Business Process Modeling and Design: AI Models and Methodology*

Manolis Koubarakis
Dept. of Informatics
University of Athens
Panepistimioupolis, TYPA Buildings
157 81 Athens, Greece
manolis@di.uoa.gr

Dimitris Plexousakis †
Dept. of Computer Science
University of Crete
Heraklion, Crete
713 05 Greece
dp@csd.uch.gr

Abstract

We present a formal framework for representing enterprise knowledge. The framework is largely based on ideas from AI and its concepts (objectives and goals, roles and actors, actions and processes, responsibilities and constraints) allow business analysts to capture knowledge about an enterprise in an intuitive and formal way. We also present a methodology which allows business analysts to go from high-level enterprise objectives, to detailed and formal specifications of business processes that can be enacted to realise these objectives. The specifications can be formally verified for correctness.

1 Introduction

The problem of *representing, analysing* and *managing* knowledge about an organisation and its processes has always been very important. Recently, management and computer science researchers have debated the use of information technology for tackling this complex problem [Hammer and Champy, 1993; Davenport, 1993; WFMC, ; NIST-PSL, ; IDEF, ; Ould, 1994; Georgakopoulos *et al.*, 1995; Leymann and Altenhuber, 1994; Jennings *et al.*, 1996; Yu *et al.*, 1996; Kirn and O'Hare, 1997]. Ultimately this research community is interested in improving the understanding of organisations and their processes, facilitating process design and analysis and supporting process management (i.e., process enactment, execution and monitoring).

The role of Artificial Intelligence in Business Process Design, Analysis and Reengineering is two-fold: AI can provide both the enabling technology for representing and automatically reengineering processes, and tools to support process redesign [Hamscher, 1994]. The majority of attempts to put AI to work in this respect have insisted on the first of these roles. It remains a challenging issue to develop tools for evaluating and for assisting the production of designs.

In this paper we firstly present a formalism that can be used to represent knowledge about organisations and their business processes. Our work is motivated by successful enterprise modeling projects particularly F^3 [Bubenko, 1994; Loucopoulos and Kavakli, 1995] and EKD [Kavakli and Loucopoulos, 1998; Bubenko *et al.*, 1998]. However, unlike these projects, our framework emphasises formality and advocates the use of AI techniques for representing knowledge about organisations and their processes. More precisely we propose to use situation calculus [McCarthy and Hayes, 1969; Reiter, 1991] and the concurrent logic programming language ConGolog [De Giacomo *et al.*, 1997] for representing knowledge about organisations and their processes. In this respect we continue the work of one of us [Plexousakis, 1995; 1996] (but also [Yu *et al.*, 1996; Lesperance *et al.*, 1999a; Fox and Gruninger, 1998]) who suggested to use these formal tools to model organisations.

This paper also presents a methodology which enables business analysts to go from high-level enterprise objectives, to detailed and formal specifications of business processes for realising these objectives. The methodology can be used by an enterprise that wishes to develop a new business process, or alternatively model, document and analyse formally an existing process. Our methodology utilises previous results developed in [Plexousakis, 1995].

A few words about our representational framework are in order here. We will represent enterprise knowledge using an extension of the formalism of *situation calculus* [McCarthy and Hayes,

*The work of the first author was partially supported by BT Labs, Ipswich, U.K. through a Short Term Research Fellowship. Most of this work was carried out while the author was a lecturer in the Dept. of Computation, UMIST, U.K. The work of the second author was partially supported by the Univ. of South Florida through a Research and Creative Scholarship Grant.

†On leave from the Dept. of Computer Science and Engineering, University of South Florida, Tampa, FL.

1969; Reiter, 1991]. This formalism has been designed especially for knowledge representation and reasoning in dynamically evolving domains. Technically, our basic tool will be a *many-sorted first-order language* $\mathcal{L}$ which is defined in the following way. The logical symbols of $\mathcal{L}$ include parentheses, a countably infinite set of variables, the equality symbol $=$ and the standard sentential connectives. The remaining machinery of $\mathcal{L}$ (sort, predicate and function symbols) will be defined in Sections 2, 3, 4 and 5 where intuitive modeling concepts will need to be formalised.

The rest of this paper is structured as follows. Sections 2, 3, 4 and 5 present our enterprise model. Section 6 presents our methodology. Section 7 discusses related work and presents our conclusions.

2 The Organisational Submodel

Our approach to enterprise modeling follows the lead of F^3 [Bubenko, 1994; Loucopoulos and Kavakli, 1995] and EKD [Kavakli and Loucopoulos, 1998; Bubenko *et al.*, 1998]. We will present an *enterprise model* which consists of five interconnected submodels (organisational submodel, objectives and goals submodel, process submodel, concepts submodel and constraints submodel) that can be used to describe *formally* different aspects of an organisation.

In this section we initiate the presentation of the five submodels making up our enterprise modelling framework. The first submodel is the organisational submodel with main concepts *actor* and *role*. An *actor* is a person or a software/hardware system in the context of the organisation we are modelling (e.g., an employee, a customer, a printer etc.). Actors are distinguished into *human* and *automated* ones. Actors are capable of executing certain activities, but they might not be capable of executing others.

An *organisational role* involves a set of *responsibilities* and *actions* carried out by an actor or a group of actors within an organisation [Ould, 1995; Yu, 1994; Dobson *et al.*, 1994; Kinny *et al.*, 1996]. Organisational roles can take many forms [Ould, 1995]: a unique functional group (e.g., Systems Department), a unique functional position (e.g., Managing Director), a rank or job title (e.g., Lecturer Grade A), a replicated functional group (e.g., Department), a replicated functional position (e.g., Director), a class of persons (e.g., Customer) or an abstraction (e.g., Progress Chasing).

Role instances are acted out by actors.[1] Different actors can play different roles at different moments of time (e.g., today the Managing Director can be John Smith, tomorrow it can be Tony Bates). Many instances of the same role can be active at any moment in time.

The concepts introduced above can be defined formally by introducing appropriate constructs of $\mathcal{L}$ and writing axioms that capture their semantics. We introduce unary predicates $Actor, HumanActor$, $AutomatedActor$ and $Role$, and binary predicate $PlaysRole$ with obvious meaning. The following axiom gives the relation between actors, human actors and automated actors:[2]

$$(\forall x)(Actor(x) \equiv HumanActor(x) \lor AutomatedActor(x))$$

Example 2.1 Throughout this paper we will demonstrate the features of our proposal by considering an imaginary Computer Science department DEPT as the organisation considered by our study. We assume that this department has so far no postgraduate program, and it is now considering the development of processes for the admission and education of postgraduate students. Using the predicates introduced above, the following sentences of $\mathcal{L}$ can be introduced in the organisational submodel for DEPT:

$$HumanActor(John),\ HumanActor(Mary)$$
$$Role(Tutor),\ Role(Secretary)$$
$$PlaysRole(John, Tutor)$$
$$PlaysRole(Mary, Secretary)$$

3 The Objectives and Goals Submodel

An *enterprise goal* is a desired state of affairs [Fikes and Nilsson, 1971; Dardenne *et al.*, 1993; Yu, 1994; Loucopoulos and Kavakli, 1995; Lee, 1994; Ould, 1994; Yu *et al.*, 1996; Kavakli and Loucopoulos, 1998]. Examples of enterprise goals are the following: "all customers enquiries are answered within one day", "profits are maximised" and so on.

In our framework goals are associated with the following components of other submodels:

- *Roles and actors* (organisational submodel). Goals are assigned to roles as a matter of policy by the organisation. Organisational goals become responsibilities of roles and the actors playing these roles.
- *Processes* (process submodel). The *purpose* of a process is the achievement of one or more goals. For example, the process of managing project X might have the purpose of achieving the goal "project X is completed successfully".

[1]Role instances can also be acted out by *groups of actors* (e.g., a committee or a meeting). Currently our framework is rather poor with respect to role playing by groups of actors e.g. we have no explicit notion of joint goals, joint processes etc. [Tidhar, 1993].

[2]Notation: We will use strings of characters starting with an uppercase letter to denote predicate, function and constant symbols of $\mathcal{L}$. Variable names start with a lowercase letter. We will not show sorts explicitly but they will be clear from the context.

- *Entities* (concepts submodel). Every goal refers to certain enterprise entities. For example, the goal "two C++ programmers should be hired by the Systems Department" refers to entities "Systems Department" and "C++ programmer".

Explicit capturing of enterprise goals is important because it allows us to study organisations and their processes from an *intentional* point of view [Yu and Mylopoulos, 1994b; Yu, 1994]. For example, this enables us to represent not only "what" information (e.g., what sub-processes form a process) as in standard process representations, but also "why" information (e.g., why a specific activity is done). When goals are combined with other intentional concepts like actors and roles, we are also enabled to represent "who" information (e.g., "who is responsible for bringing about a state of affairs").

3.1 Organising Goals

Organisational goals can be *reduced* into alternative combinations of subgoals [Dardenne *et al.*, 1993; Chung, 1993; Yu, 1994; Lee, 1994; Bubenko, 1994; Loucopoulos and Kavakli, 1995; Kavakli and Loucopoulos, 1998; Bubenko *et al.*, 1998] by using *AND/OR goal graphs* originally introduced in the area of problem solving [Fikes and Nilsson, 1971]. For example, the goal "our sales targets are achieved" can be AND-reduced to two goals "our sales targets for product A are achieved" and "our sales targets for product B are achieved".

We utilise the notion of goal reduction to define the concept of objective. An *organisational objective* is a goal that does not present itself through goal reduction. In other words, an objective is a top-level goal; it is an *end* desired in itself, not a *means* serving some higher level end [Loucopoulos and Karakostas, 1995].

Goals can *conflict* with each other [Dardenne *et al.*, 1993; Chung, 1993; Yu and Mylopoulos, 1994a; Loucopoulos and Kavakli, 1995; van Lamsweerde *et al.*, 1998]. In our framework goals $G_1, \ldots, G_n$ conflict if they cannot be *satisfied* simultaneously given our knowledge about the enterprise [van Lamsweerde *et al.*, 1998]. Goals can also *influence* positively or negatively other goals [Mylopoulos *et al.*, 1992; Chung, 1993; Yu and Mylopoulos, 1994a; Yu, 1994; Loucopoulos and Kavakli, 1995]. Such interactions between goals must be noted explicitly to facilitate goal-based reasoning (see Section 6).

3.2 Defining Goals Formally

Organisational goals can be described formally or informally. Organisational objectives and other high-level goals are usually difficult to formalise. These goals should be described only informally, and reduced step by step to more concrete and formal goals. Appropriate formal concepts and tools for assisting goal reduction (in the context of requirements modelling) are discussed in [Dardenne *et al.*, 1993].

Because a goal is a desired state of affairs many concrete and formal goals can be formalised as *sentences* of $\mathcal{L}$ as demonstrated by the following example.

Example 3.1 The operational goal "enquiries are answered as soon as they are received" can be formalised by the following sentence of $\mathcal{L}$:

$$(\forall a)(\forall e)(\forall x)(\forall s)(\forall s')\ (PlaysRole(a, Secretary) \wedge$$

$$Received(e, a, s) \wedge s' = Do(x, s)\ \supset\ Answered(a, e, s'))$$

where predicates *Received* and *Answered* have obvious meaning and $s' = Do(x, s)$ means that s' is the situation (i.e., state) resulting from the execution of action x in situation s. More details about the situation calculus and its machinery are given in Section 4. Note also that the use of a formal language forces one to be very precise and dispense with informal concepts such as "as soon as".

4 The Process Submodel

A good process model should allow representation of "*what* is going to be done, *who* is going to do it, *when* and *where* it will be done, *how* and *why* it will be done, and *who is dependent* on its being done" [Curtis *et al.*, 1992]. The process model presented in this section allows one to answer five of these seven questions. We do not include a spatial attribute for processes and we do not consider dependencies [Yu, 1994] explicitly.

The main concepts of the process submodel are: *action, process, role, actor* and *goal*. The process submodel is connected to the organisational submodel through the concepts of *actor* and *role*. All actions carried out as part of a process are executed in the context of an organisational role by an actor playing that role. In this respect we have been inspired by the Role-Activity diagrams of [Ould, 1995]. The process submodel is also closely related with the objectives and goals submodel: processes are operationalisations of organisational goals [Bubenko, 1994; Anton *et al.*, 1994].

4.1 Primitive and Complex Actions

Our process submodel is built around the concepts of situation calculus [McCarthy and Hayes, 1969; Reiter, 1991] and the concurrent logic programming language ConGolog [De Giacomo *et al.*, 1997]. The situation calculus is a first-order language for representing dynamically evolving domains. A *situation* is a state of affairs in the world we are modelling. Changes are brought to being in situations as results of actions performed by actors. Actions are distinguished into *primitive* and *complex*. Usually an action is considered to be primitive if no decomposition will reveal any further information which is of interest. To deal with these new concepts, we enrich our language $\mathcal{L}$ with a sort *Action* for actions and a sort *Situation* for situations. Actions are denoted by first-order terms of the form $act(args)$ where act is a function symbol and $args$ is a list of first-order terms (e.g., $SendOfferLetter(act, app)$). By convention the first argument in the list $args$ is always the actor executing the action.

For an action α and a situation s, the term $Do(\alpha, s)$ denotes the situation that results from the execution of action α in situation s. Relations whose truth values may differ from one situation to another are called *fluents*. They are denoted by predicate symbols having a situation term as their last argument. Similarly, the term *functional fluent* is used to denote functions whose denotation varies from one situation to another.

Primitive actions are introduced formally by expressions of the following form:

action α
  precondition ϕ_1
  effect ϕ_2
endAction

where α is an action, and ϕ_1, ϕ_2 are formulas of $\mathcal{L}$.

Example 4.1 The following expression defines the action of forwarding an application *app* by actor *act*1 to actor *act*2:

action $ForwardApp(act1, act2, app)$
  precondition $Has(act1, app)$
  effect $Has(act2, app) \wedge \neg Has(act1, app)$
endAction

Our framework permits the recursive definition of *complex actions* (simply actions from now on) by adopting the exact syntax and semantics of ConGolog [De Giacomo *et al.*, 1997]:

- *Primitive actions* are actions.
- The special action of *doing nothing* is an action and is denoted by **noOp**.
- *Sequencing.* If α_1, α_2 are actions, then $\alpha_1; \alpha_2$ is the action that consists of α_1 followed by α_2.
- *Waiting for a condition.* If ϕ is a formula of $\mathcal{L}$ then $\phi?$ is the action of waiting until condition ϕ becomes true.
- *Non-deterministic choice of actions.* If α_1, α_2 are actions, then $\alpha_1 | \alpha_2$ is the action consisting of non-deterministically choosing between α_1 and α_2.
- *Non-deterministic choice of action parameters.* If α_1, α_2 are actions, then $\Pi_x(\alpha_1)$ denotes the non-deterministic choice of parameter x for α_1.
- *Non-deterministic iteration.* If α is an action, then α^* denotes performing α sequentially zero or more times.
- *Conditionals and iteration.* If α_1, α_2 are actions, then **if** ϕ **then** α_1 **else** α_2 defines a conditional and **while** ϕ **do** α_1 defines iteration.
- *Concurrency.* If α_1, α_2 are actions, then $\alpha_1 \parallel \alpha_2$ is the action of executing α_1 and α_2 concurrently.
- *Concurrency with different priorities.* If α_1, α_2 are actions, then $\alpha_1 \gg \alpha_2$ denotes that α_1 has higher priority than α_2, and α_2 may only execute when α_1 is done or blocked.
- *Non-deterministic concurrent iteration.* If α is an action, then $\alpha^{\parallel}$ denotes performing α concurrently zero or more times.
- *Interrupts.* If $\vec{x}$ is a list of variables, ϕ is a formula of $\mathcal{L}$ and α is an action then $\langle \vec{x} : \phi \to \alpha \rangle$ is an interrupt. If the control arrives at an interrupt and the condition ϕ is true for some binding of the variables then the interrupt triggers and α is executed for this binding of the variables. Interrupts are very useful for writing *reactive* processes.
- *Procedures.* Procedures are introduced with the construct **proc** $\beta(\vec{x})$ **endProc**. A *call* to this procedure is denoted by $\beta(\vec{x})$.

Examples of complex actions are given in Figures 1, 2 and 3 (see Section 6). Other examples of ConGolog programs can be found in [De Giacomo *et al.*, 1997; Lesperance *et al.*, 1999a].

4.2 Categories of Actions

We distinguish actions into *causal* and *knowledge-producing*. Causal actions change the state of affairs in the enterprise we are modelling (e.g., the action of forwarding an application form). Knowledge-producing actions do not change the state of the enterprise but rather the mental state of the enterprise actors (e.g., a perceptual or a communicative action) [Scherl and Levesque, 1993; Lesperance *et al.*, 1999b]. It is known that knowledge-producing actions can be defined in the situation calculus formalism [Scherl and Levesque, 1993; Lesperance *et al.*, 1999b].

Finally, actions can be *exogenous*. This concept corresponds to the notion of external event in other process frameworks. Exogenous actions are necessary in an enterprise modeling framework since they allow us to "scope" our modeling and consider certain parts of the enterprise (or its environment) as being outside of the area we are modelling. Exogenous actions can also be handled by the situation calculus formalism [De Giacomo *et al.*, 1997].

4.3 Business Processes

A *business process* can now be informally defined as a network of actions performed in the context of one or more organisational roles in pursuit of some goal. Formally, a business process is defined by an expression of the following form:

process *id*
  purpose *goals*
  RoleDefs
endProcess

where *id* is a process identifier, *goals* is a list of goals (separated by commas) and *RoleDefs* is a sequence of statements defining roles and their local ConGolog procedures. The purpose statement in a process definition introduces the purpose of a process i.e., the organisational goals *achieved* by the process. The concept of purpose captures *why* a process is done [Curtis *et al.*, 1992].

Processes are distributed among organisational roles and ConGolog procedures are used to capture the details of a process. Roles and their procedures are defined by expressions of the following form:

role *id*
  responsibility *resps*
  ProcedureDefs
endRole

where *id* is a role identifier, *resps* is a list of goals (separated by commas) and *ProcedureDefs* is a set of ConGolog procedures. The responsibility statement declares that role *id* is responsible for achieving the goals in list *resps*. Examples of role definitions are given in Figures 1, 2 and 3 (see Section 6).

5 The Concepts and Constraints Submodels

The *concepts* submodel contains information about enterprise entities, their relationships and attributes. Information in this submodel is formally expressed by sentences of $\mathcal{L}$ using appropriate predicate and function symbols (e.g., for our DEPT enterprise a predicate $Has(act, app)$ might be used to denote that actor *act* has application *app*). Enterprise data are part of this submodel.

The *constraints* submodel is used to encode restrictions imposed on the enterprise. Constraints can be formally expressed by sentences of $\mathcal{L}$ using the machinery of the situation calculus and the symbols defined in the rest of the submodels. Constraints can be static (i.e., referring to a single situation) or dynamic (i.e., referring to more than one situation) [Plexousakis, 1996]. An example of a static constraint is given in Example 6.1 (Section 6.5).

This section finishes the presentation of our enterprise model. Let us now turn to our methodology.

6 A goal-oriented methodology for business process design

The objective of this section is to develop a methodology which can be used by an enterprise that wishes to develop a *new* business process. The methodology starts with the objectives of the enterprise concerning this new development and produces a detailed formal specification of a business process which achieves these objectives. The formal specification is developed as a set of submodels (based on the conepts discussed in previous sections) that capture the new process from various viewpoints.

The steps of the proposed methodology are the following:

- Identify the organisational objectives and goals. Initiate goal reduction.
- Identify roles and their responsibilities. Match goals with role responsibilities.
- For each role specify its primitive actions, the conditions to be noticed and its interaction with other roles.
- Develop ConGolog procedures local to each role for discharging each role's responsibilities.
- Verify formally that the ConGolog procedures local to each role are sufficient for discharging its responsibilities.

The steps of the methodology are ordered, but some of them will in practice need to run concurrently. Also, backtracking to a previous step will often be useful in practice. These issues are discussed in detail below. The final product of an application of the methodology is a complete enterprise model that can be used to *study* and *analyse* the proposed business process. The process specification can also serve as a guide for the development of an information system that implements the process.

Although this section will only concentrate on using the proposed methodology for the specification of a new business process, the methodology can also be used (with minimal changes) for modelling and documenting formally an existing process.

This paper will not cover certain other issues related to applying this methodology to an industrial application (i.e., who must participate in the modelling team, what techniques can be used for eliciting the required knowledge etc.). For these problems we can utilise the guidelines provided by similar frameworks e.g., EKD [Bubenko *et al.*, 1998].

6.1 Identifying Objectives and Goals

The first step of the proposed methodology is the elicitation of an *initial statement* of the enterprise objectives and goals concerning the new process. This will involve brainstorming sessions with the enterprise stakeholders, studying documents (e.g., mission statement) outlining the strategy of the enterprise to be modelled (and possibly other enterprises in the same industry sector), and so on

[Bubenko *et al.*, 1998]. During this activity the analyst using our methodology must try to uncover not only prescriptive goals, but also descriptive ones [Anton *et al.*, 1994].

After we have a preliminary statement in natural language of the enterprise objectives and goals, then the process of constructing a corresponding AND/OR goal graph by asking "why" and "how" questions can begin [Dardenne *et al.*, 1993]. This process involves reducing goals, identifying conflicts and detecting positive and negative interactions between goals. The process of goal reduction will lead to a better understanding of the organisational goals, and very often to a reformulation of their informal definition.

An important issue that needs to be addressed at this stage is the distinction between *achievable* and *unachievable* (or *ideal*) goals. Ideal goals need to be considered, but in the process of AND/OR-reduction they need to be substituted by weaker goals that are actually achievable [van Lamsweerde *et al.*, 1995].

After the AND/OR graph corresponding to informal goals is sufficiently developed and stable, the process of *goal formalisation* can start.

This is also a good point to start in parallel the activity of *entity identification* and *formalisation*. This involves specifying enterprise entities, their relationships, their attributes and their constraints; formalising them; and populating the corresponding concepts and constraints submodels. A side-product of this activity is a vocabulary sufficient for goal formalisation. The activity of entity identification and formalisation will go on throughout subsequent steps of the methodology as the business process will be analysed in more detail and new entities will come into the picture.

This step of our methodology is identical with goal reduction steps in goal-oriented requirements modeling frameworks [Yu and Mylopoulos, 1994a; Mylopoulos *et al.*, 1992; Dardenne *et al.*, 1993; van Lamsweerde *et al.*, 1995] and related goal-oriented enterprise modeling frameworks [Bubenko, 1994; Loucopoulos and Kavakli, 1995; Kavakli and Loucopoulos, 1998; Bubenko *et al.*, 1998].

Example

Let us consider our imaginary Computer Science department DEPT and its objectives regarding the new postgraduate program. Let us assume that the departmental paper addressing this new development has been studied and the following high-level goals have been discovered:

- G_1 : The number of admitted postgraduate students is maximised.
- G_2 : The amount of money spent by the postgraduate program office for advertising the program is kept under 1,000.
- G_3 : Academic standards are upheld during the admission process.

Let us first consider goal G_1 and ways to decompose it into finer goals. G_1 can be AND-decomposed into the following subgoals:

- G_{11}: The graduate program is advertised aggressively.
- G_{12}: Enquiries are answered as soon as they are received.
- G_{13}: Evaluation of applications and notification of applicants is completed very quickly.

Goal G_{11} can be AND-decomposed further as follows:

- G_{111}: Universities where members of staff have established academic connections are targeted.
- G_{112}: A presentation of the program is included in the departmental WWW page.
- G_{113}: Advertisements appear in appropriate media.

This is a good point to consider the *duality* between the two semantically different concepts of goals and non-primitive actions and the resulting issue of what natural language constructs to use for expressing them. As in [Lee, 1994] we have chosen to use passive voice (e.g., "is maximised", "is advertised" etc.) for expressing goals.

Goal G_{111} can now be OR-decomposed as follows:

- G_{1111}: An information package is sent to the undergraduate office and each member of academic staff in the targeted universities.
- G_{1112}: An information package is sent to the undergraduate office, and an e-mail is sent to each member of academic staff in the targeted universities.

Let us now consider the issue of goal influences. Notice that goal G_{1111} influences goal G_2 negatively due to the cost of producing and sending out many information packages. Also, goal G_{1112} influences G_2 negatively but not as much as G_{1111} because the cost in this case is small. Since G_2 is an objective that needs to be satisfied, the analyst should choose goal G_{1111} for implementation and not goal G_{1112}. The issue of goal influencing has been studied in the context of non-functional requirements by [Mylopoulos *et al.*, 1992; Chung, 1993] and their methodology and algorithms are directly applicable to our case. In this context the issue of decision rationale is also important [Lee, 1990]. For every choice among alternatives that we make, our decision rationale must be recorded explicitly so it can guide future changes to the developed process. To achieve this we use an *issue-argumentation* model in the spirit of [Lee, 1990] and [Yu and Mylopoulos, 1994b]. We omit the details of this model due to limited space.

Let us now consider goal G_{13}. This goal can be AND-decomposed into the following subgoals:

- G_{131}: All correspondence with applicants is promptly handled by a member of administrative staff.
- G_{132}: Applications that arrive are promptly forwarded to a member of academic staff who does an initial evaluation of each application immediately. If an application is promising, it is immediately forwarded to members of academic staff whose research profile matches the wishes of the applicant.
- G_{133}: Applications received by members of academic staff are evaluated immediately upon receipt.

At this point the need for specifying *organisational roles* that will take over responsibility for satisfying one or more of the above goals becomes clear.

6.2 Identifying Roles and their Responsibilities

The second step of the methodology is the identification of roles and their responsibilities. Role identification is achieved by interacting with the enterprise stakeholders and by considering goals at the lowest level of the developed goal hierarchy. Given one of these goals and the roles currently existing in the organisation, the analyst should then decide whether one of these roles (or a new one) can be designated as responsible for achieving the goal. If this is possible then the goal becomes a role responsibility, otherwise it needs to be refined further.

Role identification is not an easy task and the following heuristics can be used to guide the analyst in role identification [Ould, 1994]. These heuristics are also relevant to the next two steps of the methodology.

- Identify roles with job titles or posts only if the responsibilities of the role form a substantial part of the responsibilities of the post.
- Define roles that have high cohesion i.e., the activities that form the role are closely related, and collectively serve an organisational goal.
- Define roles that are loosely coupled i.e., they do not involve too much interaction. Consider whether roles involving too much interaction can be merged into a single role.
- Choose roles that capture *what* we are doing rather than *how* or by *whom* it is done.
- Avoid introducing roles that have few activities of their own or are simply third parties to other roles' interactions.

The activities of goal reduction (Step 1), role identification and responsibility matching (Step 2) will benefit from proceeding in parallel so that they can interact with each other.

Example

In our example two new roles need to be introduced: the Postgraduate Tutor (notation: $Tutor$) and the Postgraduate Secretary (notation: $Secretary$). These two roles will interact with the already existing role of member of academic staff (notation: $Faculty$). For our purposes it is not necessary to introduce a role for applicants. Applicants will be considered to be outside of this process and interaction with them could be captured through the concept of exogenous actions.

The Postgraduate Secretary will be responsible for handling all correspondence with applicants but also for forwarding applications to the Postgraduate Tutor. The Postgraduate Tutor will be responsible for doing an initial evaluation of applications and forwarding applications to appropriate members of academic staff. Members of academic staff will be responsible for evaluating promptly all applications they receive.

Now that roles have been identified and responsibilities assigned, the decomposition of goal G_{13} can be made more precise:

- G_{131}: The Postgraduate Secretary forwards applications to the Postgraduate Tutor as soon as they arrive.
- G_{132}: The Postgraduate Tutor does an initial evaluation of each application as soon as they arrive on his/her desk.
- G_{133}: Applications sent to members of academic staff by the Postgraduate Tutor are evaluate immediately upon receipt.
- G_{134}: Decisions are posted to applicants immediately.

6.3 Specifying the Internal Structure of Roles

In this step we specify the *primitive actions* (physical or knowledge producing) that are available to each role, the *conditions* to be monitored and the *interactions* with other roles. The formal machinery for specifying these concepts has already been presented in Section 4.

Example

For our example let us first consider the role $Tutor$. This role can perform the causal action $ForwardApp$ (defined in Example 4.1) and the knowledge producing action $SendMsg(sender, recipient, msg)$ which means that actor $sender$ sends message msg to actor $recipient$. A precise specification of $SendMsg$ and other useful communicative actions in situation calculus can be found in [Lesperance *et al.*, 1995].

Role $Tutor$ also needs to watch for condition $Has(actor, app)$ where $actor$ is the actor playing the role $Tutor$ and app is an application. Similarly, we can specify the internal structure for the other two roles.

6.4 Specifying ConGolog Procedures Local to Each Role

In this step of the methodology the detailed specification of the dynamics of each roles is given using the syntax of Section 4. For each role, the business analyst has to specify a ConGolog procedure called *main* which gives the details of the behaviour of the role. Of course, *main* can invoke other local procedures.

This step is strongly related to the previous one and in practice analysts may find it beneficial to run these steps in parallel. In the processes we have modelled so far, we have found ConGolog very natural and easy to use. In most cases it was straightforward to write a piece of ConGolog code for each responsibility of a role, and then combine those pieces to form a complete specification of the dynamics of the role. We expect to come up with more precise guidelines for using the language as our experience with it increases.

Example

For our example the procedures local to the three roles are shown in Figures 1, 2 and 3. There are two notational conventions to be aware of:

- $self$ is a pseudo-variable denoting the actor playing the role inside which this variable appears.
- The first argument of action terms is omitted since it is always the actor executing the action (i.e., $self$) as agreed in Section 4. For example, the action $ForwardApp$ which has been defined to have three arguments in Definition 4.1 appears with its first argument omitted in the ConGolog code for roles $Tutor$ and $Secretary$.

With the above conventions in mind the ConGolog code should be easy to understand. The reader should notice how natural it is to specify in ConGolog reactive processes using interrupts and concurrency. The only complication is in role *Secretary* where a message queue (in the spirit of [Lesperance *et al.*, 1995]) is used. Note also that the code for secretary does not handle the case where more than one members of academic staff want to supervise the same applicant. We also omit the specification of exogenous actions that capture the interaction between the role *Secretary* and the applicants (that are part of the outside environment).

Given the specifications for roles $Secretary, Tutor$ and $Faculty$, the specification of the complete business process is straighforward using the syntax of Section 4.

6.5 Formal Verification

In this step we *verify formally* that each role responsibility and each constraint is maintained by the ConGolog procedures defined locally for each role.

role $Tutor$
responsibility G_{132}

proc $main$
$\langle app: \ Has(self, app) \rightarrow$
  if $AvgMark(app) < 70$ **then**
    for $act: \ PlaysRole(act, Secretary)$ **do**
      $SendMsg(act, \ulcorner INFORM(Unacceptable(app))\urcorner)$
    endFor
  else
    for $act: \ PlaysRole(act, Lecturer)$ **do**
      $ForwardApp(act, app)$
    endFor
  endIf
$\rangle$
endProc

endRole

Figure 1: Role Postgraduate Tutor

role $Secretary$
responsibility $G_{12}, \ G_{131}, \ G_{134}$

proc $main$
$\langle infoReq \ : \ Received(self, infoReq) \ \rightarrow ReplyTo(infoReq)\rangle$
$\gg$
$\langle app: \ Has(self, app) \rightarrow$
  for $act: PlaysRole(act, Tutor)$ **do**
    $Forward(act, app)$
  endFor
$\rangle$
$\gg$
while True do
  $SenseMsg$;
  if $\neg Empty(MsgQ(self))$ **then**
    if $First(MsgQ(self)) = (lect, \ulcorner INFORM(WantsToSupervise(lect, app))\urcorner)$ **then**
      $SendOfferLetter(app)$
    else if $First(MsgQ(self)) = (tut, \ulcorner INFORM(Unacceptable(app))\urcorner)$ **then**
      $SendRejectionLetter(app)$
    endIf
  endIf
endWhile
endProc

endRole

Figure 2: Role Postgraduate Secretary

role $Faculty$
responsibility G_{133}

proc $Eval(app)$
if $GoodUniv(Univ(app)) \wedge AvgMark(app) > MinMark(self) \wedge NoOfStud(self) < MaxNoOfStud(self)$ **then**
  for $act:\ PlaysRole(act, Secretary)$ **do**
    $SendMsg(act,$
    $\ulcorner INFORM(WantsToSupervise(self, app))\urcorner)$
  endFor
endIf
endProc

proc $main$
$\langle app:\ Has(self, app) \rightarrow Eval(app)\rangle$
endProc

endRole

Figure 3: Role Academic Member of Staff

To perform verification we utilize the techniques reported in [Plexousakis, 1995; 1996], which are based on a systematic solution to the well-known in AI frame and ramification problems [Reiter, 1991]. Specifically, we are interested in determining whether: (i) responsibilities of roles can be fulfilled, and (ii) constraints defined in the constraints submodel are preserved or violated as a result of process execution. These questions are answered by reasoning with the process specification resulting from the previous step of the methodology. In case where such a proof or disproof is not possible at process specification time, strengthenings to the specifications of actions that are relevant to the responsibilities/constraints are proposed, so that any process implementation meeting the strengthened specifications provably guarantees that the responsibilities/constraints will be satisfied in the state resulting from action execution. The method proceeds by deriving ramifications of constraints and action preconditions and effects, and by using these ramifications to strengthen the action specifications [Plexousakis, 1995; 1996].

Example 6.1 Let us consider the specification of the action $SendOfferLetter$ shown below (this is a simplified version of the action used in role $Secretary$). The predicate $Accepted(app)$ denotes that application *app* has been accepted by DEPT. Similarly, $WantsToSupervise(lect, app)$ means that academic *lect* would like to supervise the student of application *app*.

action $SendOfferLetter(app)$
  precondition
    $(\exists\, lect)\, WantsToSupervise(lect, app)$
  effect $Accepted(app)$
endAction

Assume that we wish to enforce the policy that no applicant can be both accepted and rejected. This constraint may be expressed by the following sentence of $\mathcal{L}$ (and belongs to the constraints submodel):[3]

$$(\forall p)(Accepted(p) \supset \neg Rejected(p))$$

It is evident that the action specification given above does not exclude a situation in which both $Accepted(app)$ and $Rejected(app)$ are satisfied. We can easily see that if the constraint is to be preserved in the situation resulting from performing action $SendOfferLetter$ then $\neg Rejected(p)$ is a logical implication of the constraint, i.e., a ramification of the constraint and the action specification. Our ramification generator proposes that the term $\neg Rejected(p)$ be used to strengthen the action specification (by conjoining the term with the action precondition or effect). The strengthened specification is now guaranteed not to violate the constraint in any possible execution of the action $SendOfferLetter$.

Albeit short[4] and simple, the above example conveys the idea behind the derivation of ramifications for strengthening action specifications. More complex examples and details of the generation process can be found in [Plexousakis, 1995; 1996]. The same ideas can be used to verify formally that roles fulfill the responsibilities assigned to them.

The aforementioned work provides results for verifying properties of primitive actions and of processes including sequencing of actions, when the constraints refer to at most two distinct states. The derivation of similar results for processes synthesized using any of the remaining ConGolog constructs - including concurrency and nondeterminism - and for general dynamic constraints is a topic of current research. Our previous work can also accommodate knowledge-producing actions in a single-agent environment. The theoretical basis of ConGolog has been extended to include exogenous and knowledge-producing actions in a multi-agent environment [Lesperance *et al.*, 1999b]. The adaptation of these ideas in our analysis and verification techniques is an ongoing effort.

We argue that the ability to verify properties of processes is essential for business process design and re-engineering. The process specifier realizes the implications of actions as far as goal achievement is concerned and the implementor is saved the burden of having to find ways to meet postconditions and maintain invariants. Furthermore, optimized forms of conditions to be verified can be incorporated into process specifications and consistency is guaranteed by the soundness of the verification process [Plexousakis, 1996].

[3] The predicate $Rejected$ denotes that some applicant has been rejected by DEPT.

[4] We have intentionally omitted presenting all the steps in the generation process due to lack of space.

7 Discussion

The first paper to propose situation calculus and ConGolog (more precisely its earlier version Golog) for business process modeling was [Plexousakis, 1995]. Since then similar ideas have appeared in [Plexousakis, 1996; Yu *et al.*, 1996; Lesperance *et al.*, 1999a]. But so far, ConGolog has not been used in conjunction with a more general framework like ours that offers intentional concepts like actors, roles and goals. Situation calculus is also the formalism of choice for the TOVE enterprise modelling project [Fox and Gruninger, 1998].

The concepts of goals, actors and roles also appear prominently in the i^* framework [Yu, 1994] where the need for *intentional concepts* in enterprise modeling (and requirements modeling) is emphasized. i^* also supports the concept of dependency between actors something which is not offered by our framework (and would be a nice addition to it). Our methodology can undoubtedly benefit by incorporating the i^* framework, which, in addition to dependencies, models strategic rationale and intentional relationships such as, e.g., commitment.

There is a clear connection of our work to goal-oriented methodologies for requirements engineering especially KAOS [Dardenne *et al.*, 1993]. This connection has been explained in detail in previous sections of this paper so we will not elaborate on it here.

Our work is also related to the enterprise modeling frameworks of F^3 [Bubenko, 1994; Loucopoulos and Kavakli, 1995] and its successor EKD [Bubenko *et al.*, 1998]. In EKD knowledge about an organisation is partitioned into the following submodels: the goals submodel (corresponds to our objectives and goals submodel), the actors and resources submodel (roughly corresponds to our organisational submodels), the business processes submodel (corresponds to our process submodel), the concepts submodel (corresponds exactly to concepts submodel), the business rules submodel (it is more involved than our constraints submodel), and the technical components and requirements submodel (no corresponding concept in our work). In terms of formalisms, EKD uses entity-relationship models to represent structural information and Role-Activity Diagrams [Ould, 1994] to represent roles and their activities. Our proposal, compared with EKD, offers more expressive languages (situation calculus and ConGolog) and is therefore more amenable to formal reasoning. On the other hand, we have not attempted to be as comprehensive as EKD in our coverage of issues related to enterprise modeling, and, so far, we have not tried our models and methodology in significant industrial applications.

Lee's Goal-based Process Analysis (GPA) is also related to our research [Lee, 1994]. GPA is a goal-oriented method and can be used to analyse existing processes in order to identify missing goals, ensure implementation of all goals, identify non-functional parts of a process, and explore alternatives to a given process.

Finally, our work has many common ideas with the GEM models and methodology [Rao, 1996]. According to GEM business processes are collections of suitably ordered activities, enacted by individual persons, depending on their role within an organisation. Every process has a purpose which is to achieve a goal or react to an event. GEM offers a number of models that can be used for specifying processes: the role interaction model, the purpose model, the procedure model, the internal data model and the corporate data model. The GEM methodology consists of three steps: defining the scope of the business process, doing process analysis and doing system design. The process analysis step consists of the following stages: goal hierarchy analysis, basic procedure analysis, detailed procedure analysis, input/output data analysis and performance metrics specification. We have been unable to make a more detailed comparison of our work with GEM because the only related document publicly available [Rao, 1996] gives only a short informal description of the models and methodology. A methodology for developing multi-agent systems based on concepts similar to the ones in GEM appears in [Kinny *et al.*, 1996].

The vast majority of business process modeling efforts lack formal methods for verifying properties of processes. A user-assisted verification tool handling arbitrary ConGolog theories is currently under development as reported in [Lesperance *et al.*, 1999a]. Strengthening of specifications using inference rules has also been proposed in [Dardenne *et al.*, 1993] in the context of the KAOS project. Verification of process properties however is not treated systematically.

Orthogonally to verification, validation tools may be employed for testing whether processes execute as expected in various conditions. In [van Lamsweerde *et al.*, 1995] the use of operational scenarios is proposed for discovering overlooked aspects of the specified model, such as, e.g., missing goals. A simulation tool based on logic programming has been developed for validating ConGolog processes [Lesperance *et al.*, 1999a]. The tool also includes a module for progressing an initial situation, allowing it to simulate the execution of long-running processes.

Our long term research goal is to demonstrate that formal languages and methods can offer significant advantages in the design and analysis of business processes. Currently we are working on extending the techniques of [Plexousakis, 1995; 1996] to accomodate all features of ConGolog. In parallel we are in the process of applying our techniques to the modelling of large processes so that we can demonstrate clearly the benefits over other approaches. We are also interested in extending our methodology to deal with the problem of *business change* and investigate what formal techniques and reasoning can be beneficial in this case.

References

[Anton *et al.*, 1994] A.I. Anton, M.W. McCracken, and C. Potts. Goal decomposition and scenario analysis in business process reengineering. In *Proceedings of CAISE'94*, pages 94–104, 1994.

[Bubenko *et al.*, 1998] J. Bubenko, D. Brash, and J. Stirna. EKD user guide, 1998. Available from ftp://ftp.dsv.su.se/users/js/ekd_user_guide.pdf.

[Bubenko, 1994] J. Bubenko. Enterprise Modelling. *Ingenierie des Systems d' Information*, 6(2), 1994.

[Chung, 1993] L. Chung. *Represening and Using Non-Functional Requirements: A Process-Oriented Approach*. PhD thesis, Dept. of Computer Science, University of Toronto, 1993.

[Curtis *et al.*, 1992] B. Curtis, M. Kellner, and J. Over. Process Modelling. *Communications of ACM*, 35(9):75–90, 1992.

[Dardenne *et al.*, 1993] A. Dardenne, A. van Lamsweerde, and S. Fickas. Goal-Directed Requirements Acquisition. *Science of Computer Programming*, 20:3–50, 1993.

[Davenport, 1993] P.T. Davenport. *Process Innovation: Re-Engineering Work Through Information Technology*. Harvard Business School Press, 1993.

[De Giacomo *et al.*, 1997] G. De Giacomo, Y. Lesperance, and H. Levesque. Reasoning About Concurrent Execution, Prioritised Interrupts and Exogenous Actions in the Situation Calculus. In *Proceedings of IJCAI'97*, pages 1221–1226, August 1997.

[Dobson *et al.*, 1994] J.E. Dobson, A.J.C. Blyth, J. Chudge, and R. Strens. The ORDIT Approach to organisational requirements. In M. Jirotka and J. Goguen, editors, *Requirements Engineering: Social and Technical Issues*, pages 87–106. Academic Press, 1994.

[Fikes and Nilsson, 1971] R. Fikes and N. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208, 1971.

[Fox and Gruninger, 1998] M.S. Fox and M. Gruninger. Enterprise Modelling. *The AI Magazine*, pages 109–121, Fall 1998.

[Georgakopoulos *et al.*, 1995] D. Georgakopoulos, M. Hornick, and A. Sheth. An Overview of Worklfow Management: From Process Modelling to Workflow Automation Infrastructure. *Distributed and Parallel Databases*, 3:119–153, 1995.

[Hammer and Champy, 1993] M. Hammer and J. Champy. *Reengineering the Corporation: A Manifesto for Business Revolution*. Harper Collins, 1993.

[Hamscher, 1994] Walter Hamscher. AI in Business Process Reengineering. *AI Magazine*, 15(4), 1994. Report on the AAAI Workshop.

[IDEF,] http://www.idef.com/.

[Jennings *et al.*, 1996] N. R. Jennings, P. Faratin, M.J. Johnson, P. O'Brien, and M.E. Wiegand. Using Intelligent Agents to Manage Business Processes. In *Proceedings of the First International Conference on The Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM96)*, 1996.

[Kavakli and Loucopoulos, 1998] V. Kavakli and P. Loucopoulos. Goal-Driven Business Process Analysis - Application in Electricity Deregulation. In *Proceedings of CAISE'98*, 1998.

[Kinny *et al.*, 1996] D. Kinny, M. Georgeff, and A. Rao. A methodology and modelling technique for systems of bdi agents. In *Proceedings of MAAMAW-96*, 1996.

[Kirn and O'Hare, 1997] S. Kirn and G. O'Hare. *Cooperative Knowledge Processing: The Key Technology for Intelligent Organisations*. Springer, 1997.

[Lee, 1990] J. Lee. SIBYL: A Qualitative Decision Management System. In P.H. Winston and S.A Shellard, editors, *Artificial Intelligence at MIT: Expanding Frontiers*, volume 1, pages 105–133. MIT Press, 1990.

[Lee, 1994] J. Lee. Goal-Based Process Analysis: A Method for Systematic Process Redesign. In *Proceedings of the Conference on Organizational Computing Systems (COOCS'94)*, 1994.

[Lesperance *et al.*, 1995] Y. Lesperance, H.J. Levesque, F. Lin, D. Marcu, R. Reiter, and R.B. Scherl. Foundations of a Logical Approach to Agent Programming. In M. Wooldridge, J.P. Muller, and M. Tambe, editors, *Intelligent Agents Volume II – Proceedings of ATAL-95*, Lecture Notes in Artificial Intelligence. Springer Verlag, 1995.

[Lesperance *et al.*, 1999a] Y. Lesperance, T.G. Kelley, J. Mylopoulos, and E. Yu. Modeling dynamic domains with congolog. In *Proceedings of CAISE'99*, 1999. Forthcoming.

[Lesperance *et al.*, 1999b] Y. Lesperance, H. Levesque, and R. Reiter. A situation calculus approach to modeling and programming agents, 1999. Available from http://www.cs.toronto.edu/~cogrobo/.

[Leymann and Altenhuber, 1994] F. Leymann and W. Altenhuber. Managing Business Processes as an Information Resource. *IBM Systems Journal*, 33(2):326–348, 1994.

[Loucopoulos and Karakostas, 1995] P. Loucopoulos and V. Karakostas. *System Requirements Engineering*. McGraw Hill, 1995.

[Loucopoulos and Kavakli, 1995] P. Loucopoulos and V. Kavakli. Enterprise Modelling and the Teleological Approach to Requirements Engineering. *International Journal of Intelligent and Cooperative Information Systems*, 4(1):45–79, 1995.

[McCarthy and Hayes, 1969] John McCarthy and Patrick J. Hayes. Some Philosophical Problems From the Standpoint of Artificial Intelligence. In B. Meltzer and D. Mitchie, editors, *Machine Intelligence*, pages 463–502. Edinburg University Press, 1969.

[Mylopoulos *et al.*, 1992] J. Mylopoulos, L. Chung, and Nixon B.A. Representing and Using Non-Functional Requirements: A Process-Oriented Approach. *IEEE Transactions on Software Engineering*, 18(6):483–497, 1992.

[NIST-PSL,] http://www.mel.nist.gov/psl/.

[Ould, 1994] M. Ould. Modelling Business Processes for Understanding, Improvement and Enactment. Tutorial Notes, 13th International Conference on the Entity Relationship Approach (ER' 94), Manchester, U.K., 1994.

[Ould, 1995] M. A. Ould. *Business Processes: Modeling and Analysis for Re-engineering and Improvement*. Wiley, 1995.

[Plexousakis, 1995] D. Plexousakis. Simulation and Analysis of Business Processes Using GOLOG. In *Proceedings of the Conference on Organizational Computing Systems (COOCS'95)*, pages 311–323, 1995.

[Plexousakis, 1996] D. Plexousakis. *On the efficient maintenance of temporal integrity in knowledge bases*. PhD thesis, Dept. of Computer Science, University of Toronto, 1996.

[Rao, 1996] A. Rao. Modeling the service assurance process for Optus using GEM. Technical Report Technical Note 69, Australian Artificial Intelligence Institute, 1996.

[Reiter, 1991] R. Reiter. The Frame Problem in the Situation Calculus: A Simple Solution (Sometimes) and a Completeness Result for Goal Regression. In *Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy*, pages 359–380. Academic Press, 1991.

[Scherl and Levesque, 1993] R. Scherl and H. Levesque. The frame problem and knowledge producing actions. In *Proceedings of AAAI-93*, 1993.

[Tidhar, 1993] G. Tidhar. Team-oriented programming: Social structures. Technical Report Technical Note 47, Australian Artificial Intelligence Institute, 1993.

[van Lamsweerde *et al.*, 1995] A. van Lamsweerde, R. Darimont, and Massonet P. Goal-Directed Elaboration of Requirements for a Meeting Scheduler: Problems and Lessons Learned. In *Proceedings of RE'95*, 1995.

[van Lamsweerde *et al.*, 1998] A. van Lamsweerde, R. Darimont, and E. Letier. Managing Conflicts in Goal-Driven Requirements Engineering. *IEEE Transactions on Software Engineering*, November 1998. Special Issue on Managing Inconsistency in Software Development.

[WFMC,] http://www.wfmc.org/.

[Yu and Mylopoulos, 1994a] E. Yu and J. Mylopoulos. Understanding "Why" in Software Process Modelling. In *Proceedings of the 16th International Conference on Software Engineering*, pages 135–147, Sorrento, Italy, 1994.

[Yu and Mylopoulos, 1994b] E. Yu and J. Mylopoulos. Using Goals, Rules and Methods to Support Reasoning in Business Process Reengineering. In *Proceedings of the 27th Annual Hawaii International Conference on Systems Sciences*, pages 234–243, Hawaii, 1994.

[Yu *et al.*, 1996] E. Yu, J. Mylopoulos, and Y. Lesperance. AI Models for Business Process Reengineering. *IEEE Expert*, 11(4):16–23, 1996.

[Yu, 1994] E. Yu. *Modelling Strategic Relationships For Process Reengineering*. PhD thesis, Dept. of Computer Science, University of Toronto, 1994.