Proceedings of the 11th International Conference on Requirements Engineering: Foundations for Software Quality (REFSQ'05).

A Goal-Based Round-Trip Method for System Development

Gemma Grau[1], Xavier Franch[1], Neil A.M. Maiden[2]

[1] Universitat Politècnica de Catalunya (UPC).
c/ Jordi Girona 1-3, Barcelona E-08034, Spain.
{ggrau, franch}@lsi.upc.edu
[2] Centre for HCI Design, City University.
Northampton Square, London EC1V OHB, UK.
n.a.m.maiden@city.ac.uk

Abstract. In most cases information system development can be seen as an exercise of business process reengineering, either because it automates some human-based processes or because a legacy system is going to be replaced. From this point of view, we can say that the specification of the system-to-be goes from the observation and analysis of the current system to the specification of the system-to-be, going through the construction and evaluation of alternatives. Goal-oriented models are a valuable formalism to support the strategic analysis of the current process. In this paper, we propose a method supporting that round-trip engineering process, focusing in the prescriptive construction of strategic *i** models and the systematic generation of alternatives. Several requirements engineering techniques are used in order to model the existing process, which allow a reliable generation and evaluation of alternatives as well as the reuse of strategic knowledge for information system development.

1 Introduction

Development of information systems is an activity that seldom takes place from scratch. A new information system may automate some tasks that are undertaken by humans in an organization, or may substitute a system that is becoming obsolete from the organizational point of view. Therefore, most of the times we can say that information systems development and business process reengineering are two views of the same activity and therefore we can reconcile them.

From the business process reengineering perspective, the specification of the system-to-be starts from the observation of the current system and the synthesis of its model, the understanding of its rationale, the formulation of new processes or possible ways to enhance the existing ones, the generation and evaluation of alternatives and finally the construction of the detailed target specification itself. We have thus a round-trip from current to ongoing system prescriptive specification, and during this trip we need some support for the intermediate stages: for supporting the strategic analysis of the current system, its weaknesses and strengths, and its alternatives.

The use of the *i** framework in business process reengineering [23] provides an appropriate context where the current process rationale is modelled by means of intentional concepts. The resulting *i** model is the basis for searching and evaluating process alternatives and for obtaining the specification of a new system. Although there already exist several proposals that obtain detailed system specifications from *i** models [3, 13, 22], the problem of creating an *i** model in a prescriptive way is not so often addressed and then, the whole process is at risk because of the lack of reliability in the search and evaluation of process alternatives.

The methodology we propose addresses system development as an exercise of process reengineering and makes a round-trip from the detailed specification of the current system to the detailed specification of the system-to-be, focusing on how to build prescriptive *i** models and generate several alternatives in a systematic manner. Several requirements engineering techniques and artefacts are used in the methodology for obtaining requirements in a systematic way, for describing knowledge, for studying the current processes and so on, with the ultimate goal of supporting *i** modelling and obtaining the models in a more predictable way than usual in order to rely in the application of systematic patterns of evaluation of alternatives. The result of our proposal is a five phases methodology that we have applied to the Meeting Scheduler problem for illustrating the approach.

2 The *i** Language

The i* framework proposes the use of two types of models for modelling systems, each one corresponding to a different abstraction level: a Strategic Dependency (SD) model represents the intentional level and the Strategic Rationale (SR) model represents the rational level.

A SD model consists of a set of nodes that represent actors and a set of dependencies that represent the relationships among them. Dependencies expresses that an actor (*depender*) depends on some other (*dependee*) in order to obtain some objective (*dependum*). Thus, the *depender* depends on the *dependee* to bring about a certain state in the world (goal dependency), to attain a goal in a particular way (task dependency), for the availability of a physical or informational entity (resource dependency) or to meet some non-functional requirement (softgoal dependency).

A SR model allows visualizing the intentional elements into the boundary of an actor in order to refine the SD model with reasoning capabilities. The dependencies of the SD model are linked to intentional elements inside the actor boundary. The elements inside the SR model are decomposed accordingly to two types of links:

- Means-end links establish that one or more intentional elements are the means that contribute to the achievement of an end. The "end" can be a goal, task, resource, or softgoal, whereas the "means" is usually a task. There is a relation OR when there are many means, which indicate the different ways to obtain the end.
- Task-decomposition links state the decomposition of a task into different intentional elements. There is a relation AND when a task is decomposed into more than one intentional element.

For more details about *i**, we refer to [23].

3 Overview of the Methodology

The methodology we propose is related to the business process reengineering context as presented in Eric Yu's thesis [23], which provides an *i**-based framework consisting of four different phases: a model of the process based on intentional concepts, a systematic search for process alternatives, a systematic evaluation of process alternatives with respect to stakeholder interests and, finally, the possibility of connecting strategic reasoning with information system development.

From this starting framework, our methodology makes the following enrichments: a) we add a preliminary phase for domain information gathering (current system specification); b) we focus on how to build a starting *i** model from this information; c) we provide some rationale for the discovering of new strategic needs; d) we drive the systematic generation of strategic alternatives; e) we propose well-defined frameworks to drive the evaluation of alternatives.

All of these contributions have a similar aim, namely to consider business process reengineering as a prescriptive process and reduce therefore the inherent uncertainness that *i**-based reasoning has. Prescriptiveness is supported by means of the formulation of rules, guidelines, patterns and questions which articulate altogether to form a well-defined path. Our final objective is to generate the specification of the system-to-be in a highly reliable and effective way.

In Fig. 1 we present an overview of the methodology, which is composed of the above mentioned five phases. In the first phase, the current process is analysed by using several requirements engineering techniques and a descriptive model of the process is built by observation. In the second phase, an *i** model is constructed to obtain the rationale of the current system. The systematic generation of process alternatives is done in the third phase by means of the addition of new actors and/or the reallocation of responsibilities between them. Those different alternatives are

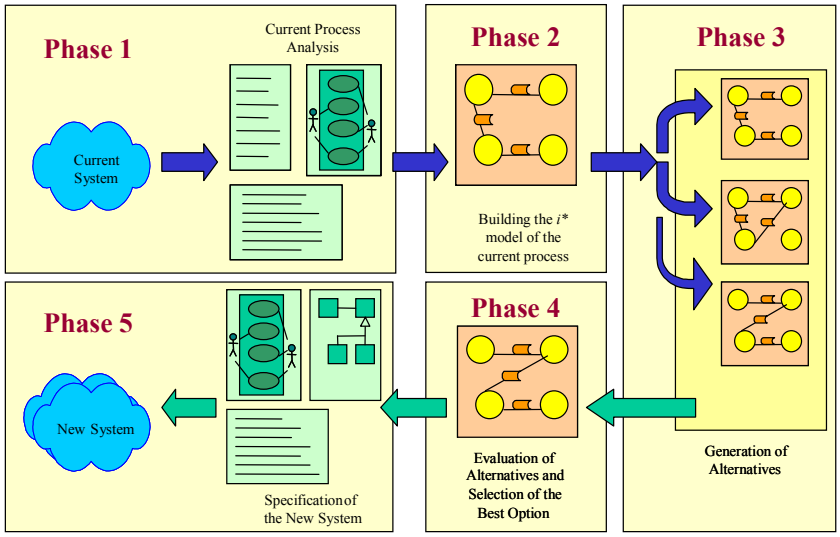

Fig. 1. Overview of the process, showing which process takes place inside each of the phases

evaluated in the fourth phase and one of them selected as solution. Finally, the specification of the new system is generated based on the chosen process alternative.

In the rest of the paper, we explain the steps of each of those phases. As phases 1 and 5 are widely explored in the literature and several proposals exist for phase 4, we focus our attention in phases 2 and 3. We apply the methodology to an example based on the Meeting Scheduler problem statement of Lamsweerde et *al.* [18], which has been chosen because it is well-known on the community and very representative of the kind of problem we are addressing.

4 Phase 1: Analysing the Current Process

The first phase of the methodology consists of capturing and recording information about the different elements of the current process in order to inform further phases. The approach we adopt here comes from the RESCUE method [13, 14], where techniques from both the HCI and RE fields are used to data gathering and human activity modelling from all the components of the current process.

Human activity modelling is centred on the human users involved in the process. Thus, the domain where the process occurs (the individual cognitive and non-cognitive components and social and co-operative of the process) needs to be analysed in order to understand the current process. Data gathering techniques include observation of the current system use; informal scenario walkthroughs; interviews with representative human users; and other similar techniques.

Once the domain of the process is analysed, the process has to be documented. Context models provide a simple approach for modelling system boundaries. They use a basic data flow diagram notation, where a number of concentric circles represent different 'levels' of involvement and interacting with the system. Once system actors are identified, flows of data are represented to or from the different circles depending on the direction of the data flow.

Human activity models (HAM) collect the data from the current process. This data can be documented in several models, being one of them activity descriptions which are structured in several fields providing, among others, the human actors involved in the system, their goals, triggering events, preconditions, assumptions, constraints, normal and alternative courses of actions and resources involved.

In our example, we use the domain theory of the Meeting Scheduler problem statement as if it where the result of that domain analysis phase. We observe that, in that first process, there is no software system to support the scheduling of meetings.

5 Phase 2: Building the *i** Model of the Current Process

The reliable generation and evaluation of alternatives requires that the *i** model of the current process must be developed in a systematic and prescriptive way, otherwise decisions can be made upon incomplete or ill-constructed knowledge. In other words, different people modelling the same process in the same organization should obtain similar results. However, the same process on different organizations could be

modelled in different ways, since the strategic reasons of every organization may change. This is due to the fact that, despite the process that is undertaken, the organizational context plays an important role. This is mentioned in [16], where there is evidence that ignoring this difference between organizational realities and process logistics often causes a mismatch in the process analysis. Thus, two different kinds of goals can be semantically distinguished [1]: *descriptive goals*, appearing on current processes analysis, and *prescriptive goals*, coming from strategic management.

Taking into account those differences, we propose to build the *i** Strategic Dependency (SD) model in two differentiated steps in order to distinguish the functionality performed by the stakeholders (dealing with *descriptive goals*) from their strategic intentionality (*prescriptive goals*). The result is an *i** model with two different parts: an *operational i** model (mainly composed of resources, tasks and some goals) and its associated *intentional i** model (which adds goals and softgoals to the operational one). In order to support this construction method and add rationale to further analysis of the current process, both SD and SR *i** models are developed. The resulting *i** model can be checked for consistency, as proposed in the RESCUE process [13].

5.1 Step 1: Actor Identification and Modelling

The first step for *i** modelling is the identification of the actors and their intentionality. The actors arise from the current process documentation obtained in the previous phase, with one *i** actor for each stakeholder and a single actor for the software system if it exists. If the system is considered to be too large or has well-differentiated parts, we may decompose it using the <<is-part-of>> construct provided by the *i** framework. Other aggregation and also specialization relationships between actors can be identified and included in the model. Due to the unavoidably iterative nature of the process, if new actors appear in later steps, a further iteration of phase 1 is needed in order to take them into account.

Some of the actors identified for the Meeting Scheduler example and their main process goals are: Meeting Initiator (*Schedule the Meeting*), Address Provider (*Provide Addresses*) and Meeting Attendee (*Attend the Meeting*). The Meeting Attendee has two specializations: Active Participant (*Gives a Talk in the Meeting*) and Important Participant (*Must Attend the Meeting*).

5.2 Step 2: Building the Operational *i** Model

In order to be prescriptive when building the *i** model at the operational level, we use both the SD model and the SR models. For obtaining the dependencies we need to explore each of the activities identified in phase 1 (see section 3) by analysing its different actions. An effective way of doing that is to enumerate chronologically all of the actions that need to be executed until completing the activity, making explicit both the actions that the actor performs by itself and the actions that the actors requires from other actors, as mentioned in [16].

Scenarios analysis has proven to be an effective technique for requirements engineering and process analysis. There are several ways of writing scenarios [21] but, as we are only interesting in the *i** model, we propose to use a simplified notation for process scenarios that we call Detailed Interaction Script (DIS). As the scenario information has already been obtained in phase 1, DIS are just intermediate models that organize scenario information in order to facilitate further *i** model construction. A DIS includes goals, actors, preconditions, triggering events and postconditions, which are obtained from the phase 1 HAM. Actions are the atomic actions of human activity diagrams written in a structured way. For each action we state: the actor who initiates the action, a short description of the action, and the resources involved in case that the action produces or consumes a certain resource. If the action requires an interaction with another actor, the actor addressee and the interacted resource are also stated.

In table 1 we can see the DIS of the activity *Invite Meeting Attendees*. We can observe that the decisions taken by the Meeting Initiator such as *Decide Participant List* or *Decide Meeting Data Range* are made explicit. Once the Meeting Initiator has the participant addresses and the initial data range, he sends the invitations to the meeting attendee.

The benefit of using DIS for analysing the scenarios is twofold. On the one hand, the analysis of a piece of process in a chronological way can be easy to perform and tends to yield similar results even if performed by different people. On the other hand, it is possible to translate the information of the table to the *i** model we are building, by following the rules:

- **Rule 1.** Every activity in which an actor is involved is modelled as a task in its SR. This task (hereafter, *activity-task*) is related to its main goal (already identified in step 1) using a Means-Ends link. Activity-tasks are named after the activity they

Table 1. Detailed Interaction Script (DIS) for the activity *Invite Meeting Attendees*

	DIS1: Invite Meeting Attendees					
Source	HAM1: *Invite Meeting Attendees*					
Actors	Meeting Initiator, Address Provider, Meeting Attendee					
Precondition	-					
Triggering Event	-					
	Action Initiator	**Action**	**Consumed Resources**	**Produced Resources**	**Action Addressee**	**Provided Resources**
Actions	Meeting Initiator	Decide Participant List		Participant List		
	Meeting Initiator	Get Participant Addresses			Address Provider	Participant List
	Address Provider	Send Participant Addresses			Meeting Initiator	Participant Addresses
	Meeting Initiator	Decide Initial Data Range		Initial Data Range		
	Meeting Initiator	Send Invitation			Meeting Attendee	Initial Data Range
Postcondition	Meeting Invitation Send to all potential participants					

are related to. Once all the activity-tasks are modelled, we obtain a first level of decomposition on the SR model of each actor. In Fig. 2 both Meeting Initiator and Address Provider, the two actors involved in the activity *Invite Meeting Attendees*, have an activity-task with that name in their SR decomposition.

- **Rule 2.** Every activity-task is decomposed into the actions of the DIS corresponding to its activity. This is done by translating each action into a task (hereafter, *action-task*) and relating this action-task with a task-decomposition link to the corresponding activity-task of the action initiator. In Fig. 2 the activity-task *Invite Meeting Attendees* is decomposed into four action-tasks: *Decide Participant List*, *Get Participant Addresses*, *Decide Initial Data Range* and *Send Meeting Invitations*.
- **Rule 3.** If the action on the DIS produces a resource, this resource becomes a resource dependency where the action addressee is the *depender* and the action initiator, the *dependee*. In the SR model, the dependency is linked to action-task that produces the resource. In Fig. 2 the Address Provided depends on the Meeting Initiator for the produced resource *Participant List*, and is the *dependee* for the consumed resource *Participant Addresses*. The Meeting Attendee depends on the Meeting Initiator for the produced resource *Initial Data Range*, but we do not know the action which consumes it.

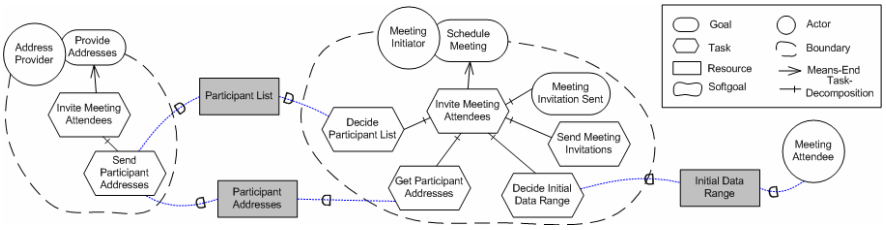

Fig. 2. Piece of the *i** SD model, concerning the dependencies derived from the Use Case *Invite Meeting Attendees*

- **Rule 4.** If the action on the DIS consumes a resource, this resource has to be produced by some other action. Thus, we look for those dependencies produced by other actions and assigned to the actor. If the resource has already been produced, we link it to the specific action-task. If not, we link it when the dependency arises. In Fig. 3 we observe that the action-task *Check Meeting Data Range* consumes the resource *Data Range*.
- **Rule 5.** Every precondition, trigger event and postcondition of the activity has to be explicitly modelled. As they represent the achievement of a certain state, they are modelled as goals in the *i** model. Preconditions and postconditions are added as task-decomposition elements of the activity-task. Trigger events are modelled as goal dependencies where the actor who initiates the activity-task is the *depender* and the one that undertakes the triggering task is the *dependee*. In Fig. 3 the postcondition of the activity *Invite Meeting Attendees* is modelled as the goal *Meeting Invitation Sent*. The task *Send Meeting Invitation* triggers the activity *Provide Data Sets*, and thus, the Meeting Attendee has a goal dependency on *Meeting Invitations Received*.

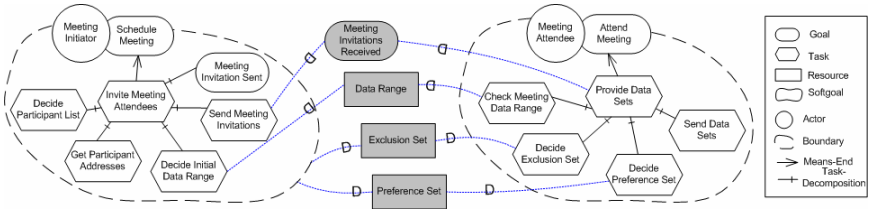

Fig. 3. Piece of the *i** SD model, concerning the dependencies derived from the Use Case *Provide Data Sets*, which is linked with *Invite Meeting Attendees*

All the proposed rules can be performed systematically, and there is only one aspect in which the process is not prescriptive. Rules translate resources to resources dependencies but, sometimes, task dependencies are more suitable. To differentiate among them we should think about what it is more important for the *dependee*, the resource sent within the parameter (*resource dependency*) or the way that resource is obtained (*task dependency*).

In [19] a methodology for building *i** models based on activities theory is presented. This proposal builds SD models before SR ones, whilst we build it in parallel but, as it also analyse activities and their actions to construct the model, the approach may seem similar to the one we present (rules 1 and 2). Nevertheless, there are some differences in the way the actions are analysed, as [19] does not provide any specific analysis of the resources involved in the actions (rules 3 and 4) and neither consider preconditions, nor postconditions nor triggering events (rule 5). Also [19] does not present any methodology for building the intentional model, as we do in the next step.

5.3 Step 3: Building the Intentional *i** Model

The intentional *i** model complements the operational *i** model and contains the intentionality behind the analysed process. Its construction entails more uncertainty but it can also be performed in a systematic way.

First, an initial set of *prescriptive goals* is obtained directly from the current process. Assuming that each of the studied activities represents the achievement of a goal, strategic goals are obtained as a response to the following question:

- Which is the final state to achieve by executing the activity?

The *dependee* and the *depender* of that goal arise by asking respectively:

- Which is the actor that needs to attain the goal?
- From which actor it depends to obtain the goal?

In the Meeting Scheduler example, the execution of the activity *Find agreeable date* leads to achieve the final state *Maximum number of Attendees in the Meeting* where is the Meeting Initiator who depends of the Meeting Attendee to achieve it. We remark that this goal is not the postcondition of the activity, which is the goal *Agreeable Date Found*. Actually, we consider *Find Agreable Date* as a means of obtaining the *Maximum number of attendees in the meeting* and we state a means-end relationship between these two elements. In Fig. 4 we can observe that we reorganize the hierarchy of the SR model in order to represent this insertion in the *i** SR model

of the Meeting Initiator. Thus, *Maximum number of attendees in the meeting* become a means of the goal *Schedule Meeting*. Moreover, the task Invite Meeting Attendees is also a means of having a *Maximum number of attendees in the meeting* and a means-end is also stated between the two.

Second, this initial set of descriptive goals and the already existing *i** operational model are the basis for obtaining new goals. Stakeholders shall participate in this process by providing information needed to apply existing techniques. Those goals can be decomposed and operationalized into constraints [5]. Thus, the goal *Maximum number of Attendees in the Meeting* can be means-ends related to the goal *Respect Attendee Data Range* and the softgoal *Meeting Details Announced promptly*. The *dependee* of those goals is the Meeting Attendee (see Fig. 4).

Also a directed inquire analysis of the process helps us determine that goals [20] and stakeholders interview techniques can also be applied [11]. We propose to apply those techniques by analyzing the operational *i** model with the stakeholders and formulate the questions on the *i* dependum*s and intentional elements. In the KAOS methodology [5], goals are classified into satisfaction, information, robustness, consistency, safety and privacy goals. Likewise classifications made in the NFR framework [4] are applicable. Following these ideas, we propose the use of a quality attributes catalogue like the ISO/IEC 9126-1 [12] to generate questions automatically and retrieve answers easily. Quality attributes may be related to the type of intentional element, for instance if the analysed element is a resource we can ask about data security or data accuracy whilst if it is a task, questions about efficiency or usability are more appropriate.

Most of the quality attributes refer to important goals and softgoals, but we are only interested in the most crucial ones. Thus the questions are formulated in terms of how critical is the attribute for the element of the process. For instance, in the operational *i** model for the Meeting Scheduler, the resource *Participant Addresses* can be analyzed by asking the questions:

- If *Participant Address* data privacy is violated, can someone get hurt or damaged? Which actors will be affected?
- If *Participant Address* is not accurate enough, will the process fail? Which actors will be affected?

The concrete writing of questions can be associated with the catalogue itself, therefore providing a systematic way to generate goals and softgoals. In Fig. 4 we state that

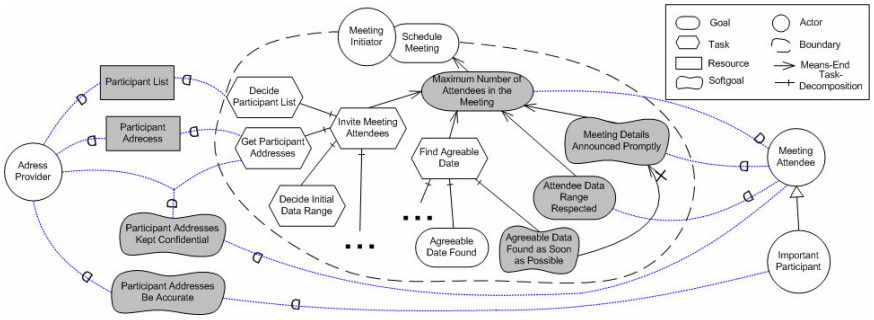

Fig. 4. Piece of operational and intentional *i** model for the Meeting Scheduler

the address privacy of the Meeting Attendee is crucial and therefore introduce a softgoal *Keep Participant Address Confidential*. The attendee depends on both Address Provider and Meeting Initiator for ensuring this. However accuracy is not so critical in general, the Meeting Attendee has not any dependency about that (we consider that in case of being inaccurate, probably he will notice about the meeting somehow –chatting, by chance, etc.- and even if not so, his absence would not compromise the occurrence of the meeting). However there is an exception. Important participants must attend the meeting or else the process fails, thus the softgoal dependency *Participant Address Be Accurate* appears for this actor.

Questions can also be applied on the SR model in a similar manner. In Fig. 4, the task *Find Agreeable Date* is further decomposed into the softgoal *Agreeable Date found as Soon as Possible*, as an answer to the question:

- If *Find Agreeable Date* is not performed efficiently in time, can the process fail?

Finally, the resulting set of goals and softgoals can be analyzed in order to identify contributions and conflicts between the different intentional elements by means of the *i** contribution link*s*. This analysis can be done as proposed in the NFR framework [4] or considering the relationships between quality attributes already stated in the quality model [8]. In Fig. 4, the softgoal *Agreeable Date found as Soon as Possible* contributes positively to the softgoal *Meeting Details Announced promptly*.

5.4 Step 4: Checking the *i** Model

The intentional elements and *dependums* of the intentional *i** model can be checked for consistency on the application of the provided rules and guidelines. Fig. 5 presents a meta-model with the simplified concepts of the three models used: HAM, DIS and *i** models. The baseline concepts mappings across those models are defined with thicker horizontal lines. Thus, meta-model maps actor goals in human activity model to conditions in DIS and *i** goals and soft goals. As the DIS are a structured version of HAM, there is a direct mapping between the concepts of activities, actions and actors between both models, whilst the resources in DIS can be of three types: produced (p), consumed (c) or provided (pv). DIS activities are modelled into *i** tasks and actions are mapped into resources or tasks in the *i** operational model.

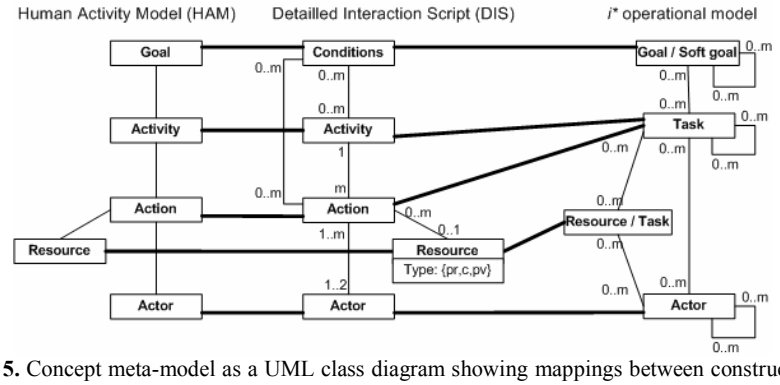

Fig. 5. Concept meta-model as a UML class diagram showing mappings between constructs in the 3 model types

The checking is done in two stages. The first stage ensures correspondence between HAM and DIS with the following checks:

Check 1.1	Every activity on the HAM should correspond to one or more activities in the DIS.
Check 1.2	Every human activity goal should correspond to one or more conditions (goals or postconditions) in the DIS.
Check 1.3	Every preconditions and assumptions on the HAM, should correspond to preconditions in the DIS. Triggering events appear to be the same in both diagrams.
Check 1.4	Every actor of the HAM is mapped into one ore more actors in the DIS.
Check 1.5	All actions on the HAM normal course and alternatives courses, should correspond to actions in DIS detailing the action initiator and, if required, also the action addressee.
Check 1.6	All resources appearing in the HAM actions, should be detailed as produced, consumed or parameter resources in the DIS.

At the second stage, cross checking is done in order to ensure the correspondence between the DIS and the resulting *i** models. Checks are:

Check 2.1	Every DIS activity is modeled as a task, called activity-task.
Check 2.2	Every main goal of an actor is means-end decomposed into those activity-tasks where the actor performs an action.
Check 2.3	Every action inside an activity is modeled as a task, which decomposes the corresponding activity-task on the SR model of the actor that initiates the action.
Check 2.4	Every provided resource involved in an interactive action, appears as an SD resource dependency or task dependency between the actors involved in the interaction, where the *depender* produces the resource and the *dependee* consumes it.
Check 2.5	Conditions of the activity are modeled as SR-goals (for preconditions and postconditions) and goal dependencies (for trigger events).
Check 2.6	Each activity-task is means-end decomposed into its main intentional goal, which can be refined into other goals.
Check 2.7	Some non-functional constraints are stated over the resources and the task, leading to softgoals both in the SR and the SD model.

6 Phase 3: Generation of Alternatives

One of the main strengths of goal-oriented modelling is its adequacy for exploring different ways to achieve strategic aims. This can be seen in many proposals. The *i** framework itself [23] seeks systematic searches for process alternatives by using means-end reasoning and hierarchical decomposition of tasks into their intentional elements. The TROPOS project [3] defines the architectural organization of the system by exploring alternatives whilst introducing new actors. Those actors are defined according to the choice of a specific architectural style and the benefits that they provide for the fulfilment of some specific functional and non-functional requirement. In the KAOS approach [5] the identification of alternative responsibilities and the assignation of actions to responsible agents is faced in the lasts phases of their goal-directed acquisition strategy.

Taking those concepts as a starting point, we propose to use the *i** model obtained in phase 2 as a basis for obtaining new alternatives, also modelled in *i**. Each of the alternative process is constructed from the same model, which is the one corresponding to the current process (generated in the previous phase) plus changes coming from the improvements that arise during the strategic analysis made in the context of process reengineering. In other words, new goals and softgoals can be

added to the *i** model of the current system before beginning the exploration of alternatives. Alternatives are generated by adding new actors to the system and reallocating responsibilities between them.

6.1 Step 1: Reengineering the current system

As the final purpose of the method is to achieve process reengineering, stakeholders may want to improve some aspects of the current process and some new goals and softgoals may arise, disappear or change. KAOS [5] proposes to drive this process by applying patterns that have an impact on the system behaviour: *Achieve* and *Cease* goals generate behaviours, *Maintain* and *Avoid* goals restrict behaviours, whilst *Optimize* goals compare behaviours. We use these patterns for analysing the SD *i** model with the stakeholders in the following way:

1. First we restrict the behaviour of the current process by analysing the intentional goals and softgoals. We classify them into two groups: the ones we want to *Maintain* and the ones we want to *Avoid*. If a goal has to be avoided, new goals and softgoals arise in order to state that.
2. This new set of goals is analysed in order to generate new behaviour. Thus, we search for new goals that we want to *Achieve* or old ones we want to *Cease*. If the new goals to *Achieve* involve the addition of new activities to the process, a DIS is created for the analysis of the system and the *i** model is completed with new SD dependencies and SR intentional elements by applying the steps on phase 2. A goal can only be *Ceased* if it does not affect the achievement of another goal. In terms of activities, this means that if some of the actions it involves are preconditions or trigger events of another activity, it cannot be removed unless the other activities are also removed.
3. Finally, *Optimize* goals are added in order to compare the behaviour. Questions such as the ones we proposed in the step 3 of phase 2, have to be applied.

All the alternatives must be generated from the same starting model. Thus, no other dependencies can arise during the following steps. In the case that some new dependencies need to be added, they should be consistent with the patterns above and the process of generating alternatives has to be started again.

6.2 Step 2: Adding New Actors to the System

The addition of actors to the system is done by means of exploring new roles to fulfil in the process. This exploration is done by means of analysing current solutions to similar processes such as organization structures or software solutions. Thus, new actors do not arise from combinatorial explosion, but from a rationale analysis.

One possibility is to apply organizational patterns in order to explore the application of well-known solutions. There exist some proposals of social patterns and organizational styles defined in terms of configurations of *i** concepts [17]. As the intention is to obtain a new process, software roles are also likely to arise. For finding which software roles are more convenient we recommend using components catalogues or, even better, taxonomies of COTS components [2].

For each added actor, we need to decide its main goal. New actors are added in both operational and intentional *i** models. In Fig. 6 we have added a Meeting Scheduler actor. Its main goal is *Meeting Be Scheduled* and, thus, the Meeting Initiator main goal has changed to *Meeting Scheduler Ordered*.

6.3 Step 3: Reallocating Responsibilities

Once the new actors have been discovered, the existing dependencies must be reallocated. We use the activity-tasks in a similar way as done in [6] restricted to tasks and resources, where some guidelines are provided to guide the insertion of an actor system into a business model.

Each activity task is analysed independently by asking the questions: Do we want this actor to keep satisfying the dependencies of the activity task? Is there any other actor that can take that responsibility? Human capabilities may be checked and software components functionalities may be matched [7] to answer that question, and depending on this answeer, one of the following two patterns are applied:

Pattern 1. We consider that the responsibility of the tasks still falls onto the current actor and, thus, every dependency related to their activity-task remains unchanged. In Fig. 6 we leave the responsibility of *Provide Data Sets* to the Meeting Attendee and none of the dependencies related to that task changes.

Pattern 2. We delegate the responsibility of a task. Thus, if actor A was performing an activity and we want actor B to do it, we delegate the responsibility of the corresponding activity-task to actor B taking into account the following aspects:

- A new activity-task with that name is added to the Means-End decomposition of the main goal of the actor B and all the dependencies related to their action-tasks are moved into the other actor. For instance, in Fig. 6 the responsibility of the task *Find Agreeable Date* goes to the Meeting Scheduler actor, who handles the dependencies going to/stemming from the part of the SR model for the Meeting Initiator of the previous model (see Fig. 3).
- The action-tasks of the reallocated activity-task are checked in order to ensure that actor B has all the knowledge and capabilities to undertake them. If actor B cannot

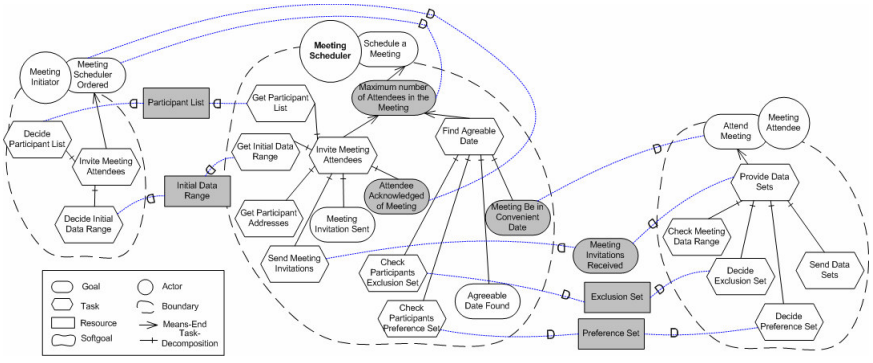

Fig. 5. Piece of i* model showing responsibility reallocation into the Meeting Scheduler actor

fulfil them, we add a dependency to the actor who can better provide it, which is usually actor A. In Fig. 6, we observe that the responsibility of *Send Meeting Invitation* is now going to the Meeting Scheduler. As he needs the *Participant List* and the *Data Range* to perform that task, we add two task dependencies to the Meeting Initiator who is the actor who knows this information.

- Finally, a goal dependency arises to model that actor A depends on actor B to achieve its intentional goals on the activity. Thus, in the example, the Meeting Initiator depends on the Meeting Scheduler for accomplish the goal postconditions of the activities *Attendee Acknowledged of Meeting* and *Maximum number of Attendees in the Meeting* (see Fig. 6).

6.4 Step 3: Checking consistency between alternatives

Consistency checks can also be applied to ensure certain equivalence between the original *i** model and the *i** model for the generated alternative:

Check 3.1	Every activity-task on the original model is also considered in the alternative model.
Check 3.2	In the alternative, every actor that is responsible for an activity-task has all the knowledge to perform all the actions (by itself or by means of a dependency to another actor)
Check 3.3	The intentional goals and softgoals of an actor in the original model, still being satisfied by means of dependencies to the actor that performs the activity-task in the alternative model.

7 Phases 4 and 5: Evaluating Alternatives and Defining the New System

The systematic evaluation of process alternatives with respect to stakeholder interests and the connection of the provided strategic reasoning with information system development are already addressed issues in literature. In consequence, as already mentioned for phase 1, these two last phases do not enforce the adoption of any specific methods. Instead, the most adequate to each situation should be chosen.

The systematic evaluation of process alternatives is already addressed in Yu's work [23]. In the origin of the *i** framework, the SD model supports the systematic identification of stakeholders and their interests and concerns, whilst the SR model supports the systematic evaluation of alternatives through the concepts of ability, workability, viability, and believability. Also, the AGORA method [15] provides techniques for estimating the quality of requirements specifications in a goal-oriented setting. In [9, 10] structural analysis of actor-dependency models is performed by defining metrics over the models with respect some properties considered of interest for the modelled system. (such as security, accuracy or efficiency). Dependencies are used to analyse the behaviour of the system.

The link between strategic reasoning and information system development has been widely addressed. Several proposals exist providing guidelines for mapping an *i** model to an UML use cases and classes specification, among them we remark [3, 13, 22].

8 Conclusions

We have proposed a methodology that addresses system development as an exercise in process reengineering that makes a round-trip through five different phases: domain information gathering, specification of the current system in *i**, systematic search for process alternatives, evaluation of the modelled alternatives and, finally, a prescriptive specification of the system-to-be.

We think that the main contributions are the following. First, we give a well-defined method for system development which is more prescriptive and has more level of detail than usual for so comprehensive methods. Other proposals that have this level of detail are more local than ours. Detail and prescription is achieved by means of rules, guidelines, checks, patterns and answer-questions.

Second, we use existing requirement engineering techniques when possible. We are putting together concepts from goal-oriented modelling (e.g., KAOS patterns, *i** language, organizational patterns, etc.), specification (human activity models), knowledge gathering (inquiry cycle, …) and others. In addition we are providing innovations as needed. Distinguishing among operational and organizational *i** models when analysing the current systems helps in making the process more guided. The use of a quality attributes catalogue as the ISO/IEC 9126-1 for driving softgoal identification and the clear cut-criteria to discern if they are needed are helpful to add rationale to the model. Our treatment of resources elements is very precise. The ordering of the KAOS pattern, although yet to be thoroughly validated, is another example of how necessary we find it is to identify efficient ways to build *i** models.

Acknowledgements

This work has been partially supported by the CICYT programme, project TIN2004-07461-C02-01. G. Grau work is supported by an UPC research scholarship.

References

1. Antón, A.I., McCracken, W.M., Potts., C.: "Goal Decomposition and Scenario Analysis in Business Process Reengineering". In *Proceedings of the 6th International Conference on Advanced Information Systems Engineering*, 1994. pp. 94-104.
2. Ayala, C.P., Botella, P., Franch, X.: "On Goal-Oriented COTS Taxonomies Construction". In *Proceedings of the 4th International Conference on COTS Based Software Systems*, 2005. Springer-Verlag, LNCS 3412. pp. 90-100.
3. Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., Mylopoulos, J.: "Tropos: An Agent-Oriented Software Development Methodology". In Journal of Autonomous Agents and Multi-Agent Systems. Kluwer Academic Publishers, Vol 8, Issue 3, 2004. pp. 203-236.
4. Chung, L., Nixon, B., Yu, E., Mylopoulos, J.: *Non-Functional Requirements in Software Engineering*. Kluwer Academic Publishers, 2000.
5. Dardenne, A., van Lamsweerde, A., Fickas, S.: "Goal-directed Requirements Acquisition", Science of Computer Programming, 20, 1993, pp. 3-50.

6. Estrada, H., A. Martínez, A., Pastor, O.: "Goal-based business modeling oriented towards late requirements generation". In *Proceedings of the 22nd International Conference on Conceptual Modeling*, 2003. Springer-Verlag, LNCS 2813. pp. 277-290.
7. Franch, X.: "On the Lightweight Use of Goal-Oriented Models for Software Package Selection". In *Proceedings of the 17th International Conference on Advanced Information Systems Engineering*. 2005. Springer-Verlag, LNCS 3520. pp. 551-566.
8. Franch, X., Carvallo, J.P.: "Using Quality Models in Software Package Selection". IEEE Software, 20(1), 2003, pp. 34-41.
9. Franch, X., Grau, G., Quer, C.: "A Framework for the Definition of Metrics for Actor-Dependency Models". In *Proceedings of the 12th IEEE International Conference on Requirements Engineering*, 2004. pp. 348-349.
10. Franch, X., Maiden, N.A.M.: "Modeling Component Dependencies to Inform their Selection", In *Proceedings of the 2nd International Conference on COTS Based Software Systems*, 2003. Springer-Verlag, LNCS 2580. pp. 81-91.
11. Goetz, R., Rupp, C.: "Psychotherapy for System Requirements". In *Proceedings of the 2nd IEEE International Conference on Cognitive Informatics*, 2003. pp. 75-80.
12. ISO/IEC Standard 9126-1 Software Engineering – Part 1: Quality Model, 2001.
13. Jones, S., Maiden, N.A.M.: "RESCUE: An Integrated Method for Specifying Requirements for Complex Socio-Technical Systems". Book chapter in *Requirements Engineering for Sociotechnical Systems*, Idea Group Inc., 2004.
14. Jones, S., Maiden, N.A.M., Manning, S., Greenwood, J.: "Human Activity Modelling in the Specification of Operational Requirements: Work in Progress". In *Proceedings of the Workshop Bridging the Gaps between Software Engineering and Human-Computer Interaction*, 2004.
15. Kaiya, H., Horai, H., Saeki, M.: "AGORA: Attributed Goal-Oriented Requirements Analysis Method". In *Proceedings of the 10th IEEE International Conference on Requirements Engineering*, 2002. pp. 13-22.
16. Katzenstein, G., Lerch, F.J.: "Beneath the Surface of Organizational Processes: A Social Representation Framework for Business Process Redesign". ACM Transactions on Information Systems, Vol.18, No. 4, 2000. pp. 383-422.
17. Kolp, M., Giorgini, P., Mylopoulos, J.: "Organizational Patterns for Early Requirements Analysis". In *Proceedings of the 15th International Conference on Advanced Information Systems Engineering*, 2003. Springer-Verlag, LNCS 2681. pp. 617-632.
18. van Lamsweerde, A., Darimont, R., Massonet, P.: "The Meeting Scheduler Sytem – Problem Statement". 1992. http://www.lore.ua.ac.be/Teaching/SSPEC2LIC/MeetingScheduler.pdf.
19. Neto, G.C., Gomes, A.S., Castro, J.B.: "Mapeando Diagramas da Teoria da Atividade em Modelos Organizacionais Baseados em i*". In *Proceedings of the 7th Workshop em Engenharia de Requisitos*, 2004, pp 39-50.
20. Potts, C., Takahashi, K., Antón, A.I.: "Inquiry-Based Requirements Analysis". IEEE Software, 11(2), 1994. pp. 21-32.
21. Rolland, C., Ben Achour, C., Cauvet, C., Ralyté, J., Sutcliffe, A., Maiden, N.A.M., Jarke, M., Haumer, P., Pohl, K., Dubois, E., Heymans, P.: "A Proposal for a Scenario Classification Framework". Requirements Engineering Journal. Vol 3.No 1. 1998. pp.23-47.
22. Santander, V.F.A., Castro, J.F.B.: "Deriving Use Cases from Organizational Modeling". In *Proceedings of the 10th IEEE Requirements Engineering Conference*, 2002. pp 32-39.
23. Yu, E.: *Modelling Strategic Relationships for Process Reengineering*, PhD. thesis, University of Toronto, 1995.