRATE I* ORGANIZATIONAL MODELING WITH USE CASES IN UML

JGOOSE: UNA HERRAMIENTA DE INGENIERÍA DE REQUISITOS PARA LA INTEGRACIÓN DEL MODELADO ORGANIZACIONAL I* CON EL MODELADO DE CASOS DE USO EN UML

André Abe Vicente[1] Victor F. A. Santander [2,4] Jaelson B. Castro [3] Ivonei Freitas da Silva[4] Francisco G. Reyes Matus[2]

Recibido el 10 de enero de 2007, aceptado el 11 de marzo de 2009
Received: January 10, 2007 Accepted: March 11, 2009

RESUMEN

En los días actuales los sistemas computacionales se caracterizan por su complejidad, dinamismo y gran importancia estratégica. En este complejo escenario de especificación de software, generar documentación de alta calidad es una difícil tarea. En general los clientes no saben exactamente lo que desean y muchas veces los requisitos del software no reflejan las reales necesidades de los clientes y del ambiente organizacional. Es bastante común encontrar requisitos inconsistentes e incompletos. En este contexto, uno de los grandes desafíos está en la necesidad de integrar los requisitos organizacionales y funcionales del sistema computacional que será desarrollado. En este trabajo se presenta la herramienta computacional JGOOSE (*Java Goal Into Object Oriented Standard Extension*) que permite integrar diagramas de casos de uso en UML con requisitos organizacionales representados utilizando la técnica i*. Se presenta la utilización de la herramienta en el caso de estudio de un Sistema de Gestión de Evento Científico.

Palabras clave: Casos de uso, modelado organizacional, soporte computacional.

ABSTRACT

Nowadays Computational Systems are being characterized by their complexity, dynamism and great strategic importance. In this complex context of software specification, generating high quality documentation is very difficult. Usually, clients are not sure about their needs and sometimes software requirements do not represent clients and organizational environment needs. Inconsistent and incomplete requirements are very frequent. One of the challenges to solve this problem is to integrate organizational and functional requirements of the system to be developed. In this paper the JGOOSE (Java Goal into Object Oriented Standard Extension) tool used to assist requirement engineers in the development of use cases from the organizational models represented by i technique is presented. To validate the tool, it was used and applied to the Conference Management System case study.*

Keywords: Use cases, organizational modeling, computational tool.

INTRODUCTION

The detection of most software project problems takes place on the initial steps of modeling software systems. These initial steps are called requirements engineering process (RE) and the main activities of this process can be defined as elicitation, analysis, negotiation, specification, management and requirement validation [23].

Understanding the necessity and meeting customers goals have always been one of the biggest challenges of Software Engineering. Requirement Engineering focus

[1] Instituto de Ciências Matemáticas e Computação. Universidade de São Paulo (USP). Av. Trabalhador São-carlense, Nº 400. São Carlos-SP, Brasil. E-mail: avicente@icmc.usp.br
[2] Universidad de Talca. Facultad de Ingeniería. Curicó,Chile. E-mail: vsantander@utalca.cl, freyes@utalca.cl
[3] Centro de Informática. Universidade Federal de Pernambuco (UFPE). Recife - PE, Brasil. E-mail: jbc@cin.ufpe.br
[4] Universidade Estadual do Oeste do Paraná (UNIOESTE). Cascavel Paraná. Brasil. E-mail: ifreitas@unioeste.br

on proposing methods, techniques and tools that help comprehension and requirement registration process that the software must accommodate. Different from other software engineering sub-areas, the requirements area must deal with the knowledge involving interdisciplinary, social science and cognitive science aspects [13].

The main reason for failures in the RE is the lack of an adequate organization understanding by software systems developers, also caused by the frequency that organizational changes occur, and the changes which can not be accommodated by existing software systems [1]. During the requirements analysis phase, analysts with the stakeholders help, need to identify the organizational goals and the functional and non-functional requirements associated with the computer system being developed. Consequently, requirements engineer should examine and model the stakeholders' interests, and how they should be seen or negotiated, by various alternative organizational systems structures and environments. However, the production of quality specifications is relatively difficult.

In this context, it is important to highlight Eric Yu's propose [25], which expresses the importance of separating the early and late requirement elicitation phases. Early requirement phase activities are typically informal, and usually describe functional, organizational and non-functional requirements. This phase emphasizes the understanding of motivations and reasons that support system requirements. Late requirement phase activities are usually focused on the completeness, consistency and automated verification of requirements.

Framework i*, proposed by Yu [25] allows intention, relationship and motivation modeling among organizational members. From these models you can have a better understand of organizational environment functioning, human and work relations among the organization participants. With this information, the computational requirements solution for the organizational processes can be better elicited and specified.

However, actually the most used methodologies to developed computational systems are object oriented methodologies. Object oriented software have been developed using UML (Unified Modeling Language) [5] which is a broad format and well defined to modeling systems for oriented objects development, providing viewing patterns mechanisms, specification, development and software systems documentation.

On the other hand, most modern computational systems have been developed by using object oriented methodology. Object oriented software have typically been developed by using UML, which is a broad, well defined modeling system for object oriented developments, supplying viewing, specification, and construction mechanism patterns, and software system documentation.

The UML generally employs use cases to describe in a clearly and consistently way what the system should do through interactions between users and software systems. Use Cases are responsible for the decision and description of system functional requirements. However, the success of the software project depends primarily on a good understanding of the company`s organization environment and its processes in terms of objectives, business rules, tasks, resources and the relationships among its actors. In the context of requirements engineering, these aspects can be translated as the system organizational requirements or initial requirements (early requirements), and the Unified Modeling Language does not have adequate mechanisms for modeling these aspects.

In many studies [1, 6, 8, 19-21] the authors argue that UML support only the capture phase of final requirements. Even using mechanisms like business use cases, previous studies state that UML can not show how the system reaches its desired organizational goals, the motivations to develop the system, which alternatives are considered and alternative consequences to the stakeholders. Therefore, we concluded that we must use another technique like i* to represent such aspects, which focus is to describe not only the organizational relationships between the various organizational actors, but also the understanding of the reasons involved in the decision process.

Thus this paper proposes integration between i* framework and the object oriented modeling using the UML modeling language. This integration process is based on many researches, among them [1]. These researches not only support this integration, but propose guidelines for the use of mapping technology to integrate i* and UML use cases [20 - 21] or UML class diagrams [8, 18-19]. Moreover, some of these papers propose integration in a semi-automated way through the implementation of tools [2, 18] that make this process easier.

This paper briefly presents mapping guidelines proposed in [19-20], which allow deriving UML use cases from the organizational models developed with i* framework. In addition, it's described a software tool called JGOOSE (Java Goal into Object Oriented Standard Extension) which was developed in this research to provide semi-

automatic support of these guidelines. To validate the software tool, a Conference Management System Case Study is presented.

The paper is organized as follow: in the section "EARLY REQUIREMENTS CAPTURE USING I* TECHNIQUE", the use of i* framework to capture the early system requirements is described. In the next section, an overview of the proposed mapping of use cases from i* organizational models is presented, together with guidelines that propitiate this mapping. In "JGOOSE TOOL" describes the tool developed to support this mapping and then, a case study using the tool is presented. Finally, the final paper considerations and future effort are made.

EARLY REQUIREMENTS CAPTURE USING I* TECHNIQUE

The i* framework was developed to support the analysis process and conceptual modeling, under a strategic (motivations and reasons) and intentional vision of processes that involve many participants called *Actors*. These *Actors* depend on one another to reach *objectives/goals*, execute tasks and provide *resources*. Different from other modeling techniques, which typically describe a process in terms of activity and flow stages between entities, i* technique is distinguished for being concerned on the reasons or motivations that are associates to the behavior aspects of the process. In general, traditional modeling techniques allow the description of component system (actors) in terms of its state, capacity (processes that can execute) and behavior (how and when the processes are executed), but they can not express the reasons involved in the processes (the motivation).

Further than assisting the requirement engineering process in its initial specification phase, the i* technique can be applied in other areas [25] such as, business re-engineering process, analysis of organizational impacts, software processes modeling, and also for the development of agent-oriented systems [7, 21].

Two models are considered to describe these areas: the Strategic Dependency Model **(SD)** and the Strategic Rationale **(SR)** model. The Strategic Dependency Model is composed of **Nodes** and **links**.. Knots represent the **actors** in the environment and the links represent the **dependency** between the actors. The actor is an entity that carries an action to fulfill a goal, in the organizational environment context. **Actors** on one another to reach objectives, carry through tasks, and get resources in the organizational environment. The actor who depends on any form of another actor is called ***Depender*** and the actor who answers and satisfies the ***Depender*** is called of ***Dependee***. The object or element of dependence between ***Depender*** and ***Dependee*** is called of ***Dependum***. Therefore there will be a relationship of the type ***Depender*** → ***Dependum*** → ***Dependee***. The Strategic Rationale Model is a complementary model to the Strategic Dependency model. This model allows understanding in a detailed form, the reasons between each actor and its dependences.

While the Strategic Dependency Model provides an abstraction level, in which it only models the external relationships between actors, the Strategic Rationale Model allows a bigger understanding regarding the actor's strategic reasons in relation to the organization processes and how to express it. The model of Strategic Reasons assists the Requirements Engineering process, allowing that process elements and the reasons that ones express. In the Requirement Engineering, the Strategic Rationale Model can be used to understand how systems are related/involved in actors routines of the organization to generate alternatives, as well as model and support the reasoning of organizational actors regarding these alternatives.

While the Strategic Dependency Model provides an abstraction level, in which it only models the external relationships between actors, the Strategic Rationale Model allows a bigger understanding regarding the actor's strategic reasons in relation to the organization processes and how they are expressed. The model of Strategic Reasons assists the Requirements Engineering process, allowing the expression of process elements and the reasons supporting them. In Requirement Engineering, the Strategic Rationale Model can be used to understand how systems are related/involved in actors routines of the organization to generate alternatives, as well as model and support the reasoning of organizational actors regarding these alternatives.

Figures 1 and 2 present the use of the i* technique through the Conference Management System modeling. This model was introduced in [27], and it was modeled through the Tropos framework in [22]. The models SD and SR on figure 1 and 2, respectively, represent the external dependencies between the organizational actors, as well as the internal strategic reasons of these actors, in relation to the involved processes in **a** conference management that will use a software called Conference Management System to administer the article submission process and publication of proceeding events. Figures 1 and 2 present a correspondent number for easier reference to these elements throughout the text.

Several scientific events (conferences, congresses, symposiums, meeting…) frequently congregate researchers and practitioner communities of the area, having as main focus the spreading of work results that involve topics related to the event, offering activities as lectures, tutorials, mini-courses and presentation of techno-scientific articles. Generally, each activity is supervised by a specialized program committee who is responsible for the critical selection of work to be presented. In this manner, we consider this conference management application domain.

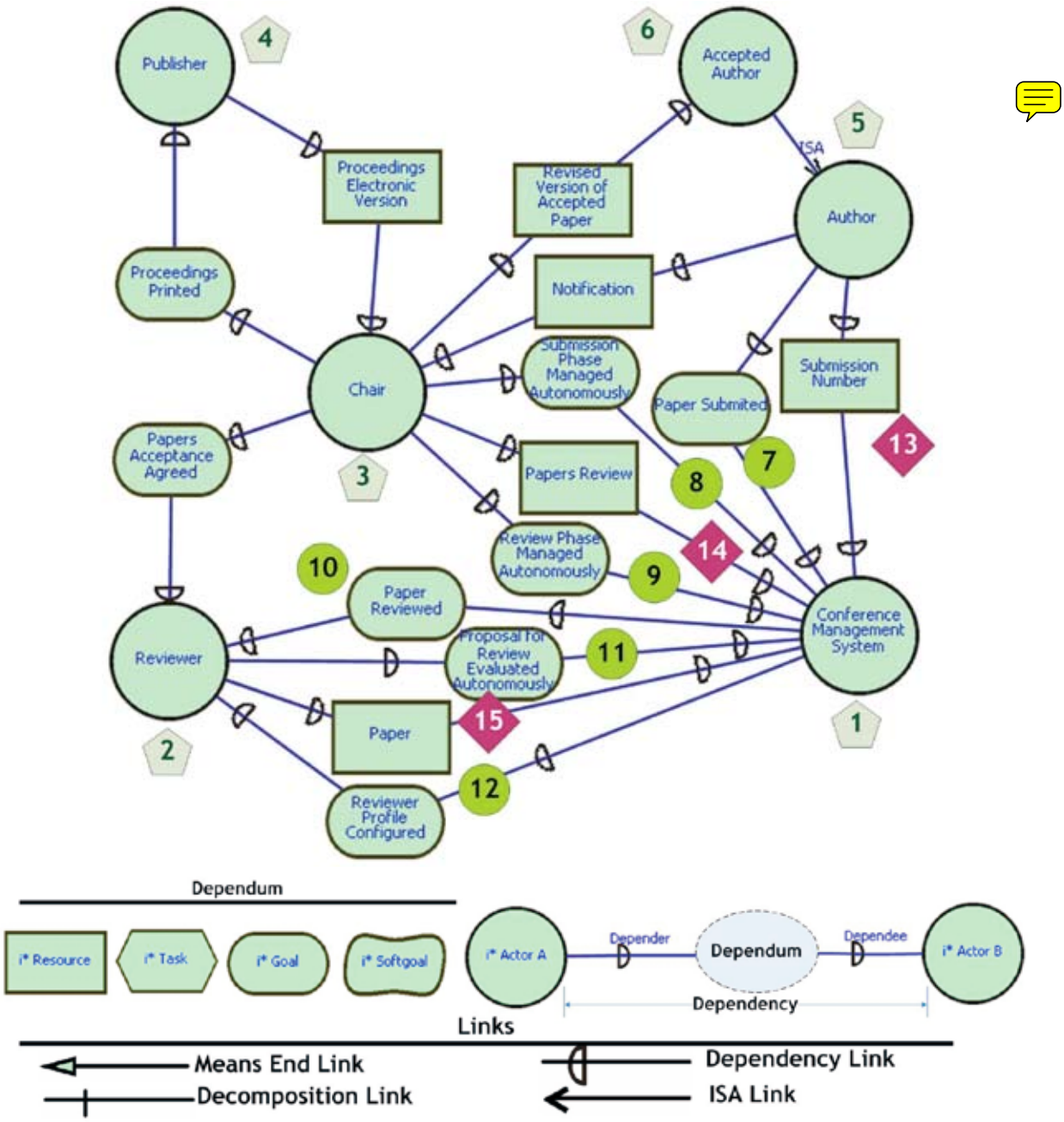

Figure 1. Strategic Dependency Model – Conference Management System – Early Architecture Release (Adapted from [2]).

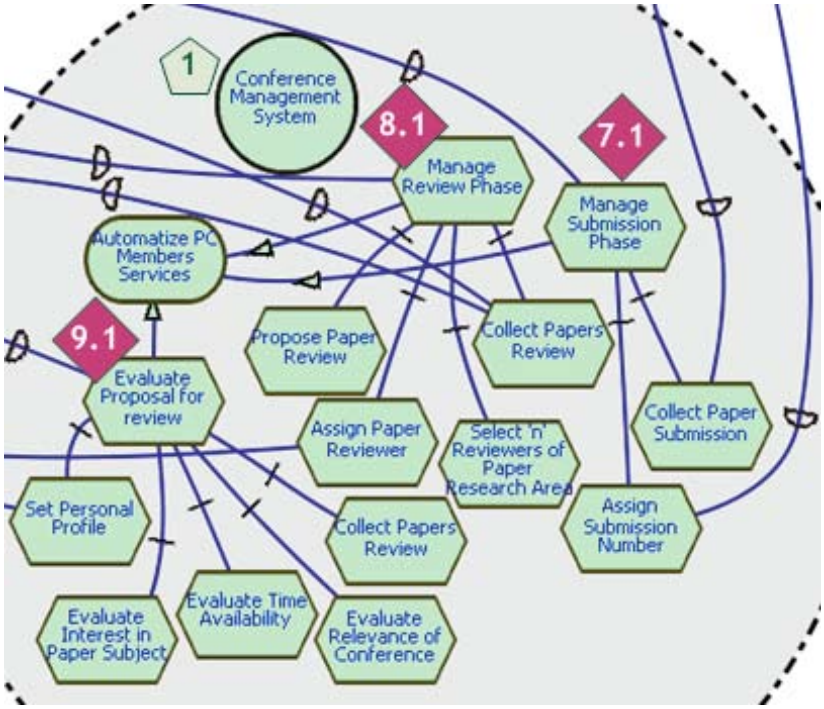

Figure 2. Strategic Rationale Model-Conference *Management System* - Early Architecture Release (Adapted from [2]).

In figure 1 it can be observed that a conference involves many individuals. During the submission phase, Authors (*Author* **(5)** actor) submit papers are informed if their articles were received and a *submission identification number* **(13)** is issued by the system. In the revision phase, the committee (*Chair* **(3)** actor) must deliver to the article revision contacting potential reviewer (*Reviewer* **(2)** actor) and asking them to revise a number of articles according to their expertise. Eventually, revisions are requested and generally they decide on the acceptance or rejection of the submissions. In the final phase, the Authors (*Author* **(5)** actor) need to be notified of the decisions and, in case of acceptance of the article, production and submission of the revised version of their articles will be requested. The publisher (*Publisher* **(4)** actor) must collect the final versions and print the event proceedings.

In spite of supplying tips on why the processes are structuralized in a certain way, the Strategic Dependency Model (SD) (illustrated in figure 1) does not support enough elements to suggest, explore and evaluate alternative solutions. As the analysis process continues, additional responsibilities for the Conference Management System are discovered. These responsibilities will be represented in Strategic Rationale Model (SR) illustrated in figure 2.

The Strategic Rationale Model (SR) of figure 2 expands the actor who represents the *Conference Management System* **(1)** delegating some tasks to it on which External actors to the system depend on these tasks. These tasks are summarized as follow:

- ***Manage Submission Phase (7.1)***: that is divided in:
  1. Collect Submission Paper
  2. Assign Submission Number
- ***Evaluate Propose for review (9.1)*** that is decomposed in five sub-tasks:
  1. *Set Personal Profile*
  2. *Evaluate Interest in Subject Paper*
  3. *Evaluate Time Availability*
  4. *Evaluate Relevance of Conference*
  5. *Collect Papers Review*
- ***Manage Review Phase (8.1)*** that is divided in:
  1. *Propose Paper Review*
  2. *Select "n" Reviewers of Paper Research Area*
  3. *Assign Paper Reviewer*
  4. *Collect Papers Review*

The initial version of the Conference Management System supports the submission, revision and notification phases of the conference management process. There is a version modeled which improves this initial version, decomposing the Conference Management System in four functions for the agents: *Submission Supervisor*, *Revision Supervisor*, *Notification* and *Copyholder Supervisor*. However, for our Case Study it was used only the early requirements

modeling represented from the Strategic Dependency diagram of the figure 1 and Strategic Rationale diagram represented by figure 2, which expands the actor who represents Conference Management System.

DERIVING UML USE CASES FROM EARLY REQUIREMENTS

The mapping guidelines proposed by Santander [19, 20] have as objective to associate i* models (SD and SR Diagrams) for UML Use Case diagrams. The initial stage of this integration process is the development of SD and SR models of the system that illustrate the early requirements of a computational system.

In general, the mapping process occurs from SD and SR organizational models, which initiates the integration and discovery process of the use cases for one target system of the organization. In the first moment **(step 1)**, the actors for the UML Use Case diagrams are discovered and later, the use cases are discovered for these actors **(step 2)**, as well as the main and alternative flows (primary and secondary scenes) of the discovered use cases **(step 3)**.

The input in this integration process is the Strategic Dependency (SD) Model for steps 1 and 2. Through SD Model it can be detected which i* actors in has a dependence relation with the System, as well as detecting which elements (goal, tasks, soft-goals and resources) lik these actors to the system. The main and alternative flows descriptions of use cases (step 3) are derived from the Strategic Rationale Model (SR). These flows derived from the internal links of the computational system, which can be divided in decomposition links and means-end links.

The used guidelines to derive use cases from the i* modeling were defined in [19, 20] and are summarized as follow:

1º Proposal step: Actor Discovery

Guideline 1: all i* actor should be considered as a possible mapping for actor in use case;
For example, the *Reviewer* Actor (2) can be analyzed - see figure 1.

Guideline 2: initially, we must analyze if the i* actor is external to the intended computational system. In case that the actor is external to the system, the actor is considered candidate actor in Use Cases;

For example, the *Reviewer* actor (2) –figure 1– is external to the *Conference Management System* (1) and can be a candidate for a Use Cases actor. The *Publisher* actor (4) has dependence relation only with the *Chair* actor (3), thus this actor is not considered a candidate Actor for a Use Case.

Guideline 3: if the actor is external to the system, it should be guaranteed that the actor in i* is a candidate actor in the Use Case diagram. For this purpose, the following analysis is necessary:

> **Sub-Guideline 3.1:** verify if there is exists at least one dependence of the actor analyzed in relation to the i* actor in i* that represents the intended computational system; The *Reviewer* actor in i* can be mapped for Use Case actor, considering its associate dependencies (e.g. *Paper Reviewed* **(10)**), that characterize its importance in the context of its interaction with the Conference Management System.

Guideline 4: i* actors in i*, related through IS-A mechanism in the organizational models and individually mapped for use case actors in use (applying guidelines 1, 2 and 3), will be related in the use cases diagram through the <<generalization>> relationship.

For example, ISA relationship between Author (5) and Accepted Author (6) in figure 1, can be mapped for a generalization relationship between these actors in the Use Cases diagram (see figure 8).

After all the actors have been discovered their respective Use Cases can be found.

2º Proposal step: Use Cases Discovery

Guideline 5: for each discovered actor in step 1, we should observe all dependences (*dependum*) of the actor view point as *dependee* in relation to the actor that represents the intended computational system (computational system → *dependum* → actor), aiming to discover use cases for the analyzed actor;

> **Sub-Guideline 5.1:** to evaluate **goal** dependences - each such dependence must directly be mapped for actor use case.
> **For example,** in figure 1, the *Reviewer Profile Configured* dependence **(12)** between *Conference Management System* **(1)** (*Depender*) and *Reviewer* **(2)** (*Dependee*) can be mapped for the Use Case *Reviewer Profile Configured*
> **Sub-Guideline 5.2:** to evaluate task dependences - each such dependence must directly be mapped for actor use case.

Sub-Guideline 5.3: to evaluate resource dependences - each such dependence must directly be mapped for actor use case.
Sub-Guideline 5.4: all the soft-goals dependences are not mapped - normally this dependence type is associated in the organizational modeling to a non-functional requirement associated with the computational system. Thus, this dependence type is mapped for a non-functional requirement of the intended system. These requirements can be visualized in step 3 of the tool (figure 6) through the button "Show NFRS".

In the example of the figure 1 we don't have a task or resource relationship and so, the sub-guidelines 5.2 and 5.3 can not be applied due to the non existence of a relation type "computational system → task/resource → actor".

Guideline 6: analyze special situations, where an actor discovered (following the step 1), has dependencies (as *depender*) in relation to an i* actor that represents intended computational system or part of it. (actor → *dependum* → computational system). These dependencies, according to [18], usually generate Use Cases, due to the fact that the *dependee* is a software system and the *depender* (Use Case actor) must interact with the system to achieve the goal associated with the generated Use Case.

For example, in figure 1, the dependence *Submission Number* (13) between *Author* (5) (*depender*) and *Conference Management System* (1) (*Dependee*) can be mapped for a Use Case *Submission Number*.

Guideline 7: classify each Use Case according to the type associated to its goal (contextual goal, user goal, sub-function goal). This guideline is based on the classification proposal by Cockburn [8].

- A **business goal** represents an high-level intention related to business processes, that the organization or users have in the organizational environment context.
- A **summary goal** represents an alternative to the satisfaction of the business goal.
- An **user goal** results in the direct discovery of an excellent and valid functionality for the actor organization using the software system.
- Finally, **sub-function level goals** are necessary to reach objective of user.

To support requirements engineers to identify new Use Cases and to have a better understand of the same ones, it is recommended to generate a table that contains the actor, its Use Case and the corresponding guideline (see table 1).

Table 1. Mapped Use Cases Example – C*onference Management* System.

Dependence Actor	Dependence	Dependence Type	Guideline
Reviewer	*Reviewer Profile Configured*	Goal	G5,G5.1
Author	*Paper Submited*	Goal	G.6

After identified all the Use Cases for the actors now it we can follow the next step for the detailed description of the same ones.

***3º Proposal step**:* Use Cases main and alternative flow description and discovery.

Guideline 8: to analyze each actor and its relationships in the SR Model to extract information that can lead to the description of main and alternative flows, as well as pre-conditions and pos-conditions of the discovered actor use cases.

Sub-Guideline 8.1: to analyze the sub-components in a task decomposition linking in a possible mapping for steps in the primary scenario description (main flow) of use cases.
Sub-Guideline 8.2: to analyze means-end links type in a possible mapping for alternative steps in the use cases description.
Sub-Guideline 8.3: to analyze the relationships of sub-components dependences in the Strategic Rationale model in relation to other actors of the system. These dependences can originate pre-conditions and pos-conditions for the discovered use cases.

For example, we can generate a Use Cases description based in *Submission Phase Managed Autonomously* dependency **(7)** (see figure 2) that occurs between the *Chair* actor **(3)** and the *Conference Management System* **(1)**. This dependency is mapped for Use Case through guideline 6. This Use Case satisfied by the task set that decomposes the *Manage Submission Phase* **(7.1)**. The *Manage the Submission Phase* task is decomposed in the tasks: C*ollect Submission Paper* and *Assign Submission Number*. These sub-tasks are mapped for the main scenario steps of the *Submission Phase Managed Autonomously* Use Case.

Guideline 9: To investigate the possibility to derive new use cases goals from the steps observations in the scenarios (events flows) of the discovered use cases. Each step of a use case must be analyzed to verify the possibility of being refined in a new use case.

A **Guideline 9** is not supported by the current version of JGOOSE tool. However, it will be implemented in posterior versions of the tool, using an interaction mechanism with the requirements engineer who makes possible the generation of new Use Cases.

Guideline 10: To develop the use case diagram using the discovered use cases, as well as observing the relationships of the << include>>, << extend>> and << generalization>> used to structure the use cases specifications.

The presented guidelines represent a systematic means to derive use cases from organizational models in i*. However, the analysis effort can significantly be reduced with the support of a computational tool. Therefore, aiming to provide a semi-automated support for these guidelines JGOOSE tool was developed and it is described in the next section.

JGOOSE TOOL

The JGOOSE tool (*Java Goal Into Object Oriented Standard Extension*) [30][29] was developed using Java 1.5 [12] and Eclipse IDE [10].

The mapping process described in the previous section was executed in JGOOSE tool from diagrams i* created by the OpenOME tool (Open Organizational Modeling Environment) [16] or OME3 [14] to UML use cases and they are showed in the tool. The main change in the JGOOSE tool comparing with its previous version (GOOSE) [3] is the new adopted systematic implementation that makes it more flexible to integrate with other tools, and significantly improves its usability. Previously the tool built the use cases diagram through the extension *Rose Extensibility Interface*, an interface that allows customization of menus, creating scripts to automate tasks and access to elements of the Rational Rose tool. However, the GOOSE development made the application dependent on the Rational Rose proprietary tool.

In JGOOSE tool, the diagram and its textual description are displayed in the tool (figure 6) and it is also stored in Java classes, where, through new versions of JGOOSE tools, there is a possibility to translate into specific UML CASE tool format or the XMI standard (XML Metadata Interchange) [26]. The XMI standard use proposed by the OMG [15] has become a widespread standard for data exchange between CASE tools, making information exchange easier between different tools through a standard flexible format that can be easily decipherable to the information.

JGOOSE tool progress regarding its previous version (GOOSE) is listed bellow:

1) **New guidelines**
   - Development of **sub guideline 5.4** that concerns about softgoals that are mapped as a possible non-functional requirement (NFRs) related to the system.
   - Development of **guideline 8.1** that concerns about softgoals that are part of tasks decomposition and are mapped as special requirements in the Use Case description.

2) **New features:** new features were designed for a clearer i*>> UML Use Case mapping process with JGOOSE. For example, the information of all elements in one i* TELOS file which would be mapped to Actors and Use Cases with its descriptions.

3) **TELOS File:** support for open files generated by OME3 and OpenOME Tool.

4) **Tool Development using Object Oriented** of Java Language. The i* objects and mapped Actors, Use Cases and relationships between then are generated with Java Classes that represents these objects in a consistent way.

5) **Use Cases Textual Description** as the Cockburn's template [9] .

JGOOSE tool use

The internal structure tool (represented by the figure 3) is basically:

(1) Specific tokens capture in a TELOS file generated by the OpenOME or OME3 tool. This information describes the objects in the i* SD / SR diagram, their elements (actors, goals, tasks, soft-goals and resources) and its links (dependency, task decomposition and mean-end links) and all the attributes that characterize these objects.

(2) Store the Diagram SD / SR using data structure: Define a data structure that stores SD / SR diagram information.

(3) Map the SD/SR Structure to a data structure that represents UML Use Cases: guidelines proposed by Santander will be used [20] (guidelines 7 and 9 weren't developed).

(4) Translate the Use Cases Structure to the XMI standard (XML Metadata Interchange): feature aimed to generate an tool output format that can be open by some UML CASE tools. This feature will be coded in future versions of the tool.

The necessary steps to map a model SD / SR stored in a TELOS file (*. tel) to an Use Cases diagram using the JGOOSE Tool are described bellow with the tool screenshots that illustrate its operation (figures 4, 5 and 6):

- **STEP 1 - Information capture of TELOS file:** The user must open a TELOS file that stores a SD/SR diagram. This command can be executed through the "Open TELOS File" button (Abrir Arquivo TELOS) or in the "Arquivo » Open TELOS File" file menu. After this step, the i* actors are showed and the user must choose one actor who will represent the Computer System. Moreover this first screen shows all Actors, Elements and mapped Links of Telos file.

- **STEP 2 – Guidelines Selection:** The guidelines *D.1 - D.4* will always be default selected. The guidelines *D.8 - D.8.3* (see section 3) may be selected only if the model selected by step 1 is a SR model, only this model provides information to find the events flow for a Use Case. In this step, the user can also read a brief tutorial about the mapping guidelines (figure 5).

- **STEP 3 – Mapped Actors, Use Cases and descriptions:** Finally on the 3rd and final stage shows mapped Actors, IS-A Actors, Use Cases, and their descriptions as well as Non-Functional Requirements (NFRs), and the guidelines that was used to map such elements. This information is displayed in the user interface (figure 6).

The following section presents the case study Conference Management System described in figures 1 and 2. This case study was aided by JGOOSE tool.

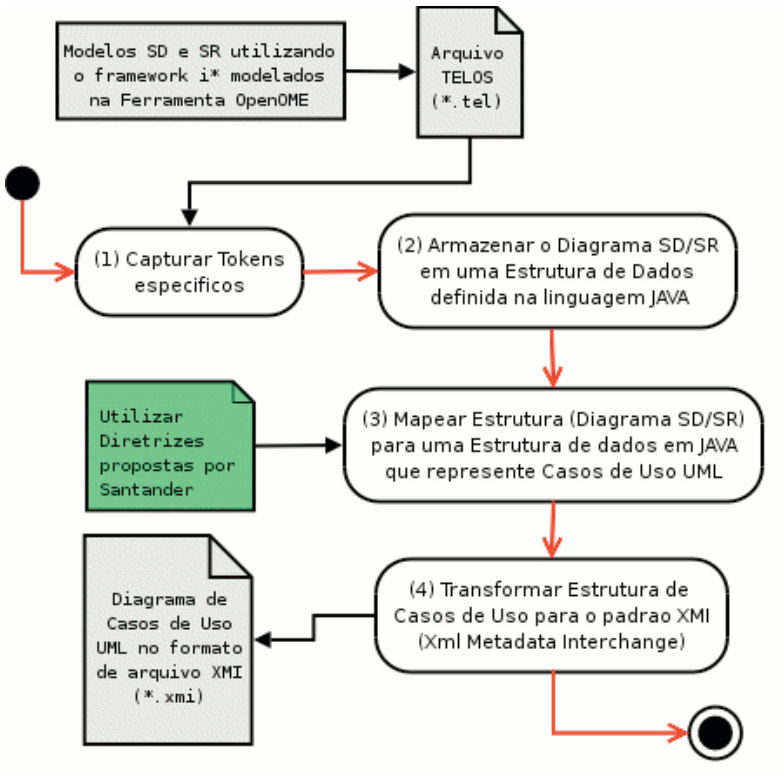

Figure 3. Internal structure of JGOOSE tool for mapping i * - UML Use Cases.

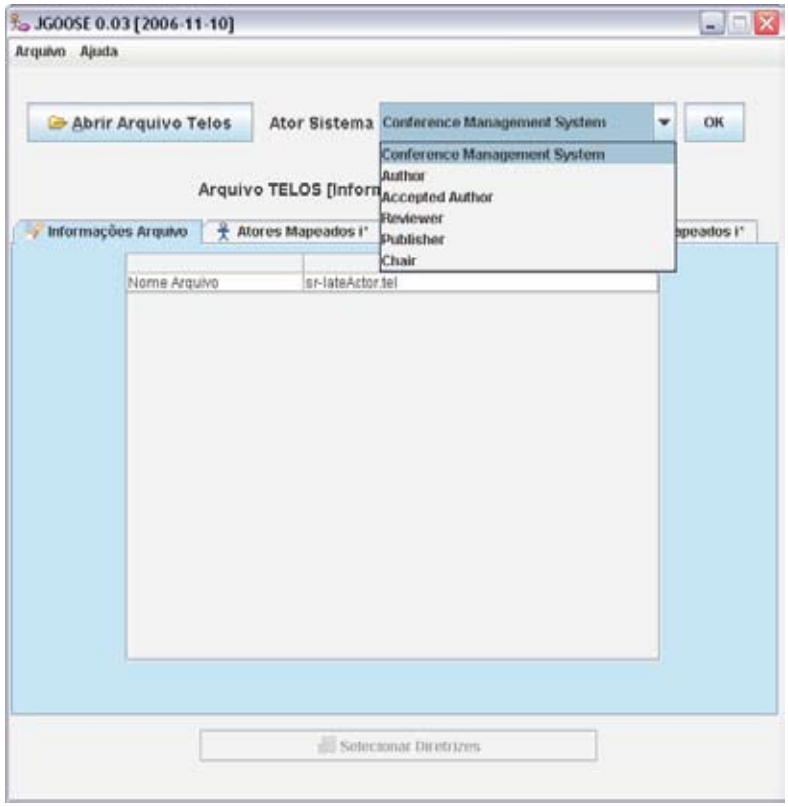

Figure 4. Step 1- JGOOSE tool Main Window.

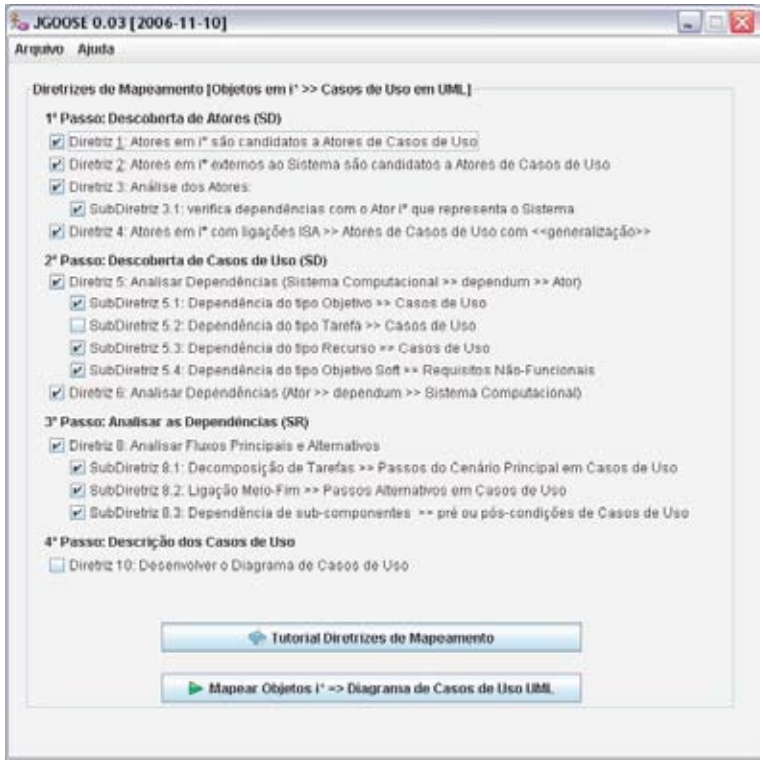

Figure 5. Step 2 - Guidelines mapping selection.

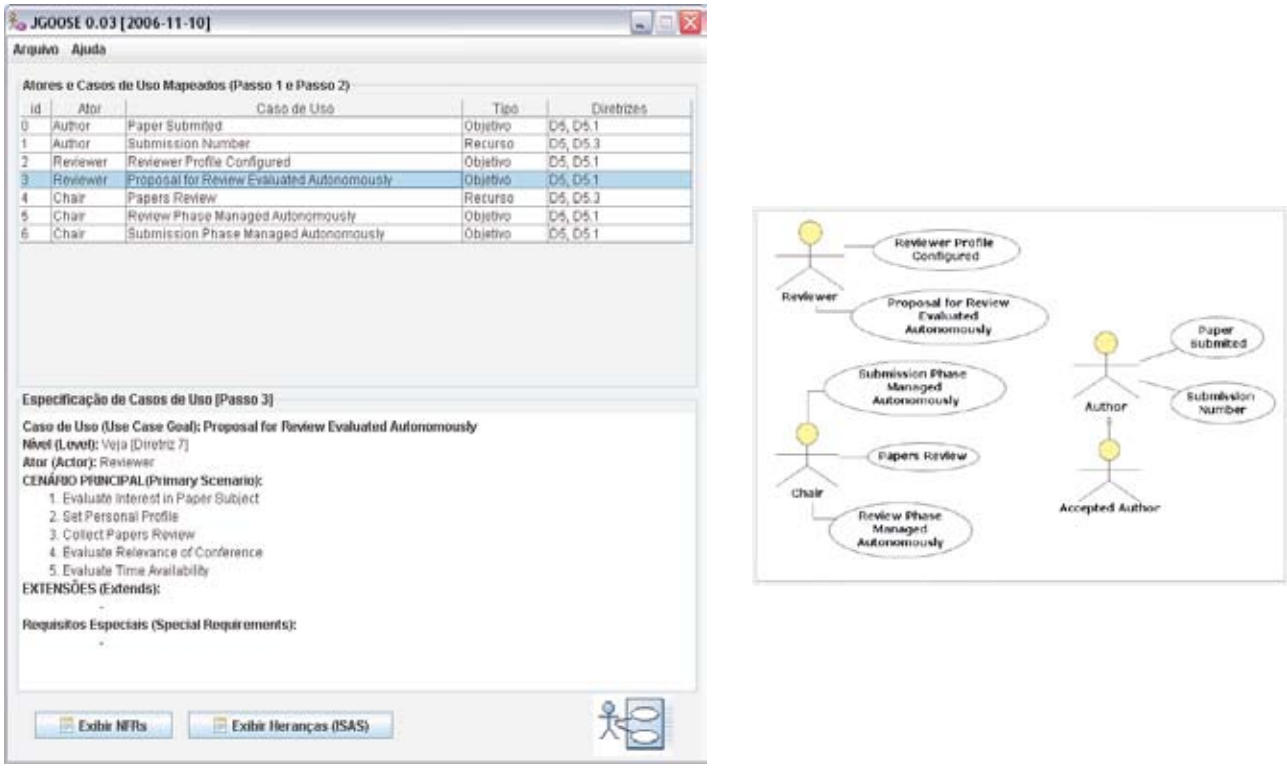

Figure 6. Step 3 - Actors, Use Cases and mapped Scenarios for the Conference Management System.

CASE STUDY

The mapping process of SD and SR models described in figures 1 and 2 to UML Use Cases occurs semi-automatically using the JGOOSE tool and this process derives Actors, Use Cases and your descriptions. This mapping process is described below:

1st Step – Mapping Actors

Figure 1 represents the Strategic Dependency model (SD) of the *Conference Management System*. It displays numbers corresponding to some of the elements in order to facilitate the reference in the text.

Initially, the tool uses the Step 1 guidelines, observing each of the actors **(1 - 6)** in the Strategic Dependency Model (figure 1) for a possible actor mapping in a Use Case Model. Thus, the actors *Conference Management System* **(1)**, *Reviewer* **(2)**, *Chair* **(3)**, *Publisher* **(4)**, *Author* **(5)**, *Accepted Author* **(6)** are assessed. The actor *Conference Management System* **(1)** should not be mapped because it represents the computer system to be developed. The actor that represents the computer system is selected by the user in the 1st step of the tool. The actor *Publisher* **(4)** is not mapped either, because he doesn't have any relationship with the Actor that represents the Computer System. The other actors are mapped and *Accepted Author* **(6)** is mapped with a <<generalization>> relationship, because of their IS-A relationship with actor *Author* **(5)** (see guideline 4).

Finally the table 2 shows mapped and not mapped Actors and their respective guidelines.

Table 2. Mapped Actors – Conference Management System.

Actor	Guideline	Mapping
Chair	G.1, G.2 and G.3	Mapped
Reviewer	G.1, G.2 and G.3	Mapped
Author	G.1, G.2 and G.3	Mapped
Accepted Author	G.1, G.2 and G.4	Mapped
Publisher	G.1, G.2	Not Mapped
Conference Management System	G.1	Not Mapped

2nd Step – Use Cases Mapping

Next, in Step 2, the JGOOSE tool looks for Use Cases examining the relationship between actor's dependencies

and the computer system (Conference Management System) in the SD model (figure 1). You can observe relationships mapped between *"Mapped Actors » Dependency » System"* and *"System » Dependency » Actor mapped"*:

- **Reviewer:** *Reviewer Profile Configured* **(12)** and *Paper Reviewed* **(10)** goals link the actor *Reviewer* **(2)** already mapped for actor's use case to the computer system represented by *Conference Management System* **(1)** and therefore are mapped to Use Cases using sub-guidelines of guideline 5 ("System » dependum » Reviewer"). The resource *Paper* **(15)** and the goal *Proposal Review Evaluated Autonomously* **(11)** has connection with the system like "Reviewer » dependum » System" and should be mapped following guideline 6.
- **Chair:** the *Submission Phase Managed Autonomously* **(8)** and *Review Phase Managed Autonomously* **(9)** goals links the actor *Chair* **(3)** already mapped for Use Case's actor and consequently are mapped to Use Cases using guideline 6. The resource *Papers Reviewed* **(14)** also has link with the computer system and consequently it is mapped using the guideline 6.
- **Author:** The *Paper Submitted* goal **(7)** and the *Submission Number* resource *(13)* are connected with the system through the already mapped actor *Author* **(5)**. Hence, they are mapped to Use Cases using the guideline 6.
- **Accepted Author:** The actor has no element connected to the system. However it inherits all the actor's use cases already mapped by Author **(5).**

Table 3 summarizes the step 2, showing the dependencies mapped to use cases and their respective actors.

3rd Step - Building Use Cases Description (Scenarios)

Finally, based on the guideline 8.1 of 3rd step of guidelines, the JGOOSE tool generates the steps of use cases: *Submission Phase Managed Autonomously* **(7)**, *Review Phase Managed Autonomously* **(8)** *Proposal for Review Evaluated Autonomously* **(9)**, observing the decompositions in the SR model (figure 2).

A first draft of the main scenario, according to the template proposed by [9] is generated for each use case. The description for the Use Case *Review Phase Managed Autonomously* can be seen in figure 7. Although, the main scenario generated should be properly examined and rewritten and rearranged by the requirements engineer. Thus, after following mapping guidelines, the JGOOSE tool generates the Actor and use cases relationship (showed in figure 6) mapped to the Conference Management System and the main scenario steps for the *Submission Phase Managed Autonomously*, *Proposal for Review Evaluated Autonomously* and *Review Phase Managed Autonomously* use cases.

Figure 8 illustrates the UML Use Case diagram for the *Conference Management System*, which you can view the actors *Author*, *Accepted Author* linked to actor *Author* by a generalization link, *Reviewer, Chair* and their Use Cases.

Table 3. Use Cases Mapped – *Conference Management System.*

Dependency Actor	Dependency	Dependency Type	Guideline
Reviewer	*Reviewer Profile Configured*	Goal	G.5 G.5.1
Reviewer	*Paper Reviewed*	Goal	G.5 G.5.1
Reviewer	*Paper*	Resource	G.6
Reviewer	*Proposal for Review Evaluated Autonomously*	Goal	G.6
Chair	*Submission Phase Managed Autonomously*	Goal	G.6
Chair	*Review Phase Managed Autonomously*	Goal	G.6
Chair	*Papers Reviewed*	Resource	G.6
Author	*Paper Submitted*	Goal	G.6
Author	*Submission Number*	Resource	G.6

Especificação de Casos de Uso [Passo 3]

Caso de Uso (Use Case Goal): Review Phase Managed Autonomously
Nível (Level): Veja [Diretriz 7]
Ator (Actor): Chair
CENÁRIO PRINCIPAL(Primary Scenario):
1. Select 'n' Reviewers of Paper Research Area
2. Collect Papers Review
3. Assign Paper Reviewer
4. Propose Paper Review

EXTENSÕES (Extends):
-
Requisitos Especiais (Special Requirements):
-

Figure 7. Use Case Specification – *Review Phase Managed Autonomously.*

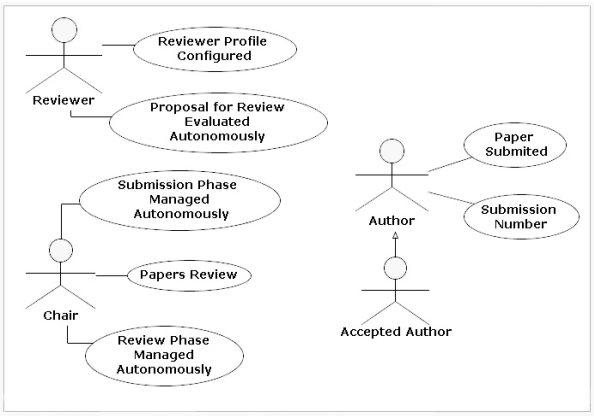

Figure 8. Use Case Diagram – *Conference Management System.*

CONCLUSION AND FUTURE WORKS

In this paper, it was argued that use cases development can be improved with the use of i* technique. In [6] the authors describe some of the benefits of use cases development from the i* technique. The integrated use of organizational modeling i* and UML Use Cases modeling provide a consistent link between computer systems allowing better assessment of the dynamics in business processes and organizational environments.

Initially, this paper presented guidelines that allow the use of i* organizational models in the scenarios development in Use Cases form. Afterwards, the JGOOSE tool is presented with its operation, which provides support in the semi-automatic use of the guidelines. The guidelines and the tool were applied to a Conference Management System case study, getting the system UML Use Cases. From the case study was possible to see that information in both, existing strategic dependency and strategic rationale model, serve as a basis for the development a Use Case diagram that meets the customers / users goals. In this context, we should emphasize the support offered by the JGOOSE semi-automated tool which minimizes the requirements engineers' effort in Use Cases mapping when applying the proposed guidelines. Additionally, to help this mapping process, the JGOOSE tool supports some usability, flexibility and portability requirements which are fundamental aspects in a computational tool design.

Some related work are cited below: Pedroza's [17] and Alencar's [18] proposal which aims to derive classes diagram from i* models; Rosa`s and Santander`s [29, 31], which works with the requirement elicitations for legacy software proposing mapping the DFDs (Data Flow Diagrams) to i* models; and Estrada's work [11] that suggests organizational and functional goals derived from business models. Moreover, Silva's [22] work related to Tropos Project [24] proposes a mapping from i* technique to a architecture level of Multi-Agent Systems.

In the automation mapping process aspect, we have some tools available, such as GOOD (Goals into Object Oriented Development) [8] and its improved version XGOOD (eXtended Goal Into Object Oriented Development) [18], that map i* objects to UML Classes Diagram, and the GOOSE tool (Goal Into Object Oriented Standard Extension) [3], the JGOOSE previous version tool, which maps i* objects to UML Use Cases diagram.

Nowadays we are investigating the possibility of the integrate the present proposal to the OOMethod [32]. OOMethod is a methodological approach with tools to support automatic code generation from OO conceptual models. We believe that our proposal can help to generate OO conceptual models from i* models in a better systematic way.

In future endeavor we will address the following aspects:

- Mapping process Improvement, as well as JGOOSE tool interface;
- use the tool in other case studies, mainly studying the tool behaviour in more complex diagrams SD/SR;
- Improving the tool, enabling the UML Use Cases mapping saved in XMI standard [4] which are compatible with a larger number of UML CASE tools;
- Adequately treat non-functional requirements using NFR Framework proposed in [28].

ACKNOWLEDGEMENTS

This work was supported by the following research grants: CAPES-DGU Proc. BEX1959/08-5, PIBIC/UNIOESTE/PRPPG (019/2005-PRPPG) and CNPq (130012/2008-4).

REFERENCES

[1] F. Alencar, J. Castro, G. Cysneiros and J. Mylopoulos. "From Early Requirements Modeled by i* Technique to Later Requirements Modeled in Precise UML". III Workshop de Engenharia de Requisitos. Rio de Janeiro, Brasil. 2000.

[2] F.M.R. Alencar, F.P. Pedroza, J. Castro, C.T.L. L. Silva and R.A. Ramos. "XGOOD: A Tool to Automatize the Mapping Rules between I* Framework and UML". IDEAS´06 - IX Workshop Iberoamericano de Ingeniería de Requisitos y Ambientes de Software. La Plata, Argentina. 2006.

[3] M. Brischke, V. Santander and J. Castro. "GOOSE: A Tool for integrating Organizational and Functional Modeling". Jornadas Chilenas de Computación - V Workshop Chileno de Ingeniería de Software. Valdivia, Chile. 2005.

[4] S. Brodsky, G. Doney and T. Grosse. "Mastering XMI Java Programming with XMI, XML, and UML". John Wiley & Sons, New York, USA. 2002.

[5] G. Booch, J. Rumbaugh and I. Jacobson. "UML: Guia do Usuário". Elsevier. 2ª Edição. Rio de Janeiro, Brasil. 2005.

[6] J. Castro, F.M.R. Alencar, V.F.A. Santander and C.T.L.L. Silva. "Integration of i* and Object Oriented Models" (to appear). In: Eric Yu, John Mylopoulos, Neil Maiden, Paolo Giorgini. (Org.). Social Modelling for Systems. MIT Press. Vol. 1, p. 19. 2006.

[7] J.F.B. Castro, M. Kolp and J. Mylopulos. "Towards Requirements-Driven Information Systems Engineering: The Tropos Project". Information Systems Journal, Elsevier, Amsterdam, The Netherlands. 2002.

[8] G.A. Cysneiros. "Ferramenta Para o Suporte do Mapeamento da Modelagem Organizacional em i*". Dissertação de Mestrado. Universidade Federal de Pernambuco. Recife-PE, Brasil. 2001.

[9] A. Cockburn. "Writing Effective Use Cases, Humans and Technology". Addison-Wesley. 2000.

[10] E. Project. "Eclipse Project". 2006. Fecha de consulta: Novembro de 2006. URLs: http://www.eclipse.org

[11] H. Estrada, A. Martínez, O. Pastor y J. Sanches. "Generación de Especificaciones de Requisitos de Software a partir de Modelos de Negocios: un Enfoque basado en Metas". V Workshop de Engenharia de Requisitos WER'02. Valencia, España. 2002.

[12] Sun Developer Network (SDN). "Java 2 Platform Standard Edition (J2SE) 1.5". 2006. Fecha de consulta: Noviembre de 2006. URLs: http://java.sun.com/j2se/1.5.0/

[13] J.C.S.P. Leite. "Requisitos: a ponte entre a organização e o software". Palestra da Rio Info - Seminário Internacional de Engenharia de Software. 2006.

[14] E. Yu. "Organization Modelling Environment". 2002. Fecha de consulta: Dezembro de 2006. URLs: http://www.cs.toronto.edu/km/ome

[15] OMG. "Object Management Group (OMG)". 2006. Fecha de consulta: Abril de 2006. URLs: http://www.omg.org

[16] E. Yu and Y. Yu. "OpenOME Organization Modelling Environment". 2006. Fecha de consulta: Dezembro de 2006. URLs: http://www.cs.toronto.edu/km/openome

[17] F. Pedroza, F.M.R. Alencar, J.F.B. Castro, F.R. C. Silva e V.F.A. Santander. "Ferramentas para Suporte do Mapeamento da Modelagem i* para a UML: eXtended GOOD – XGOOD e GOOSE". WER'04 - Workshop em Engenharia de Requisitos. Tandil, Argentina. 2004.

[18] F.P. Pedroza. "Automatizando as Regras de Mapeamento entre a Modelagem i* e a Modelagem UML usando XMI para Implementação de um Simulador de Rede Ópticas". Dissertação de Mestrado. Universidade Federal de Pernambuco. Recife-PE, Brasil. 2005.

[19] V.F. Santander and J.F Castro. "Deriving use cases from organizational modeling". IEEE Joint International Requirements Engineering Conference - RE 02. Essen, Germany. 2002.

[20] V.F.A. Santander. "Integração de Modelagem Organizacional com Modelagem Funcional na Engenharia de Requisitos". Tese de Doutorado. Universidade Federal de Pernambuco. Recife-PE, Brasil. 2002.

[21] I.G.L. Silva. "Projeto e Implementação de Sistemas Multi-Agentes: O Caso Tropos". Dissertação de Mestrado. Universidade Federal de Pernambuco. Recife-PE, Brasil. 2005.

[22] C.T.L.L. Silva, J. Castro, P.A.J. Tedesco, A.M. D. Moreira and J. Mylopoulos. "Improving the

architectural design of multi-agent systems: the tropos case". International Workshop on Software Engineering for Large-Scale multi-agent systems - SELMAS06. International Conference on Software Engineering ICSE. Shanghai, China. 2006.

[23] I. Sommerville and G. Kotonya. "Requirements Engineering: Processes and Techniques". John Wiley & Sons. New York, NY, USA. 1998.

[24] Tropos. "Tropos - Requirements-Driven Development for Agent Software". 2006. Fecha de consulta: Agosto de 2008. URLs: http://www.troposproject.org

[25] E. Yu. "Modelling Strategic Relationships for Process Reengineering". Phd Thesis. University of Toronto. 1995.

[26] XMI "Object Management Group OMG XML Metadata Interchange (XMI) Specification". 2005. Fecha de consulta: Março de 2006. URLs: http://www.omg.org/technology/documents/formal/xmi.htm

[27] F. Zambonelli, N.R. Jennings and M. Wooldridge. "Developing multiagent systems: The gaia methodology". ACM Trans. Softw. Eng. Methodology. Vol. 12 Nº 3, pp. 317-370. 2003.

[28] L. Chung, B.A. Nixon, E. Yu and J. Mylopoulos. "Non-Functional Requirements in Software Engineering". Kluwer Academic Publishers. 2000.

[29] M.C.S. Rosa. "Elicitação de Requisitos Funcionais e Não-Funcionais em Software Legado com Ênfase na Engenharia de Requisitos Orientada a Objetivos". Monografia de graduação. Universidade Estadual do Oeste do Paraná. Cascavel-PR, Brasil. 2005.

[30] A.A. Vicente "Ferramenta JGOOSE - Java Goal Into Object Oriented Standard Extension". 2006. Fecha de consulta: Maio de 2007. URLs: http://andvicente.googlepages.com/aav(undergraduate)

[31] V.F.A. Santander, A. Abe Vicente, F. Koerich e J. F.B. Castro. "Elicitação de Requisitos Organizacionais, Não-Funcionais e Funcionais em Software Legado com Ênfase na Engenharia de Requisitos Orientada a Objetivos". X Workshop Iberoamericano de Ingeniería de Requisitos y Ambientes de Software. Isla Margarita, Venezuela. 2007.

[32] O. Pastor and J.C. Molina. "Model-Driven Architecture in Practice: A Software Production Environment Based on Conceptual Modeling". Springer. 2007.