From E-R to “A-R” – Modelling Strategic Actor Relationships for Business Process Reengineering*

Eric S. K. Yu and John Mylopoulos

Department of Computer Science, University of Toronto
Toronto, Ontario, Canada M5S 1A4

Abstract. As information systems are increasingly being called upon to play vital roles in organizations, conceptual modelling techniques need to be extended to relate information structures and processes to business and organizational objectives. We propose a framework which focuses on the modelling of strategic *actor relationships* (“A-R”) for a richer conceptual model of business processes in their organizational settings. Organizations are viewed as being made up of social actors who are *intentional* – have motivations, wants, and beliefs – and *strategic* – they evaluate their relationships to each other in terms of opportunities and vulnerabilities. The framework supports formal modelling of the network of dependency relationships among actors, and the systematic exploration and assessment of alternative process designs in reengineering. The semantics of the modelling concepts are axiomatically characterized. By embedding the framework in the Telos language, the framework can also potentially serve as an early-requirements phase tool in a comprehensive information system development environment.

1 Introduction

The need to model and understand the organizational or business environment within which an information system is intended to operate is well recognized (e.g., [2, 1]). The recent concept of business reengineering further highlights the need to relate information systems to business objectives. When used innovatively, information technology can bring about dramatic improvements in organizational performance, such as increased speed, reduced costs, and improved quality and service. By enabling people to work in ways that were not possible before, information systems often play key roles in reengineered business processes (e.g., [18, 9, 15]).

Conceptual modelling techniques can potentially be applied to help understand and redesign business processes. Basic concepts for modelling the world such as *entities, activities, assertions,* and *time* have been formalized in a number of modelling frameworks (e.g., [12, 10, 22, 31]). However, to more fully support the types of knowledge and reasoning involved in business redesign, a specialized ontology with additional concepts would be helpful.

In order to understand a business process, it is often not enough to know what entities exist, what activities occur, and what relationships hold, but also *why* they

* in *Entity-Relationship Approach (ER’94) — Business Modelling and Re-Engineering, (Proc. 13th Int. Conf. on the Entity-Relationship Approach, Manchester, U.K., December 1994) Springer-Verlag, LNCS-889, pp. 548–565.*

exist, occur, or hold. In the reengineering literature, it has been argued that without an understanding of *why* things are done the way they are, one is likely to use computers simply to automate outdated processes, and thus unable to realize the true potential that information technology has to offer [14].

Business processes, unlike processes that are executed by machines, exist in social organizational settings. Organizations are made up of social *actors* who have goals and interests, which they pursue through a network of relationships with other actors. A richer model of a business process should therefore include not only how work products (entities) progress from process step to process step (activities), but also how the actors performing these steps relate to each other *intentionally*, i.e., in terms of concepts such as goal, belief, ability, and commitment. When an organization seeks new ways for organizing work, actors who have goals and interests are likely to evaluate these proposal *strategically*, e.g., in terms of potential opportunities and threats. A model for supporting business process reengineering should be able to express and support reasoning about these types of intentional and strategic *actor relationships* ("A-R").

In this paper, we present the i^* framework (pronounced *i-star*) for modelling intentional, strategic actor relationships. The framework consists of two main components. The *Strategic Dependency* (SD) model describes a business organization in terms of the dependencies that actors have on each other in accomplishing their work. It is used to represent a particular design for a business process. The *Strategic Rationale* (SR) model describes the reasoning that actors have about the different possible ways of organizing work, i.e., different configurations of Strategic Dependency networks. It is used to assist actors in understanding the existing process, and to systematically generate alternatives in order to arrive at new process designs that better address business objectives and private concerns.

Earlier versions of the framework has been presented in the context of requirements engineering [33], business process reengineering [35, 36], software process modelling [37], and analysis of the organizational impact of computing [34]. This paper extends earlier work by defining the features of the SR model and giving the highlights of its formalization. It also further clarifies how the framework assists in the understanding of business processes, and the generation and evaluation of alternatives. A popular reengineering example from the goods acquisition domain (from [14]) is used to illustrate the framework throughout.

In section 2, we briefly review the features of the SD model. Section 3 presents the features of the SR model. In section 4, we illustrate how the framework can be used to assist in a business process reengineering effort. In section 5, we present some highlights of the semantics that underlie the modelling concepts, and their formal representation in the conceptual modelling language Telos. In section 6, we discuss our approach and compare it with related work. We conclude in section 7 by placing this work in the larger context of E-R and conceptual modelling and outline some future directions.

2 The Strategic Dependency (SD) Model

A common way of describing a business process is by identifying the work products that flow from one work unit to another. These are often called work flow models (Figure

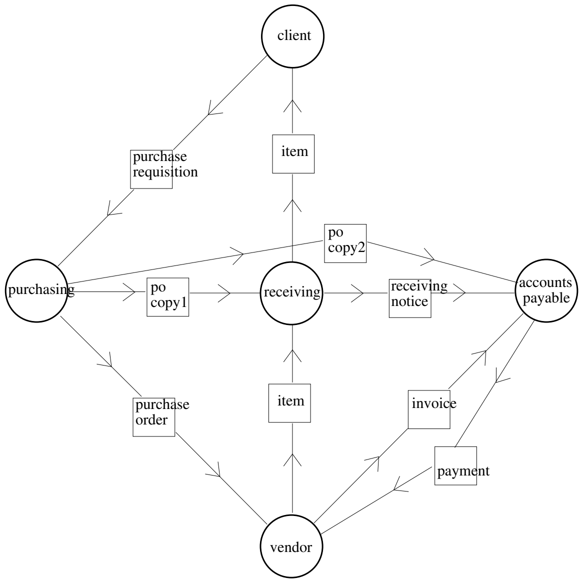

Fig. 1. "Work flow" model of a goods acquisition process

1). More detailed models would identify activities within each unit.

These models typically describe what entities (and relationships) exist in an organization, what activities occur, and what conditions hold at what time, but cannot express why. They are non-intentional in that actors or work units in these models are not taken to have motivations, intents, or rationales.

In a Strategic Dependency model, actors are taken to have goals, and use means-ends knowledge in attempting to achieve goals. In an organizational setting, actors are able to achieve many things that they are unable to achieve in isolation. Each organizational actor depends on others for some part of what it wants, and are in turn depended on by others. One consequence of this is that they are no longer entirely free to choose their own goals or actions.

Figure 2 shows a Strategic Dependency model for the goods acquisition example. A client depends on purchasing in order to have an item. Purchasing depend on the vendor to deliver the item, and on receiving to receive it. The vendor depends on accounts payable for payment, while accounts payable depends on purchasing information, receiving status, and the invoice.

A Strategic Dependency model is a graph, where each node represents an *actor*, and each link between two actors indicates that one actor depends on the other for something in order that the former may attain some goal. We call the depending actor the *depender*, and the actor who is depended upon the *dependee*. The object around which the dependency relationship centres is called the *dependum*.

By depending on another actor for a dependum, an actor is *able* to achieve goals that it was not able to do without the dependency, or not as easily or as well. At the same time, the depender becomes *vulnerable*. If the dependee fails to deliver the dependum,

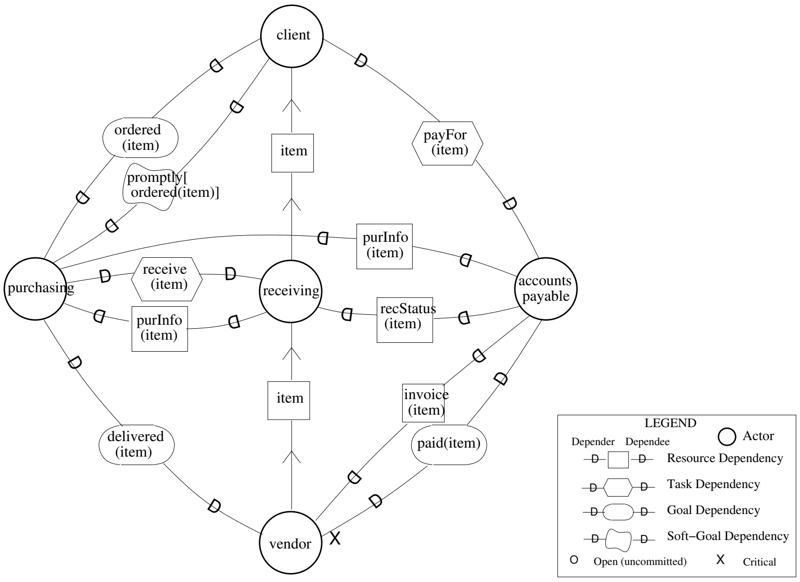

Fig. 2. A Strategic Dependency model of a goods acquisition process

the depender would be adversely affected in its ability to achieve its goals.

We distinguish among four types of dependencies, based on the type of the dependum. In a *goal dependency*, an actor depends on another to bring about a condition in the world. The dependum (the goal) is an assertion that the dependee will make true. The dependee is free to choose *how* to accomplish the goal. The depender is only interested in the outcome. In a *task dependency*, an actor depends on another to carry out an activity (the dependum). The activity specification constrains the choices that the dependee can make regarding *how* the task is to be performed. Typically, this is expressed in terms of the components of the tasks and their interrelationships. In a *resource dependency*, an actor depends on another for the availability of an entity. Entities represent objects in the world. They can be physical or informational.

A *softgoal dependency* is a hybrid of goal and task dependency. An actor depends on the dependee to bring about a condition in the world, but the criteria is not sharply defined as in the case of (hard-)goal dependency. Typically, the dependee has a number of ways for achieving the goal. The depender indicates which combination of choices would sufficiently meet the desired softgoal. We say that a softgoal is *satisficed* rather than satisfied [5].

A dependency can be *open*, *committed*, or *critical*, reflecting the degree of dependency [35].

A Strategic Dependency model presents a richer picture of an organization than conventional workflow models that are based on non-intentional entity and activity relationships. If an item is not received, or payment is not forthcoming, one could not infer from a workflow model what activities might ensue, unless these are explicitly specified. In an intentional model, because actors are taken to be goal-oriented and have freedom to choose actions (decision-making) within limits, one could infer what actors might do without all details being explicitly described.

A business process would typically appear as a chain of dependency relationships, rather than as a sequence of input-output flows. However, in an intentional dependency model, many additional relationships can be expressed, covering associated concerns such as risks and incentives. A dependency need not have an accompanying flow. These other concerns are not usually regarded as part of a process per se, although they are often crucial to the success of a process, and therefore should be modelled.

To model complex patterns of social relationships, the SD model differentiates the generic concept of actor into roles, positions, and agents. A *role* is an abstract actor. Concrete, physical *agents* such as human beings (or software agents) play roles. A *position* is a collection of roles that are typically played by a single agent. Roles, agents, and positions can be related by intentional relationships, besides being associated by the *plays*, *occupies*, and *covers* relationships. For example, an agent can have an expectation on a position that it offers good opportunities for career advancement [37]. The different types of actors, as well as dependums, are organized using conceptual modelling dimensions such as classification, generalization, and aggregation.

3 The Strategic Rationale (SR) Model

While the Strategic Dependency model provides a description of external relationships among actors, it hides the relationships that are inside an actor, e.g., how incoming dependencies (for which the actor is dependee) are related to outgoing dependencies (actor is depender). In the Strategic Rationale model, we model the internal relationships within an actor, so that we can describe and support actors' reasoning about their external relationships.

We show *how* an actor meets its incoming dependencies (or internal goals and desires) by modelling actor's "ways of doing things" – which we call tasks. A task is broken down into its components. Components are broken down into sub-components, and so forth. However, unlike in the conventional, non-intentional modelling of activities and their decomposition into sub-activities, the SR model recognizes the presence of freedom and choice at each level of decomposition. Each component of a task is an *intentional element*, the internal counterpart to the concept of *dependum* in the SD model. An intentional element (or simply element) can be a *goal*, a *task*, a *resource*, or a *softgoal*.

Since there can be more than one way to achieve a goal, to perform a task, to produce a resource, or to satisfice a softgoal, we introduce an intervening *means-ends link* between an element (the end) and each way (the means) of decomposing it into sub-elements. For example, to have an item ordered, one could order by phone, or one could order by issuing a purchase order (Figure 3).

An actor need not address incoming dependencies entirely by its own effort. Intentional elements can be delegated to other actors by way of outgoing dependencies. For example, a third way to have an item ordered is to have it done by a purchasing specialist.

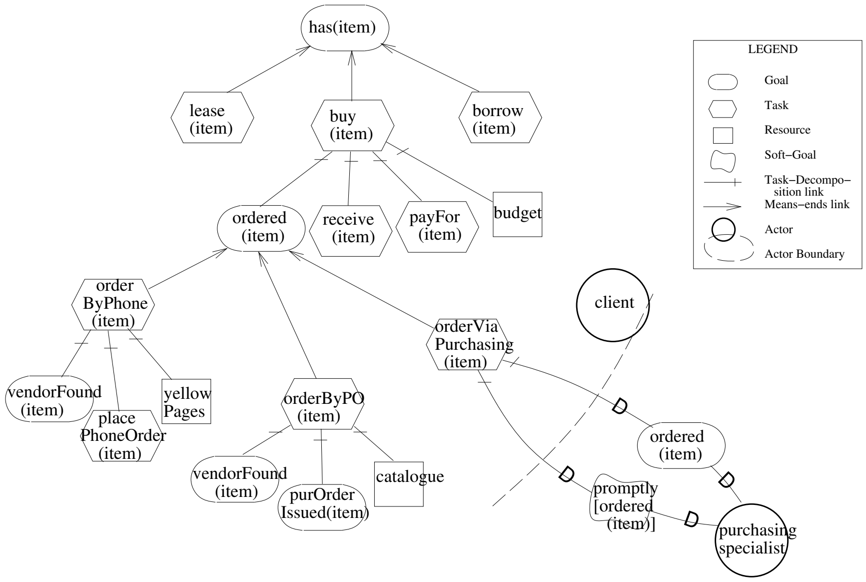

Fig. 3. A Strategic Rationale model showing alternative ways of accomplishing "having an item"

A Strategic Rationale model is a graph. There are four main types of nodes – *goal*, *task*, *resource*, and *softgoal* – and two main types of links – *means-ends links* and *task decomposition links*. Subtypes of means-ends links are based on the type of the nodes that the link connects. For example, a Goal-Task link is a means-ends link with a task as the means and a goal as the end.

A task decomposition link can be a *subgoal*, *subtask*, *resource*, or *softgoal* link. For each type of task decomposition link, there is a corresponding type of dependency link. For example, when a subgoal is delegated, the link becomes a goal dependency link. A task decomposition link or dependency link can be *open* or *committed*. There can be *constraints* amongst components of a task, such as temporal precedence. These are expressed in the formal notation (the assertion language of Telos), but are not shown in the graphical presentation.

We use the term *routine* to refer to a hierarchy of successive decompositions and means-ends reductions which includes only one alternative at each choice point. For example, buying an item by having a purchasing specialist order it is one routine for achieving the goal of having an item (see Figure 3). Another routine might involve borrowing it through some particular channel.

Means-ends links are seen as applications of generic means-ends relationships that are potentially applicable in other contexts. We use the term *rule* to refer to a generic means-ends relationship.

In trying to come up with innovative ways for reorganizing work, the focus is on key elements that would make significant differences when comparing new proposals to the existing process and amongst each other. It would be counter-productive for a modelling scheme to require, at the process design stage in a reengineering effort, an exhaustive specification of how an actor does it work. Hence, in the SR model, we do not assume that each task decomposition provides the complete list of components. The components included are those that are sufficiently significant ("strategic") to warrant attention during the process design/redesign stage. Elements that are left out are assumed to be those that can be dealt with unproblematically by the actor at the time of task execution, and therefore have no strategic implications. We call these *primitively workable* elements. In the development of routines for understanding or exploration of alternatives, the *workability* of a routine is evaluated recursively from the workability of its elements.

Softgoals are treated a little differently from the other three types of intentional elements. Softgoals provide a qualitative assessment scheme on top of the rudimentary assessment of workability. A softgoal is typically a quality (or non-functional) attribute on one of the other intentional elements in a routine, e.g., that a payment be issued *promptly*. Pay-when-invoiced and pay-when-goods-received are two different ways of making payment. These are functional alternatives because each produces the desired effect that payment is made. The *promptly* softgoal is a qualitative goal on *how* the functional effects are to be achieved. Because functional alternatives also address non-functional softgoals (as well as functional (hard) goals), the contribution that each functional alternative makes towards a non-functional goal is also represented as a means-ends link. These links, however, have additional attributes which indicate the *sense* (positive or negative) and *extent* of the contribution. Following [5], we use a notion of *satisficing* to distinguish between contributions that sufficiently or insufficiently address or fulfil a softgoal. These are marked as $\varDelta$ and ∇ respectively in the graphical notation.

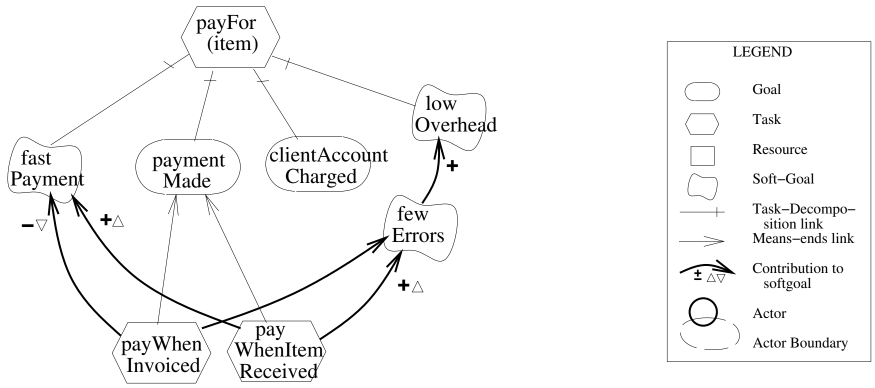

Fig. 4. Using softgoals to evaluate and guide generation of alternatives

The example in Figure 4 shows that pay-when-goods-received contributes positively to the softgoal of prompt payment, and is considered to adequately address the goal. Pay-when-invoiced contributes negatively to prompt payment, but not excessively so. Each softgoal node has a *satisficing status*. The status of a softgoal node can be computed by a labelling procedure from the statuses of descendent nodes in the network [5].

4 Using Strategic Actor-Relationship Modelling in Reengineering

Reengineering involves developing a good understanding of the current process, generation of new alternatives, and the evaluation of alternatives. The set of modelling concepts in the i^* framework facilitates these aspects of reengineering.

Understanding the current process. The Strategic Dependency model encourages a deeper understanding of a business process by focusing on intentional dependencies among actors, beyond the usual understanding based on the flow of physical or informational entities and the activities that process them. The SD model helps identify what is at stake, for whom, and what impacts are likely if a dependency fails. For example, who would care if an item is not received, or if an item is not paid for? By following the chain of dependencies, one can identify how actors are able to expand what they are able to accomplish by depending on others, and also the vulnerabilities that accompany the opportunities. For example, a client is able to have an item ordered, even if she did not have the knowhow or resources to do so (Figure 1). But in depending on a purchasing specialist, she also becomes vulnerable to the latter's failures. The SD model facilitates the identification of participants and stakeholders, and thus in determining the appropriate scope for a reengineering effort.

The Strategic Rationale model encourages a more specific understanding of the reasons behind why things are done in a certain way in an organization. The "whys" are revealed as decomposition and means-ends links that lead to outgoing dependencies are sought. Alternatively, starting from incoming dependencies, inquiry into the routines used by an actor would reflect the asking of "how" questions. The deeper understanding that is captured in the SD and SR models reflects the goal-seeking, free but socially-constrained, yet strategically-concerned character of organizational actors that is absent in conventional models of business processes.

Generation of new alternatives. The search for new and innovative alternatives to an existing business process is the central objective of business reengineering. The explicit representation of means-ends relationships in the SR model provides a systematic way for exploring the space of possible new process designs. Once an initial understanding of the existing process has been developed using the SD and SR models, other means to the identified ends can be systematically sought. Generic knowledge in the form of rules can be used to suggest new possibilities. For example, when an expert system capable of doing ordering of simple items becomes available, this knowledge can be coded as a rule. When searching for new ways to have items ordered, this would be identified as an alternative.

New alternatives often challenge hidden assumptions in existing process. For example, in searching for ways to make payment, pay-when-goods-received might be identified as an alternative to the customary pay-when-invoiced-received. The new rule challenges the assumption that invoices are necessary in the goods acquisition process [36].

The reengineering literature tends to emphasize the benefits of radically new ways of doing work. However, when new alternatives are proposed, one must also consider their implications on many other factors. The SR model facilitates the identification of cross-impacts with other issues by the use of multiple means-ends links to softgoals. Means-ends rules can be used in reverse (given means, identify the ends) to find out

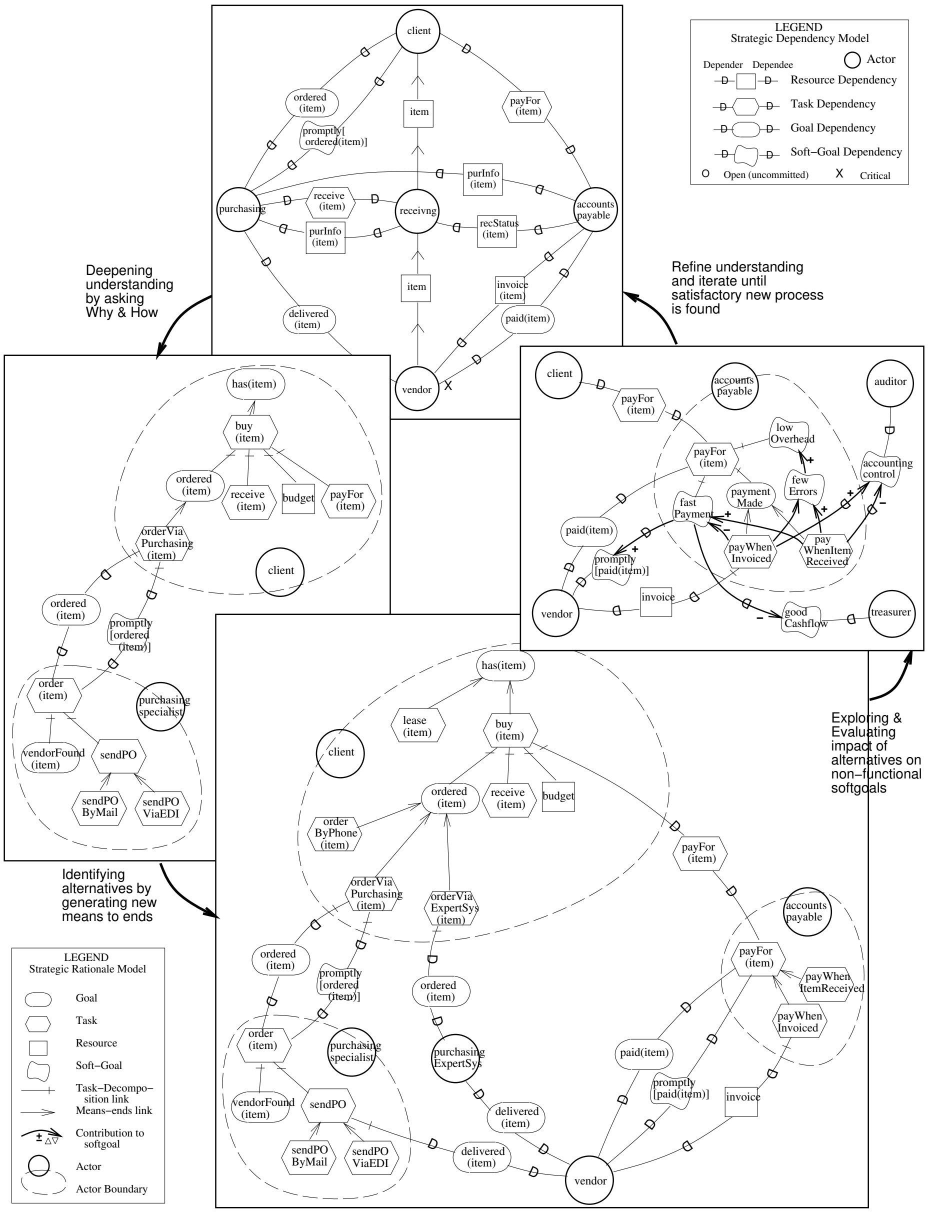

Fig. 5. An illustration of some of the features of i^* for supporting reengineering

what other goals are affected when adopting a new alternative. Such links may be traced to other affected actors (stakeholders) through the SD model. For example, while pay-when-goods-received eliminates invoices and therefore significantly reduces error, it is not so good for accounting control, which is a concern of the auditor. It also affects cash flow negatively, which is a concern of the corporate treasury.

Evaluation of alternatives. The framework also supports the evaluation of alternatives. The concept of workability provide a first-cut assessment of proposed routines. The evaluation of the satisficing statuses of softgoals provide a finer-grained, qualitative assessment. An interactive process of exploration and judgement is assumed. Figure 5 illustrates a reengineering scenario using the i^* framework.

5 Formal Representation

Formal representation of the modelling concepts enables computer-based tools to be developed to support the modelling and reasoning. Techniques for means-ends reasoning have been well-developed in the field of artificial intelligence (e.g.,[25]). External characterization of intentional agents using concepts such as beliefs, goals, ability, and commitment have also been developed (e.g.,[6, 32]). In the usage context of our framework, the objective of formal representation is not to have computer-generated reengineering solutions, but to use means-ends rules to suggest potential solutions, to check constraints, to maintain a network of rationales and assumptions, and also to benefit from knowledge structuring facilities that conceptual modelling schemes provide. Techniques from AI need to be adapted to suit our objective of modelling human social organizations, rather than the creation of artificial, computational agents. In this section, we present some highlights in the formal characterization of the SR and SD models. Further details are given in [38].

The characterization of the SR model is intended to capture the following intuitions. During process design, one of the basic activities is to look for routines that are *workable*. Routines are obtained by recursively reducing goals (or other intentional elements) using means-ends rules and task decomposition. For an open element to be workable, and actor x either knows how to do it (primitively workable), or knows someone who can do it. For a committed element to be workable, either x knows how to do it, or has commitment from someone who can do it, or else x must *further reduce* it through a routine until it is workable. We simplify the presentation by using a generic intentional element η. The variations for goal, task, resource, or softgoal as intentional elements are given in [38].

We say that a task t is workable if all its components (predicate el, for element) are workable, and all of its constraints (predicate cx) are believed to hold.

- Wt: $W(x,t) \subset (\forall \eta(el(\eta,t) \supset W(x,\eta)) \wedge \forall \kappa(cx(\kappa,t) \supset B(x,\kappa)))$

The criteria for an element being workable depends on whether it is an open element or a committed element of the task. An open element η (satisfying predicate oel) is workable if η is an open dependency ($\overrightarrow{D}$), or if it is workable under the (stronger) criteria of a committed element. A committed element η (predicate cel) is workable if η is primitively workable (predicate $\mathcal{E}_x$), or if there is some workable means-ends link (predicate me) linking it to a workable routine, or if η is an outgoing dependency and there is another agent y committed to producing η for x ($\overleftarrow{CD}$). ($\mathcal{U}_x$ is x's repertoire of routines.)

- We: $W(x,\eta) \subset (oel(\eta) \wedge W_o(x,\eta)) \vee (cel(\eta) \wedge W_c(x,\eta))$
- Weo: $W_o(x,\eta) \subset \overrightarrow{D}\ (x,\eta) \vee W_c(x,\eta)$

- Wec: $W_c(x,\eta) \subset \mathcal{E}_x(\eta) \vee \exists l \exists u(me(l,\eta,u) \wedge W(x,l) \wedge \mathcal{U}_x(u) \wedge W(x,u))$
$\vee(\overrightarrow{D}(x,\eta) \wedge \exists y B(x, C\overleftarrow{D}(y,x,\eta)))$

A routine u is workable if all of the elements specified in its `how` attribute (i.e., the "means" part) is workable and all of its subroutines are workable.

- Wu: $W(x,u) \subset \mathcal{U}_x(u) \wedge \forall \eta'(how(u,\eta') \supset W(x,\eta'))$
$\wedge \forall u'(subroutine(u',u) \supset W(x,u'))$

A means-ends link l with u as the means and η as the end is workable if the agent has a rule for that means-ends relationship and the agent believes the applicability condition α of that rule to hold. ($\mathcal{H}_x$ is the actor's repertoire of rules.)

- Wl: $W(x,l) \subset \exists \eta \exists u(\mathcal{U}_x(u) \wedge me(l,\eta,u) \supset \exists \alpha(\mathcal{H}_x(\eta,u,\alpha) \wedge B(x,\alpha)))$

In the Strategic Dependency model (SD), the external actor relationships are characterized in terms of more basic intentional concepts, namely, belief, goal, ability, and commitment. We use a right-pointing arrow to denote outgoing dependency (actor is depender) and the left-pointing arrow for incoming dependency (actor is dependee).

The opportunity aspect of a dependency is characterized as: actor x has open dependency if it believes that there exists some actor y who offers to achieve η, and that if y commits to it, then η will be workable for x.

- Dr: $\overrightarrow{D}(x,\eta) \supset B(x, \exists y(\overleftarrow{D}(y,\eta) \wedge (C(y,x,\eta) \supset W(x,\eta))))$

The offer of a dependency implies that the dependee is *able* to achieve η.

- De: $\overleftarrow{D}(y,\eta) \supset A(y,\eta)$

We take ability to mean that the actor has a routine for achieving η.

- Ae: $A(y,\eta) \equiv \exists u(\mathcal{U}_x(u) \wedge purpose(u,\eta))$

In order for a dependency to work, we need two assumptions. We need that the dependee y not only be able to produce η (have a routine), but that η is workable for y (i.e., the routine be workable). This is the role of commitment. Commitment bridges the gap between ability and workability. We call this the *Workability Commitment Assumption* – depender x believes that if dependee y is able to achieve η and it commits to some depender to achieve η, then η is workable for y.

- WCA: $B(x, A(y,\eta) \wedge \exists z C(y,z,\eta) \supset W(y,\eta)\ )$

We also need what we call the *Workability Transfer Assumption*, which says that if η is workable for y and y commits to produce η for x, then η becomes workable for x.

- WTA: $B(x, C(y,x,\eta) \wedge W(y,\eta) \supset W(x,\eta)\ )$

These assumptions are asserted in the belief context of x the depender. Beliefs are part of the rationale network in the SR model, and are supported by evidence via rationale links [38].

In the above, we have only addressed the opportunity aspect of a dependency. The vulnerability aspect of dependency relationships is also characterized axiomatically. The axioms for open, committed, and critical dependency (on the depender side) are based on how badly the depender is affected if the dependum is not delivered. The details are given in [38]. Reference [5] provides a formal scheme for generating a network of softgoal nodes and links and for evaluating softgoal statuses.

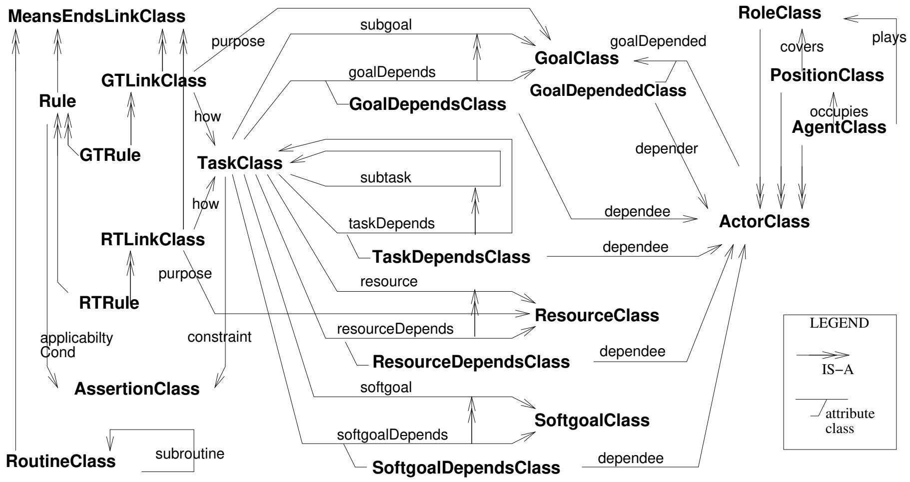

Fig. 6. A partial schema, showing task decomposition links and some classes of dependency links

The i^* framework is embedded in the conceptual modelling language Telos [22]. In doing so, we obtain an object-oriented representational framework, with classification, generalization, aggregation, attribution, and time. The extensibility of Telos, due to its metaclass hierarchy and treatment of attributes as full-fledged objects, facilitates the embedding of new modelling features.

Figure 6 presents a partial schema for the SR model. (A schema for the SD model has been presented in an earlier paper [37].) The middle section of the figure deals with task decomposition links and their corresponding dependency links. A task can be decomposed into subgoals, subtasks, resources and softgoals. Each of these have a dependency link counterpart. The dependee actor is attached as an attribute of the link from the task to its dependum. This permits a dependum to have multiple dependees.

An incoming dependency is a link from an actor to a dependum, with the depender represented as an attribute on the link. Figure 6 only shows the one for goal class (attribute `goalDepended`), the others are similar.

The left-hand section of Figure 6 shows relationships among means-ends links, rules and routines. Means-ends links have a purpose and a how. Each type of means-ends link is a specialization of this, with different types of intentional element as purpose and how. For brevity, the figure only shows the Goal-Task and Resource-Task types of means-ends links. A rule is a specialization of a means-ends link, with the added attribute of applicability condition. A routine is a specialization of a means-ends link, with subroutine as an additional attribute.

For modelling complex organizational relationships, actors are specialized into agents, roles, and positions (top right corner of Figure 6). Agents occupy positions; a position covers a number of roles; roles are played by agents.

The following is a sample of how the dependencies of an actor may be coded in Telos. The syntax is slightly simplified for presentation.

```
Class PurchasingSpecialist IN ActorClass
   WITH
  goalDepended, commits
    ord: ItemBeOrdered(i:Item)
          WITH dependee
                 cl: Client
          END
  softgoalDepended, commits
    ordp: ItemBeOrderedPromptly(i:Item)
          WITH dependee
                 cl: Client
          END
  goalDepends, committed
    del: ItemBeDelivered(i:Item)
          WITH dependee
                 vdr: Vendor
          END
  taskDepends, committed
    rcv: ReceiveItem
          WITH dependee
                 rcvg: Receiving
          END
  resourceDepended, commits
    pi: PurchasingInfo
          WITH dependee
                 rcvg: Receiving
                 ap: AccountsPayable
          END
 END
```

An example of rule representation is as follows.

```
Class CanOrderByExpertSystem IN Rule
 WITH
  purpose
    ord: ItemBeOrdered(i:Item)
  how
    es: OrderViaPurchasingExpertSystem
  applicabilityCondition
    expertSystemCanHandle:  $ SimplePurchase(ord) and LowQuantity(ord) $
 END

Class OrderViaPurchasingExpertSystem IN TaskClass
 WITH
  goalDep
    esord: ItemBeOrdered(i:Item)
      WITH dependee
             pes: PurchasingExpertSystem
      END
 END
```

```
Class OrderByPhone IN TaskClass
 WITH
  subgoal
    vf: VendorFound(vdr:Vendor,i:Item)
  subtask
    ppo: PlacePhoneOrder
  resource
    yp: YellowPages
 END
```

6 Discussion

The modelling of actor relationships extends traditional conceptual modelling techniques, such as E-R modelling, by explicitly dealing with the intentional and strategic dimension inherent in most business and organizational domains. The conceptual modelling approach to software engineering and information system development emphasizes the need to represent and utilize pertinent knowledge to support each phase of development and on-going evolution [23]. The i^* framework aims to add to this line of research ([12, 22, 16, 24, 5, 26]) by elaborating on the link between business redesign and technical system development. The eventual aim is to have a comprehensive development support environment spanning from business and organization design to software implementation.

Tools can be developed to manage the potentially large body of knowledge involved in a reengineering effort, and to supporting reasoning with them. Libraries of knowledge containing case experiences and general principles can be collected, organized, and applied to new cases with computer support. This knowledge would also facilitate on-going software and business process evolution.

The i^* framework draws on concepts of social organization from organization theory (e.g., [21, 30]), adapts formal agent modelling techniques from AI (e.g., [6, 32], which in turn draws on work in logic dealing with intentional concepts), and builds on existing conceptual modelling frameworks ([12, 22]).

Although a number of basic concepts of the framework are derived from AI concepts, the framework differs from AI frameworks in several significant ways. While AI aims to create computer programs or agents (e.g., robots) which are capable of solving problems on their own, our focus is on modelling and designing the network of relationships among social actors. Instead of focusing on the "operational" aspect of agents (e.g., proving the consequences of planned actions), we take a strategic view of actors' relationships and their reasoning about these relationships. The strategic view is less concerned about detail, and more concerned about broader issues such as opportunities and vulnerabilities, and the implications that each alternative process design might have for various stakeholders and participants. We allow a high degree of incompleteness in the modelling of tasks and routines. We do not require goals to be fully reduced to primitive actions, and rely instead on a notion of workability. We adopt a qualitative reasoning approach to allow many types of issues and concerns to be dealt with (as softgoals) within a single framework.

The framework is intended to provide interactive support for actors' reasoning in their design efforts, not to fully automate the design process. This framework may be seen as a specialization of design rationales and decision support frameworks (e.g., [28, 19, 5]) by providing an ontology for dealing more specifically with organization modelling and design. The qualitative reasoning scheme of [5] for dealing with non-functional requirements in software development is adapted and applied to organization modelling, complementing the functional components of the framework (goals, tasks, and resources). The three types of functional intentional elements are derived from the three basic ontological categories supported by the requirements modelling language RML [12] – assertion, activity, and entity.

The enterprise modelling framework of [3] also makes extensive use of conceptual modelling techniques to model business goals and rationales behind information system development. The need to understand "why," and to deal with fuzzy, informal, and non-functional issues are emphasized. The "process handbook" project [20] also uses conceptual modelling to organize knowledge for reengineering. Our framework emphasizes the multi-agent, organizational dimension of business (and private) objectives, and provides a model of organizational structure based on intentional dependencies. (The name i^* refers to the "distributed intentionality" perspective offered by the framework.)

In the requirements engineering area, the goal-oriented, "composite system design" approach makes use of means-ends reasoning to derive requirements from overall system goals (e.g., [8, 11]), where the overall system includes humans and computer systems. Global goals are decomposed and reduced primarily in a top-down fashion, until they can be assigned to agents. Our framework emphasizes a distributed, modelling approach. We assume that requirements engineering often has to deal with organizations with existing work patterns and therefore desires and interests that are (already) distributed, rather than centrally or globally specified. Distributed desires and interests need to be *modelled* (through an inquiry process) in the form of an intentional structure (the SD model). The means-ends reasoning also needs to be distributed (the SR model). Our framework is therefore more readily applied to the business process *re*-engineering context, which presupposes existing processes and associated organizational constraints. However, multi-perspective approaches are also being developed in requirements engineering (e.g., [27]).

In our earlier papers on reengineering, we have illustrated how the intentional nature of the SD model is able to highlight important differences between business processes that are hard to express in conventional, non-intentional workflow models [35]. Reference [36] presented an early, informal version of the SR model, which we have developed more fully and formally in this paper.

The modelling of software processes also has commonalities with the modelling of business processes. One framework that uses conceptual modelling to advantage is [17]. However, a majority of software process models tend to be intended for process execution or enaction in some process-centred software engineering environment (the technology infrastructure). We have proposed the i^* framework for software process modelling to emphasize the need to understand and support the design or redesign of software processes and their embedding organization [37].

In our earlier papers, the SD model was called the Actor Dependency model. The

SR model was separated into a Functional Rationales model and a Non-Functional Rationales model.

7 Conclusion

E-R modelling has made important contributions to the conceptual modelling field since its introduction almost two decades ago [4]. It has proved to be a useful technique despite its simple ontology. As information system applications become more sophisticated, conceptual modelling needs to deal with richer domains with specialized ontologies [13]. One example is the need to deal with the modelling of organizations, as in business process reengineering.

Reengineering presents considerable challenges. On the one hand, there is promise of potentially dramatic benefits in organizational effectiveness. On the other hand, it involves substantial efforts and risks. Many factors can influence outcome. During the process modelling and re-design phase, one would like to have a careful and thorough examination of all relevant issues, take into account input from all stakeholders, jointly explore alternatives, and anticipate problems as much as possible.

Conceptual modelling techniques can be invaluable in providing clear representation of the key domain concepts and issues during a reengineering effort. Knowledge about the domain can be organized using structuring mechanisms such as classification, generalization, aggregation, and time. However, a richer ontology beyond traditional entity/relationship/attribute concepts would be helpful for addressing the specific needs of process modelling and reasoning.

We have proposed one approach which emphasizes that organizations are made up of strategic, intentional actors. The Strategic Dependency model allows the modelling of how strategic actors relate to each other intentionally, while the Strategic Rationale model allows modelling of the means-ends reasoning the actors have about different potential ways of relating to each other for accomplishing work.

Our work is still at an exploratory stage. The modelling framework has been applied to examples from the literature in several areas, but has yet to be tested in actual use. Tools to support the framework remain to be implemented, although some of the underlying components already exist (Telos [22], NFR Assistant [5, 26]). Further conceptual development would include exploration of other types of actor relationships.

As information systems [7] as well as human organizations [29] progress increasingly toward cooperative and distributed, networked configurations, it is becoming ever more important to have models that can help reason how complex, interlinked systems contribute to business and organizational objectives. The "distributed intentionality" perspective taken by the i^* framework offers one approach for modelling and reasoning about the complex interactions among information system components and humans in distributed, evolving business processes.

References

1. A. Borgida, S. Greenspan, J. Mylopoulos, Knowledge Representation as the Basis for Requirements Specifications, *IEEE Computer*, April 1985, pp. 82-91.

2. J. A. Bubenko, Information Modeling in the Context of System Development, *Proc. IFIP,* 1980, pp. 395-411.
3. J. A. Bubenko, Extending the Scope of Information Modeling, *Proc. 4th Int. Workshop on the Deductive Approach to Information Systems and Databases,* Lloret-Costa Brava, Catalonia, Sept. 20-22, 1993, pp. 73-98.
4. P. P. Chen, The Entity-Relationship Model – Toward a Unified View of Data, *ACM Trans. Database Sys.*, vol. 1, no. 1, 1976, pp. 9-38.
5. K. L. Chung, *Representing and Using Non-Functional Requirements for Information System Development: A Process-Oriented Approach*, Ph.D. Thesis, Dept. of Comp. Sci., Univ. of Toronto, 1993.
6. P. R. Cohen and H. J. Levesque, Intention is Choice with Commitment, *Artif. Intell.,* 42 (3), 1990.
7. Second International Conference on Cooperative Information Systems (CoopIS-94), Toronto, Canada, May 17-20, 1994.
8. A. Dardenne, A. van Lamsweerde and S. Fickas, Goal-Directed Requirements Acquisition, *Science of Computer Programming*, 20, 1993, pp. 3-50.
9. T. H. Davenport, *Process Innovation: Reengineering Work Through Information Technology*, Harvard Business School Press, 1993.
10. E. Dubois, J. Hagelstein, E. Lahou, F. Ponsaert and A. Rifaut, A Knowledge Representation Language for Requirements Engineering, *Proc. IEEE*, 74 (10), Oct. 1986, pp. 1431–1444.
11. M. S. Feather, Language Support for the Specification and Development of Composite Systems, *ACM Trans. Prog. Lang. and Sys.* 9(2), April 1987, pp. 198-234.
12. S. J. Greenspan, *Requirements Modelling: A Knowledge Representation Approach to Software Requirements Definition,* Ph. D. Thesis, Dept. of Comp. Sci., Univ. of Toronto, 1984.
13. S. J. Greenspan, J. Mylopoulos, A. Borgida, On Formal Requirements Modeling Languages: RML Revisited, (invited plenary talk), *Proc. 16th Int. Conf. Software Engineering*, May 16-21 1994, Sorrento, Italy, pp. 135-147.
14. M. Hammer, Reengineering Work: Don't Automate, Obliterate, *Harvard Business Review*, July-August 1990, pp. 104-112.
15. M. Hammer and J. Champy, *Reengineering the Corporation: A Manifesto for Business Revolution*, HarperBusiness, 1993.
16. M. Jarke, J. Mylopoulos, J. W. Schmidt, Y. Vassiliou, DAIDA: An Environment for Evolving Information Systems, *ACM Trans. Information Systems*, 10(1), Jan. 1992, pp. 1-50.
17. M. Jarke, T. Rose, Specification Management with CAD°, *Conceptual Modelling, Databases, and CASE*, P. Loucopoulos, R. Zicari, eds., Wiley, 1992, pp. 489-505.
18. P. Keen, *Shaping the Future: Business Design Through Information Technology*, Harvard Business School Press, 1991.
19. J. Lee, *A Decision Rationale Management System: Capturing, Reusing, and Managing the Reasons for Decisions*, Ph.D. thesis, MIT, 1992.
20. T. W. Malone, K. Crowston, J. Lee, B. Pentland, Tools for Inventing Organizations: Toward a Handbook of Organizational Processes, *Proc. 2nd Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*, IEEE Computer Society Press, 1993, pp. 72-82.
21. J. G. March and H. A. Simon, *Organizations*, Wiley, 1958.
22. J. Mylopoulos, A. Borgida, M. Jarke, M. Koubarakis, Telos: Representing Knowledge about Information Systems, *ACM Trans. Info. Sys.*, 8 (4), 1991.
23. J. Mylopoulos, Representing Knowledge About Information Systems, *Int. Workshop on Development of Intelligent Information Systems*, Niagara-on-the-Lake, Ontario, Canada, April 21-23, 1991, pp. 94-96.
24. J. Mylopoulos, L. Chung, B. Nixon, Representing and Using Non-Functional Requirements: A Process-Oriented Approach, *IEEE Trans. Soft. Eng.*, 18 (6), June 1992.

25. N. Nilsson, *Principles of Artificial Intelligence*, Tioga Press, 1980.
26. B. A. Nixon, Representing and Using Performance Requirements During the Development of Information Systems, *Proc. 4th Int. Conf. Extending Database Technology (EDBT 94),* Cambridge, U.K., M. Jarke et al, eds., Springer-Verlag, 1994, pp. 187–200.
27. B. Nuseibeh, J. Kramer, A. Finkelstein, Expressing the Relationships Between Multiple Views in Requirements Specification, *15th Int. Conf. Soft. Eng.*, Baltimore, 1993, pp.187-196.
28. C. Potts and G. Bruns, Recording the Reasons for Design Decisions, *Proc. Int. Conf. Software Engineering*, 1988, pp. 418-427.
29. J. F. Rockart and J. E. Short, The Networked Organization and the Management of Interdependence, *The Corporation of the 1990's – Information Technology and Organizational Transformation*, M. Scott Morton, ed., 1991.
30. W. R. Scott, *Organizations: Rational, Natural, and Open Systems*, 2nd ed., Prentice Hall, 1987.
31. C. Theodoulidis, B, Wangler, P. Loucopoulos, The Entity-Relationship-Time Model, *Conceptual Modelling, Databases, and CASE*, P. Loucopoulos, R. Zicari, eds., Wiley, 1992, pp. 81-116.
32. B. Thomas, Y. Shoham, A. Schwartz, and S. Kraus, Preliminary Thoughts on an Agent Description Language, *Int. J. Intell. Sys.*, Vol. 6, 1991, pp. 498-508.
33. E. Yu, Modelling Organizations for Information Systems Requirements Engineering, *Proc. 1st IEEE Int. Symp. Requirements Engineering*, San Diego, Calif., January 1993, pp. 34-41.
34. E. Yu, An Organization Modelling Framework for Multi-Perspective Information System Design, *Requirements Engineering 1993 – Selected Papers*, J. Mylopoulos et al., eds., Tech. Rpt. DKBS-TR-93-2, Dept. Comp. Sci., Univ. of Toronto, July 1993, pp. 66-86.
35. E. Yu, J. Mylopoulos, An Actor Dependency Model of Organizational Work – With Application to Business Process Reengineering, *Proc. Conf. Organizational Computing Systems (COOCS 93)*, Milpitas, Calif., Nov. 1-4, 1993, pp. 258-268.
36. E. Yu, J. Mylopoulos, Using Goals, Rules, and Methods To Support Reasoning in Business Process Reengineering, *Proc. 27th Hawaii Int. Conf. System Sciences*, Maui, Hawaii, Jan. 4-7, 1994, vol. IV, pp. 234-243.
37. E. Yu, J. Mylopoulos, Understanding "Why" in Software Process Modelling, Analysis, and Design, *Proc. 16th Int. Conf. Software Engineering*, May 16-21 1994, Sorrento, Italy, pp. 159-168.
38. E. Yu, *A Framework for Process Modelling and Reengineering*, Ph.D. Thesis, Dept. of Computer Science, Univ. of Toronto, forthcoming.

This article was processed using the LaTeX macro package with LLNCS style