Deriving Use Cases from Organizational Modeling

Victor F.A. Santander [*] Jaelson F. B. Castro

Universidade Federal de Pernambuco – Centro de Informática
Cx. Postal 7851, CEP 50732-970, Recife-PE, BRAZIL
Phone: (+55 81) 3271-8430, Fax: (+55 81) 3271-8438
{vfas,jbc}@cin.ufpe.br

Abstract

Use Cases Diagrams in the Unified Language Modeling (UML) have been used for capturing system functional requirements. However, the system development occurs in a context where organizational processes are well established. Therefore, we need to capture organizational requirements to define how the system fulfils the organization goals, why it is necessary, what are the possible alternatives, etc. Unfortunately, UML is ill equipped for modeling organizational requirements. We need other techniques, such as i, to represent these aspects. Nevertheless, organizational requirements must be related to functional requirements represented as Use Cases. In this paper we present some guidelines to assist requirement engineers in the development of Use Cases from the i* organizational models.*

1. Introduction

System development occurs in a context where organizational processes are well established. However, as discovered in empirical studies, the primary reason for software system failure is the lack of proper understanding of the organization by the software developers. Unfortunately, the dominant object oriented modeling technique, UML, is ill equipped for organizational requirement modeling. We need others techniques, such as i* [15] to represent these aspects. We argue that i* framework, is well suited to represent organizational requirements that occur during the early-phase requirements capture, since it provides adequate representation of alternatives, and offers primitive modeling concepts such as softgoal and goal. These early activities would enable an understanding of how and why the requirements came about.

Nevertheless, organizational requirements must be related to functional requirements represented with techniques such as Use Cases. However, Use Case development demands great experience of the requirement engineers. The heuristics presented in the literature to develop Use Cases are not sufficient to allow a systematic development. Indeed, they do not consider relevant organizational aspects such as goals and softgoal.

In this work, we propose some guidelines to support the integration of i* and Use Case modeling. We describe some heuristics to assist requirement engineers to develop Use Cases based on the organizational i* models. This paper is organized as follows. Section 2 introduces the concepts used by i* framework to represent organizational requirements and early requirements. In Section 3, we review Use Case modeling. In Section 4, we present the benefits of our approach as well as describe the guidelines to integrate i* organizational models and Use Cases diagrams. In Section 5, we introduce a brief case study to show the viability of our proposal. Section 6 discusses related works and concludes the paper.

2. The i* Modeling Framework

When developing systems, we usually need to have a broad understanding of the organizational environment and goals. The i* framework [15] provides understanding of the reasons ("Why") that underlie system requirements. I* offers two models to represent organizational requirements: the Strategic Dependency (SD) Model and the Rationale Dependency (SR) Model.

2.1. The Strategic Dependency Model - SD

This model focuses on the intentional relationships among organizational actors. It consists of a set of nodes and links connecting them, where nodes represent actors and each link represents the dependency between actors. The depending actor is called Depender and the actor who is depended upon is called Dependee. The i* framework defines four types of dependencies among actors: goal,

[*] Partially Supported by CNPq Grant No. 147192/1999-4. On-leave from Universidade Estadual do Oeste do Paraná.

resource, task and softgoal. Figure 1 shows an Strategic Dependency (SD) Model of the meeting scheduling setting with a computer-based meeting scheduler [15].

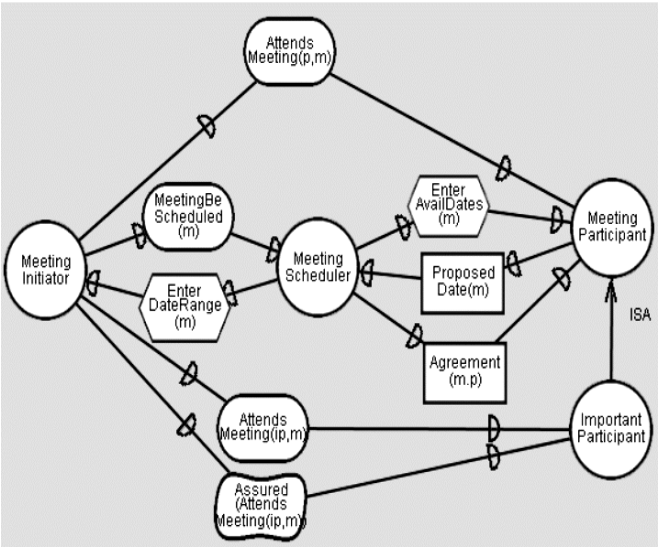

Figure 1. Strategic Dependency Model for the Meeting Scheduling Problem.

The meeting initiator depends on participant to attend the meeting. The meeting initiator delegates much of the work of meeting scheduling to the meeting scheduler. The meeting scheduler determines what are the acceptable dates, given the availability information (*task dependency EnterAvailDates(m*)). The meeting initiator does not care how the scheduler does this, as longer as the acceptable dates are found. This is reflected in the goal dependency *MeetingBeScheduled* from the initiator to the scheduler. On the other hand, to arrive at an agreeable date, participants depend on the meeting scheduler for date proposals (*resource dependency ProposedDate(m)*). Once proposed, the scheduler depends on participants to indicate whether they agree with the date (*resource dependency Agreement(m,p)*). For important participants, the meeting initiator depends critically on their attendance, and thus also on their assurance that they will attend (*softgoal dependency Assured(AttendsMeeting(ip.m))*). The meeting scheduler depends on the meeting initiator to provide a date range (*task dependency EnterDateRange(m)*) for the scheduling.

2.2. The Strategic Rationale Model - SR

The Strategic Rationale (SR) model allows modeling of the reasons associated with each actor and their dependencies. Two news links are added to previous notation:

- Means-ends: This link indicates a relationship between an end - which can be a goal to be achieved, a task to be accomplished, a resource to be produced, or a softgoal to be satisficed - and a means for attaining it.
- Task-decomposition: A task is modeled in terms of its decomposition into its sub-components. These components can be goals, tasks, resources, and/or sofgoals.

In Figure 2, we present an example of the Strategic Rationale (SR) model. We use the SR notation to detail the Meeting Scheduler actor. Due to space limitation, we do not detail the Meeting Initiator and Meeting Participant actors (see the complete model in [15]). The Meeting Scheduler actor represents a software system that partially performs the meeting scheduling, while the Meeting Initiator and Meeting Participant, are responsible for providing or receiving information to the system. The Meeting Scheduler actor possesses a Schedule Meeting task which is decomposed into three sub-components using the **task-decomposition relationship**: *FindAgreeableSlot, ObtainAgreemet and ObtainAvailDates*. These sub-components represent the work that will be accomplished by the meeting scheduler system.

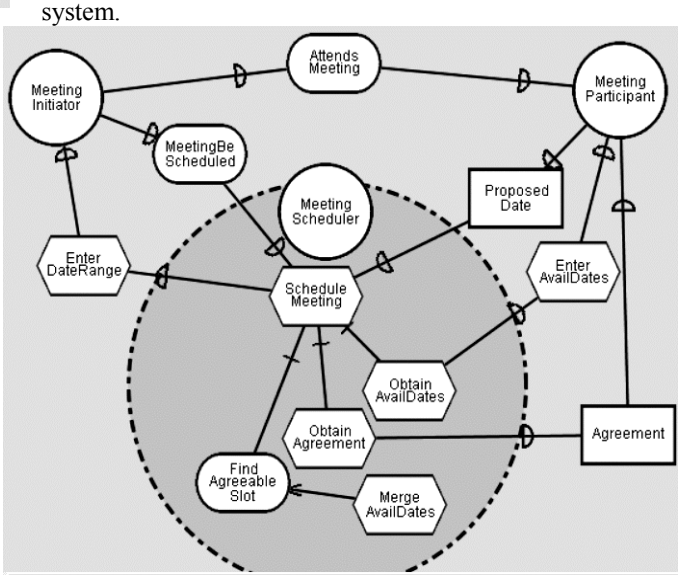

Figure 2. Strategic Rationale (SR) Model to the Meeting Scheduler System.

3. Use Cases in UML

Scenario-based techniques have been used by the software engineering community to understand, model and validate users requirements [9] [10] [13] [14]. Among these techniques, Use Cases have received a special attention in the object oriented development community. Use Cases in UML [3] are used to describe the use of a system by actors. An actor is any external element that interacts with the system. A Use Case is a description of a set of sequences of actions, including variants, that a

system performs that yields an observable result value to an actor. It is desirable to separate main (primary scenario) versus alternative (secondary scenario) flows because a Use Case describes a set of sequences, not just a single sequence, and it would be impossible to express all the details of an interesting Use Case in just one sequence.

In order to cope with increasing complexity of Use Cases description, UML caters for three structuring mechanism: inclusion, extension and generalization. For further information see [3].

4. Deriving Use Cases from Organizational Modeling.

In this section we argue how our approach can improve the Use Case development. In section 4.1 we outline the main benefits accomplished by approach and in section 4.2 we describe it in detail.

4.1. Benefits of i* and Use Case Integration

i* provides an early understanding of the organizational relationships in a business domain. As we continue the development process, we need to focus on the functional and non-functional requirements of the system-to-be. As a first step in the late requirements phase we can adopt Use Cases to describe functional requirements of the system. We argue that the Use Case development from organizational modeling using i* allows requirement engineers to establish a relationship between the functional requirements of the intended system and the organizational goals previously defined in the organization modeling. Besides, through a goal-oriented analysis of the organizational models, we can derive and map goals, intentions and motivations of organizational actors to main goals of Use Cases. We assume, that for each Use Case we have associated a main goal, which represents what the user aims to reach as a result of the execution of the Use Case. In our proposal, the Use Case scenario description is based on organizational models, which are well known and understood by all stakeholders. Note that our approach can be used for any type of system.

We can mention other important benefits obtained using our approach, such as:

- Many researchers [1] [6] [8] [14] [16] have considered goals in a number of different areas of Requirements Engineering. Goal-oriented approaches to requirements acquisition may be contrasted with techniques that treat requirements as consisting only of processes and data, such as traditional systems analysis or "objects", such as the object-oriented methods, but which do not explicitly capture why and how relationships in terms of goals.
- The relationships between systems and their environments can also be expressed in terms of goal-based relationships. This is partly motivated by today's more dynamic business and organizational environments, where systems are increasingly used to fundamentally change businesses process [16]. Deriving Use Cases from i* relationships allows traceability and evaluation of the impact of these changes into the functional requirements of the intended system;
- Some of the Use Case pitfalls and drawbacks described in [11], can be partially solved using our approach. For instance, Use Cases are written from the actor's (not the system's) point of view. We derive Use Cases from actors dependencies defined explicitly in i*. Another positive aspect is the ability to define the essential Use Cases for the intended system. This avoids defining too many Use Cases and allows managing the appropriate granularity of Use Cases. Finally, the integration between requirements engineers and customers during the organizational model development also allows customers (actors) to better understand the Use Cases originated from these models;
- To elicit and specify system requirements observing the actor's goal in relation to the system-to-be, is a way of clarifying requirements [16]. From i* we can derive these goals, associate them with system actors and then refine and clarify the requirements into Use Cases.

4.2. Proposed Approach

To guide the mapping and integration process of i* organizational models and Use Cases, we have defined some guidelines which must be applied according to the steps represented in Figure 3. In this figure, steps 1, 2 and 3 represent the discovery of system actors and its associated Use Cases diagrams and descriptions. The input for the integration process are the Strategic Dependency (SD) and Strategic Rationale (SR) models developed through i* framework. In steps 1 and 2, the input is the Strategic Dependence (SD) Model. The description of scenarios for Use Cases (step 3) is derived from elements represented in the Strategic Rationale (SR) Model. The results of the integration processes are Use Case diagrams for the intended system and scenario textual descriptions for each Use Case.

In the sequel we suggest heuristics for the Use Cases development from organizational modeling with i*.

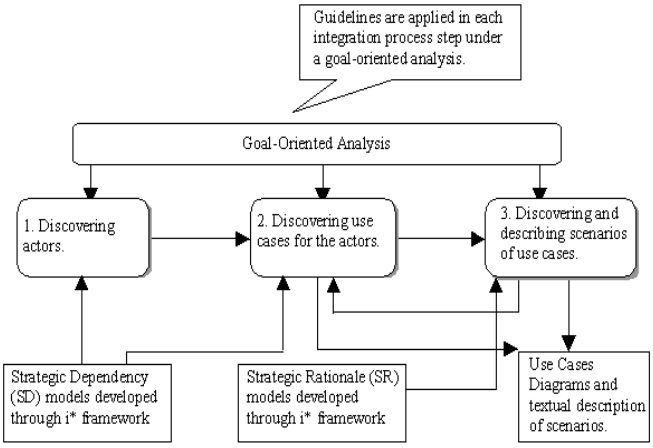

Figure 3. Steps of the integration process between i* and Use Cases in UML

1º Step: Discovering System Actors.

Guideline 1*: *every actor in i* should be considered as a possible ***Use Case actor***; *For example, the Meeting Participant i* actor in Figure 1* is a possible UML actor.

***Guideline 2*:** the actor considered in i* should be external to the intended software system. For example, the Meeting Participant actor is external to the system because it will interact with the intended meeting scheduler system.

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

> ***Guideline 3.1:*** the actor dependencies in i* must be relevant from the point of view of the intended system; For instance, the Meeting Participant actor in i* can be mapped to Use Case actor, considering that dependencies associated with it, characterizes it as important in an interaction context with the meeting scheduler system.

Guideline 4: actors in i*, related through the IS-A mechanism in the organizational models and mapped individually for actors in Use Cases (applying guidelines 1, 2 and 3), will be related in the Use Case diagrams through the <<generalization>> relationship. For instance, the IS-A relationship between Meeting Participant and Important Participant in Figure 1, can be mapped to generalization relationship between these actors in the Use Case diagram.

2º Step: Discovering Use Cases for the Actors.

Guideline 5*:** for each discovered ***actor of the system (step 1), we should observe all its dependencies (dependum) in which the actor is a ***dependee***, looking for Use Cases for the actor; Initially, we recommend to create a table containing the discovered actors and the information about the dependencies for the actor from the point of view of a dependee. Moreover, you can include which guideline(s) to be used to analyze each dependency (dependum) (see table 1). *For instance, some Use Cases can be associated with the **Meeting Participant** actor observing their dependencies presented in i*:*

> ***Guideline 5.1*:** goal dependencies - goals in i* can be mapped to Use Case goals; *For instance, in Figure 1, the goal dependency AttendsMeeting(p,m) between Meeting Initiator (Depender) and Meeting Participant (Dependee) can be mapped to the **AttendsMeeting** Use Case, which will contain the several steps accomplished by Meeting Participant to attends to the meeting.*

Actor	Dependency	Type of Dependency	Guideline to be used
Meeting Participant	AttendsMeeting(p,m)	Goal	(G5.1)

Table 1. Gathered information from SD Models to aid requirement engineers to derive Use Cases.

> ***Guideline 5.2:*** task dependencies - if an actor depends on another actor for the accomplishment of a task, it should be investigated if this task needs to be decomposed into other sub-tasks. *For example, for the task dependency EnterDateRange(m) associated with the Meeting Initiator actor (see Figure 1), we can consider that the task of supplying a date range for the meeting scheduling can include several aspects (later mapped to Use Case steps) such as to associate range dates with specific meetings, to establish priorities for specific meetings, etc. Thus, from the task EnterDateRange(m) we can generate the Use Case called **EnterDateRange** for the Meeting Initiator actor.*
>
> ***Guideline 5.3*:** resources dependencies - if an actor depends on another actor for obtaining a resource(s), why is it required? If there is a more abstract goal, it will be the candidate goal of the Use Case for the actor. *For instance, for the resource dependency Agreement(m,p) associated with the Meeting Participant actor (see Figure 1), we conclude that the main goal of obtaining of Agreement(m,p) resource is a scheduled date agreement from each participant. We could consider that in this agreement process, each participant could agree with the proposed meeting date with certain schedule restrictions or duration time. Still, the agreement could involve an analysis of other possible dates. In other words, to obtain the scheduled date agreement, several interaction steps between meeting scheduler and meeting participant could be defined in one Use Case called **Agreement** for the Meeting Participant actor.*
>
> ***Guideline 5.4:*** sofgoal dependencies - typically, the sofgoal dependency in i* is a non-functional requirement for the intended system. Hence, a softgoal does not represent a Use Case of the system but a non-functional requirement associated with a Use Case of

the system. *For instance, the softgoal Assured(AttendsMeeting(ip,m)) between Meeting Initiator and Important Participant actors can be mapped into a non functional requirement associated with the Use Case AttendsMeeting. This non-functional requirement indicates that it is necessary to assure that the Important Participant attends to the meeting*.

Guideline 6: analyze special situations, where an actor discovered (following the step 1), possess dependencies in relation to an actor in i* that represents an intended software system or part of it. These dependencies usually generate Use Cases. It is important to notice that in this situation the derived Use Case is associated with the **depender** actor in the relationship. This occurs 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 instance, the goal dependency MeetingBeScheduled between Meeting Initiator and Meeting Scheduler system in the Figure 1, points out for the definition of the Use Case **MeetingBeScheduled** for the Meeting Initiator actor, which represents the use of the system by the actor, describing the details of the meeting scheduling process.

Guideline 7: classify each Use Case according to the type associated to its goal (business, summary, user goal or subfunction). This is based on a classification scheme proposed by Cockburn [7]. A *business* goal represents a high level intention, related to business processes, that the organization or user possesses in the context of the organizational environment. An example could be the goal "organizing a meeting in the possible shortest time". A *summary* goal represents an alternative for the satisfaction of a business goal, as in the case of the goal, "meeting scheduling by software system". An *user* goal results in the direct discovery of a relevant functionality and value for the organization actor using a software system. An example could be the goal, "the meeting participant wishes to attend the meeting". Finally, *subfunction-level* goals are those required to carry out user goals. An example could be the goal, "enter date range for meeting scheduling" by the Meeting Initiator. To aid requirement engineers to identify new Use Case and better understand the discovered Use Cases, we recommended to generate a table containing the actor name, the Use Case goal and the goal classification (see table 2).

Actor	Use Case Goal	Goal Classification
Meeting Participant	AttendsMeeting	User Goal

Table 2. Use Case goal classification.

3º Step: Discovering and Describing Use Case Scenario.

Guideline 8: analyze each actor and its relationships in the Strategic Rationale (SR) model, to extract information that can lead to the description of the Use Cases scenario for the actor. It is important to remember that SR models represent the internal reasons associated with the actor goals. Therefore, we must consider internal elements which are used by the actor to achieve goals and sofgoals, to perform tasks or obtain resources. The actor has the responsibility to satisfy these elements and the decomposition in SR shows how the actor will be performing this. Typically, the dependencies associated with the actor are satisfied internally through two types of relationships used in SR: **means-ends and task-decomposition.** These relationships must be observed to derive scenario steps for the Use Cases. For instance, consider the Strategic Rationale (SR) Model in Figure 2. From the Meeting Scheduler actor point of view, we know that the Schedule Meeting task is decomposed into ObtainAvailDates, FindAgreeableSlot and ObtainAgreement. Since the software system objective is to accomplish meeting scheduling, we could consider that these tasks are the necessary high-level steps to accomplish a meeting schedule (Use Case **MeetingBeScheduled** defined for the Meeting Initiator actor). Thus, this Use Case could contain the steps (the primary scenario description) regarding the need to obtain from each Meeting Participant, the available dates for a meeting (ObtainAvailDates); the need to define the best meeting dates that could be scheduled (FindAgreeableSlot); and to obtain the participants agreement for a proposed meeting date (ObtainAgreement).

5. Case Study

In this section, we follow the steps proposed in Figure 3 and apply the appropriate guidelines to the example described in the previous section (Figure 1 and 2). Recall that Figure 1 shows a Strategic Dependency (SD) model for meeting scheduling while Figure 2 represents the Strategic Rationale (SR) model. Hence, these organizational models are used to discover and describe Use Cases in UML for the Meeting Scheduler system. We begin deriving the Use Case actors from the SD model. We then find the Use Cases for the actors observing the actors dependencies in SD model. Next, the primary scenario for one derived Use Case is described from the SR model. Last but not least, a version of the Use Case diagram in UML for the Meeting Scheduler system is generated.

- From Figure 1, we can find candidates actors for the Use Case development. According to the guidelines in the ***1st step*** of the proposal, we conclude that one of the analyzed actors does not follow guideline 2. The Meeting Scheduler actor is a system, i.e. the software to be developed. Therefore, this i* actor cannot be

considered as a Use Case actor. The other i* actors are considered appropriate because their strategic dependencies refer to relevant aspects for the meeting scheduler system (guideline 3) development. So, the list of candidates Use Cases actors includes: ***Meeting Initiator, Meeting Participant*** and ***Important Participant***. We also note that Important Participant is a type (relationship IS-A) of participant. According to guideline 4 (1st step), we consider this actor a specialization of Meeting Participant actor.

The next step is to discover and relate Use Cases for each actor according to the guidelines presented in the ***2º Step*** *(Discovering Use Cases for the Actors*).

- Initially, following the guideline 5 and observing the SD model presented in Figure 1 we can generate the table 3.

Actor	Dependency	Type of Dependency	Guideline to be used
Meeting Participant	AttendsMeeting(p,m)	Goal	(G5.1)
Meeting Participant	EnterAvailDates(m)	Task	(G5.2)
Meeting Participant	Agreement(m,p)	Resource	(G5.3)
Meeting Initiator	EnterDateRange(m)	Task	(G5.2)

Table 3. Gathered information from SD Models to derive Use Cases for the Meeting Scheduler System.

- Thus, for the Meeting Participant actor, observing this actor as Dependee, we can indicate some Use Cases originated from the actor dependency relationships (guideline 5). Initially, we should consider the goal dependency (guideline 5.1) of the actor as Dependee. In table 3, we verify the goal AttendsMeeting(p,m), which represents the need of the meeting participant actor to attend the meeting. This goal originates the Use Case ***AttendsMeeting***. Several steps are necessary to achieve this goal. Typically, this is a *user* goal (guideline 7). The fulfillment of the Use Case goal brings a relevant result for Meeting Participant actor, allowing it to attend to the meeting. Usually, the description of the primary scenario (to be accomplished later) for this Use Case, will present other user goals that can originate new Use Cases for the system.

  The next dependency associated with the Meeting Participant actor is the task dependency EnterAvailDates(m). According to guideline 5.2, we can consider the need of several interaction steps among the participants (Meeting Participant actor) and the meeting scheduler system to enter available dates. Some steps could include participants to supply a list of exclusion dates and preferred dates in a particular format, to validate these dates by the system, etc. Thus, the task EnterAvailDates(m) generate the Use Case **EnterAvailDates** for the Meeting Participant Actor.

  Continuing our analysis, we can observe associated with the Meeting Participant (Dependee) actor the resource dependency Agreement(m,p). Following guideline 5.3, we conclude that the main goal of obtaining of Agreement(m,p) resource is an scheduled date agreement from each participant. We could consider that in this agreement process, each participant could agree with the proposed meeting date with certain schedule restrictions or duration time. Still, the agreement could involve an analysis of other possible dates. In other words, the schedule of dates requires several interaction steps between the system and the Meeting Participant actor, which defines the **Agreement** Use Case of the Meeting Participant actor.
- To discovery of Use Cases candidates for the Meeting Initiator actor follows the same guidelines (*2º Step)*. We have one dependency associated with the Meeting Initiator actor (see table 3): the task EnterDateRange(m). Using guideline 5.2 for this task dependency we observe that to supply a date range for the meeting scheduling can include several aspects (sub-tasks) such as to associate range dates with specific meetings, to establish priorities for specific meetings, etc. Thus, from the EnterDateRange(m) task we generate the **EnterDateRange** Use Case for the Meeting Initiator actor.

  Having considered all dependencies for the Meeting Initiator as Dependee, we should now consider special situations (guideline 6). Observing Figure 1, we visualize the goal dependency MeetingBeScheduled between Meeting Initiator and Meeting Scheduler (software to be developed), which requires some sort of interaction. Therefore, we can define the ***MeetingBeScheduled*** Use Case that represents the use of the system by the Meeting Initiator actor. In this Use Case, we describe the details of the meeting schedule process. Note that in this special situation the depender (meeting initiator) is the Use Case actor.
- Finally, following the guideline 7 we can to classify each discovered Use Case goal, as showed in the table 4.

Thereby, after we have used the proposed guidelines (*2º Step*), we have discovered **EnterDateRange** and **MeetingBeScheduled** Use Cases for the Meeting Initiator actor as well as **AttendsMeeting, EnterAvailDates** and **Agreement** Use Cases for the Meeting Participant actor**.** Therefore**,** we can begin the description of the primary and secondary scenarios and the Use Cases relationships (*3º Step*). At this point, the Strategic Rationale (SR) model is used as source of information for the scenario description and the Use Cases relationships.

Actor	Use Case Goal	Goal Classification
Meeting Initiator	EnterDateRange	Subfunction
Meeting Initiator	MeetingBeScheduled	Summary
Meeting Participant	AttendsMeeting	User Goal
Meeting Participant	EnterAvailDates	Subfunction
Meeting Participant	Agreement	Subfunction

Table 4. Goal Classification for the Meeting Scheduler System.

For example, the ***MeetingBeScheduled*** Use Case discovered for the Meeting Initiator actor represents the use of the system by Meeting Initiator to accomplish the meeting scheduling. This Use Case should contain all the necessary steps to schedule a meeting that begins when the Meeting Initiator supplies information to the system such as the date range to schedule a meeting. Based on the supplied dates range by Meeting Initiator, the system must find available dates for all the participants for the meeting as well as elaborate a consensus dates list within which a date will be chosen to be proposed and agreed. This process must result in a consensus-scheduled date for the meeting and later in the confirmation of this date for all the participants. Thus, for the Use Case MeetingBeScheduled, we could have the primary scenario with the following steps:

Use Case: **MeetingBeScheduled**
Actor: **Meeting Initiator**
Use Case Goal: **Schedule a Meeting** (summary goal, see guideline 7)
Primary Scenario:

1. The Use Case begins with the Meeting Initiator actor supplying the system with a date range for the meeting; (the *EnterDateRange* Use Case is included <<include>> in this step).
2. The system should request from participants (Meeting Participant) an available date list for the meeting based on the proposed date range by the Meeting Initiator; (the *EnterAvailDates* Use Case is included <<include>> in this step).
3. The system should find a consensus date list, filtering information observing the available dates sent by the participants and the proposed date range sent by Meeting Initiator;
4. Based on the consensus list, the system proposes a date for the meeting to be scheduled;
5. The Meeting Initiator expects that the system requests the agreement for a scheduled meeting date. (The *Agreement* Use Case is included << include >> in this step).

The information for the description of this Use Case has as main source the Strategic Rationale (SR) Model presented in the Figure 2. Following the guideline 8, we must observe which elements are involved in the SR model to achieve the MeetingBeScheduled goal by Meeting Scheduler actor. This actor has the responsibility to achieve MeetingBeScheduled which originated the **MeetingBeScheduled** Use Case (according to guideline 6). Thus, observing the internal strategic reasons associated with Meeting Scheduler we can conclude that the base information for the step 1 in this Use Case, is extracted from the EnterDateRange task dependency, establishing the need that Meeting Initiator supplies date range for the meeting to be scheduled. Previously, in the Use Case discovery for the system, we considered that the process of establishing a date range included several steps (sub-tasks) such as to associate range dates with specific meetings, to establish priorities for specific meetings, etc. These steps should be described in the **EnterDateRange** Use Case. For this reason, this Use Case is included <<include>> in step 1.

Steps 2 and 3 are extracted from the decompositions of the task Schedule Meeting (associated with Meeting Scheduler in the Figure 2). Step 2 derives from the observation of the ObtainAvailDates task and its associated EnterAvailDates task dependency. The **EnterAvailDates** Use Case is included <<include>> because it represents the necessary steps for the entry of the available dates list by participants. Step 3 originates from FindAgreeableSlot goal and the MergeAvailDates task. This step represents the internal actions of the system to define a list of the consensus dates for the meeting scheduling. Step 4, is extracted from observation of the ProposedDate resource dependency in connection with the task Schedule Meeting (Figure 2). It is assumed, given the defined information in the models of the Figure 1 and 2, that the proposed date should be defined by the system, using some previously established and defined criterion by the Meeting Initiator, taking as base for example, priorities of organization meetings.

Step 5, derives from the system need to obtain the agreement for the chosen date for the meeting scheduling. This information arises from the observation of the task ObtainAgreement and its associated resource dependency Agreement (Figure 2). Previously, in the Use Case discovery for the system, we assumed that in order for a participant to agree with the proposed date, it was necessary the accomplishment of some interaction steps between the participant and the Meeting Scheduler. These steps should be described in the Agreement Use Case. For this reason, this Use Case is included <include> in the step 5. We can describe the others Use Cases in a similar way.

After we have applied the proposed guidelines to this case study, we can define, as described in the Figure 4, a version of the Use Cases diagram in UML for the Meeting Scheduler system.

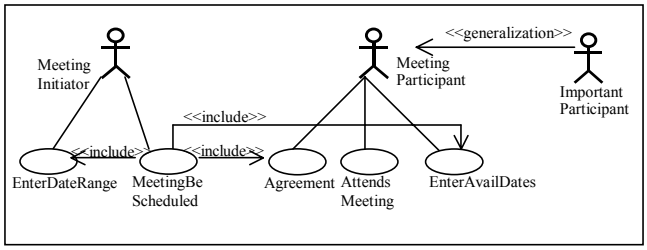

Figure 4. Use Case Diagram for the Meeting Scheduler system.

The descriptions of the discovered Use Cases could still be modified or complemented, as new relationships are elicited.

6. Conclusions and Related Works

In this paper we argued that the Use Cases development can be improved by using the i* organizational models. We presented some heuristics and a case study to show the viability and benefits of our approach.

Some related works include the requirements-driven development proposal presented in the ***Tropos*** framework [4] and the integration of i* and pUML diagrams [5]. These works argue that organizational models are fundamental for the development of quality software, which can satisfy the real needs of users and organizations. Several groups have also discussed the challenges and associated risks building quality system during goal and scenario analysis. For instance, the ScenIC method [12] uses goal refinement and scenario analysis as its primary methodological strategies. This method includes systematic strategies to identify actors, goals, tasks, and obstacles into evolving systems. In Anton et al. [2], the GBRAM method [1] is used to derive goals from a use-case based requirements specification. In the CREWS project [13] [14], the CREWS-L`Ecritoire approach [14] aims at discovering/eliciting requirements through a bi-directional coupling of goals and scenarios allowing movement from goals to scenarios and vice-versa. However, these approaches do not consider organizational models for deriving goals and scenarios for intended systems.

Further research is still required to describe more systematic guidelines, that can aid requirement engineers to relate non-functional requirements [6] (softgoals in i*) with functional requirements of the system, described through Use Cases in UML. Work is underway to incorporate goal-oriented modeling approaches [1] [8] [12] [14] into our proposal aiming at discovering other Use Cases from the exploration of already discovered goals. We also expect to develop more real case studies as well as to provide some tool support for the proposed mapping.

7. References

[1] A.I. Anton, *Goal identification and refinement in the specification of software-based information systems*. Phd Thesis, Georgia Institute of Technology, Atlanta, GA, June 1997.

[2] A.I. Anton, R.A. Carter, A. Dagnino, J.H. Dempster, and D.F. Siege, "Deriving Goals from a Use Case Based Requirements Specification", *Requirements Engineering Journal,* Springer-Verlag, Volume 6, pp. 63-73, May 2001.

[3] G. Booch, I. Jacobson, and J. Rumbaugh, *The Unified Modeling Language User Guide*, Addison-Wesley, 1999.

[4] J.F. Castro, M. Kolp, and J. Mylopoulos, "A Requirements-Driven Development Methodology", In: CAISE'01, Proceedings of the 13th Conference on Advanced Information Systems Engineering. Heildelberg, Germany: Springer Lecture Notes in Computer Science LNCS 2068, pp. 108-123, 2001.

[5] J.F. Castro, F. Alencar, G. Cysneiros, and J. Mylopoulos, "Integrating Organizational Requirements and Object Oriented Modeling", In Proceedings of the Fifth IEEE International Symposium on Requirements Engineering - RE'01, pp. 146-153, August 27-31, Toronto, 2001.

[6] L. Chung, B.A. Nixon, E. Yu, and J. Mylopoulos, *Non-Functional Requirements in Software Engineering (Monograph)*, Kluwer Academic Publishers, 472 pp, 2000.

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

[8] A. Dardene, V. Lamsweerde, and S. Fikas, "Goal-Directed Requirements Acquisition", Science of Computer Programming, 20, pp. 3-50, 1993.

[9] I. Jacobson, *Object Oriented Software Engineering: A Use Case Driven Approach,* Addison-Wesley, 1995.

[10] J.C.S.P. Leite, G. Rossi, F. Balaguer, and V. Maiorana, "Enhancing a requirements baseline with scenarios", In Proceedings of the Third IEEE International Symposium on Requirements Engineering – RE'97, pages 44-53. IEEE Computer Society Press, January 1997.

[11] S. Lilly, "Use Case Pitfalls: top 10 problems from Real projects using Use Cases", In: Proceedings, technology of object oriented languages and systems, pp 174-183, 1-5 August, 1999.

[12] C. Potts, "ScenIC: A Strategy for Inquiry-Driven Requirements Determination", In Proceedings of the Fourth IEEE International Symposium on Requirements Engineering – RE'99, Ireland, June 7-11, 1999.

[13] J. Ralyté, C. Rolland, and V. Plihon, "Method Enhancement With Scenario Based Techniques", In *Proceedings* of CAISE 99, 11th Conference on Advanced Information Systems Engineering Heidelberg, Germany, June 14-18, 1999.

[14] C. Rolland, C. Souveyet, and C.B. Achour, "Guiding Goal Modeling Using Scenarios", IEEE Transactions on Software Engineering, Vol 24, No 12, Special Issue on Scenario Management, December 1998.

[15] E. Yu, *Modelling Strategic Relationships for Process Reengineering,* Phd Thesis, University of Toronto, 1995.

[16] E. Yu and J. Mylopoulos, "Why Goal-Oriented Requirements Engineering", Proc. Fourth International Workshop Requirements Engineering: Foundations of Software Quality REFSQ'98, pp. 15-22, Pisa, June 1998.