AI Models for Business Process Reengineering

Eric S.K. Yu and John Mylopoulos, University of Toronto
Yves Lespérance, York University

MOST MODELS FAIL TO CAPTURE THE RATIONALE BEHIND PROCESSES, MAKING BUSINESS REENGINEERING LESS EFFECTIVE. THE AUTHORS DESCRIBE THEIR i FRAMEWORK, WHICH VIEWS ORGANIZATIONS AS COLLECTIONS OF ACTORS WITH STRATEGIC INTERESTS, AND INTERDEPENDENCIES INVOLVING GOALS, TASKS, AND RESOURCES. THE AUTHORS ALSO DESCRIBE THE CONGOLOG FRAMEWORK, WHICH SUPPORTS REASONING ABOUT THE DYNAMICS OF PROCESSES UNDER INCOMPLETE KNOWLEDGE.*

COMPETITIVE PRESSURES, customer demands, and ever-changing regulatory conditions are forcing many companies to rethink the way they do business. A fundamental part of this rethinking process is to link production procedures and organizational services to business goals and objectives. At present, there is little formal support for this kind of reasoning. Business process design is usually done informally, and individual design decisions are hard to relate to business objectives.

Traditional modeling techniques—structured analysis, dataflow diagrams, and entity-relationship diagrams—describe *what* a business process is, but they cannot express the *why* of the process—the motivations, intents, and rationales behind the activities and entities. This is a serious drawback. A central argument in business process reengineering is that if you don't understand why things are done the way they are, you are likely to automate outdated processes (leading to the proverbial "paving of the cow path") and miss the opportunity to innovate the process itself.

In choosing among alternative business processes, analysts must be able both to describe relationships and to propose and argue about solutions from strategic perspectives. Artificial intelligence in general and knowledge representation in particular can help in modeling organizations and in analyzing alternatives.

Recognizing the value of AI in this area, we developed a framework for modeling and analyzing organizations in support of several applications, including business process reengineering.[1,2] The *i** framework (*i** stands for distributed intentionality) views processes as involving social actors who depend on one another for goals to be achieved, tasks to be performed, and resources to be furnished. The *i** framework includes two models:

- *Strategic Dependency Model*, which describes the network of relationships among actors.
- *Strategic Rationale Model*, which describes and supports the reasoning that each actor has about its relationships with other actors.

We have formally represented these models in the conceptual modeling language Telos[3] and have based their semantics on intentional concepts—goal, belief, ability, and commitment—studied in work by Philip Cohen and Hector Levesque.[4]

To complement the *i** framework's strategic level of reasoning, we use a logic-based framework, ConGolog, to model the detailed dynamics of the business environment and processes being redesigned. The framework supports the validation and verification of business processes using simulation and automated reasoning techniques. Users can

0885-9000/96/$4.00 © 1996 IEEE

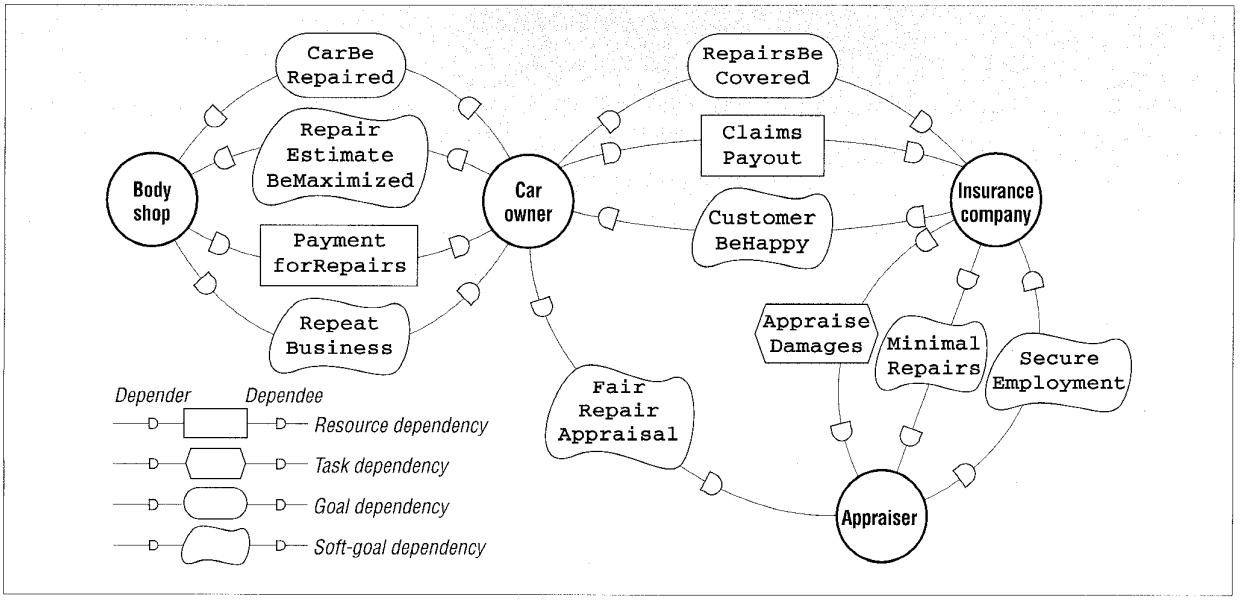

Figure 1. Strategic Dependency Model of traditional claims processing in an automobile insurance company (the "as is" arrangement).

reason about processes even with only a partial description of the world state.

In this article, we show how the *i** models and ConGolog aid the redesign of claims processing in an automobile insurance company. We also describe a toolset that aids analysis using these models, which we are currently developing at the University of Toronto. The models and their associated tools incorporate a number of AI techniques, including means-ends analysis, qualitative reasoning, agent modeling, and theories of action.

The problem

In business processes—and the information systems that support them—there is a kind of understanding entrenched as implicit assumptions. Organizations often ignore these assumptions during analysis because they are using traditional modeling techniques. Yet, without this deeper level of understanding, an organization will find it difficult to meet changing needs, as evidenced by the problem of legacy business processes and legacy information systems.

Most business processes involve many participants or players, with complex relationships among them. These relationships are *strategic* in the sense that each party is concerned about opportunities and vulnerabilities and seeks to protect or further its interests. To illustrate our framework, we borrow a hypothetical example from Michael Hammer and James Champy, who describe a redesign of an automobile insurance company's business processes.[5] Our focus here is the attempt to redesign claims processing.

In analyzing the problem, the team might ask

- Why does it take so long to have a claim settled after an automobile accident?
- Why does the company hire appraisers to assess damages?
- How else can claims be settled?
- What other concerns would arise if these new ways of handling claims were adopted?

The parties concerned are the company and its representatives and the customer. The company wants to minimize payout to claimants, so it hires appraisers, who keep repairs to the necessary minimum. At the same time, it wants to keep customers happy so that they continue to renew their policies. Car owners (the customers), on the other hand, want repair damages to be assessed fairly, and are likely to get body shops to give repair estimates that maximize the payout. The information that the claims representative collects and uses (accident particulars, witness statements) and that the appraiser collects and uses (photographs of damage, multiple repair estimates) reflect the strategic interests of these parties.

Strategic Dependency Model

The Strategic Dependency Model is a graph in which each node represents an actor, and each link between two actors indicates that Actor 1 depends on Actor 2 for something that will let Actor 1 attain some goal. Without that something, Actor 1 would not be able to achieve the goal as easily, as well, or at all. We call Actor 1 the *depender*, and Actor 2 the *dependee*. The object around which the dependency centers is the *dependum*. The depender is vulnerable to the dependee. If the dependee fails to deliver the dependum, the depender would be adversely affected in its ability to achieve its goals. A car owner, for example, can have his car repaired by a body shop, even if he does not have the ability to do the repairs himself. However, he is vulnerable to the car not being repaired.

Figure 1 shows the Strategic Dependency Model for a traditional automobile insurance business configuration. The model specifies four types of dependencies, according to the type of freedom allowed in the relationship between the depender and the dependee.

- In a *goal dependency,* an actor depends on another actor to bring about a certain state or condition in the world. The dependee is given the freedom to choose how to accomplish it. The car owner in Figure 1, for example, just wants the car to be repaired; he does not (usually) tell the body shop how to repair it.
- In a *task dependency,* an actor depends on another to carry out an activity. The task specification prescribes how the task is to be performed. In Figure 1, for example, an appraiser must follow company procedures when appraising damages.
- In a *resource dependency,* an actor

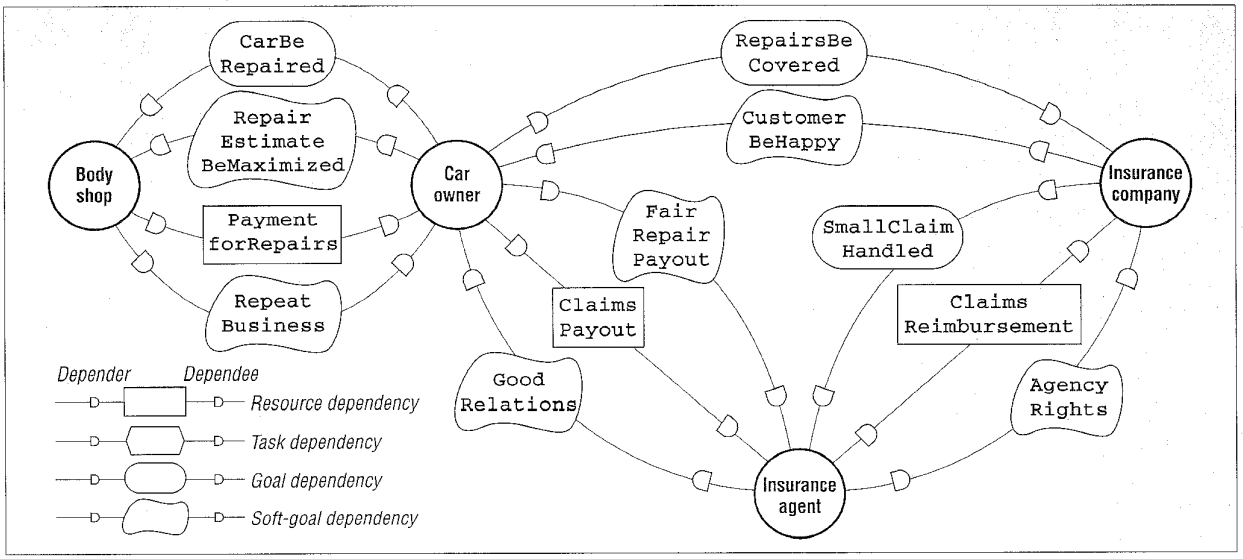

Figure 2. Strategic Dependency Model for redesign proposal "Let the insurance agent handle the claims."

depends on another for the availability of some entity. Resources can be physical or informational. In Figure 1, for example, the body shop expects the car owner to pay for repairs.

- A *soft-goal dependency* is similar to a goal dependency except that there are no a priori, sharply defined criteria for success. The meaning of the soft goal is elaborated on and clarified between the depender and the dependee in terms of the methods that might be used to address it. In Figure 1, for example, insurance companies can use various methods to attract and keep customers, but it is up to the customers to decide which methods make them happy.

The model also distinguishes three levels of dependency strength—open, committed, and critical—according to vulnerability. We do not show these in the figure, but briefly they are

- *Open.* Not obtaining the dependum would affect the depender to some extent but not seriously.
- *Committed.* Not obtaining the dependum would cause some action to fail that the depender has planned in achieving a goal.
- *Critical.* Not obtaining the dependum would cause all actions to fail that the depender has planned in achieving a goal.

Figure 1 captures several dependencies. The car owner depends on the insurance company to reimburse him for the repairs from an accident (`ClaimsPayout`). For this, the car owner pays an insurance premium to get coverage (`RepairsBeCovered`). The insurance company wants to offer good service to the customer to keep the business (`CustomerBeHappy`). To maintain profitability, the company depends on appraisers to appraise damages so that only the minimal necessary repairs are approved. The car owner depends on the claims appraiser for a fair appraisal. However, the appraiser can be expected to act in the interests of the insurance company, because she depends on the company for employment. The car owner, in turn, can depend on the body shop to give an estimate that maximizes his interests, because the body shop depends on the car owner for repeat business.

This kind of analysis reveals important aspects of the claims process that traditional process models would miss.

Figure 2 shows the changes in dependencies when the process-redesign team explores innovative solutions to claims processing. Because it costs as much to process a small claim as a large claim, one way to reduce administrative costs is to reduce insurance company involvement in handling small claims. The team asked, "Why not let the insurance agent handle small claims?" The insurance agent would do all the inquiry and payout, while the insurance company would concentrate on large claims that more significantly affect profit. The agent gets to cement his relationship with the customer, while the customer is more likely to get a fair hearing from the agent about a fair payout amount. This keeps the customer happy, which is what the insurance company wants.

As Figure 2 shows, shifting the claims-handling responsibilities from the appraiser to the insurance agent means that the agent's information needs must change radically. For example, the appraiser is not even involved in small claims. Using the new configuration of strategic dependencies, the team can derive the information that must be shared among or sent to insurance agents and the insurance company, and decide how accurate and up-to-date the information must be.

Once the team ventured from the traditional wisdom of how an insurance business should run, even more radical solutions emerged. One was "Why not let the body shop handle the claims?" Traditionally, body shops are not likely to be on the side of the insurance company. For example, you wouldn't expect an insurance company to be willing to pay according to a body shop's repair estimates, because the body shop is on the customer's side, as Figure 1 illustrates. However, for small claims, it might not be a bad idea to bypass all the paperwork and help the customer get his car fixed as quickly as possible. This meets the customer's goal of having his car fixed promptly and, at the same time, reduces costs dramatically for the insurance company.

Figure 3 shows the Strategic Dependency Model for this proposal. This proposal raises concerns about possible fraud, as indicated by the insurance company's dependencies on the car owner and on the body shop not to be fraudulent.

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 activities and entity flows. It helps identify what is at stake and

for whom, and what impacts are likely if a dependency fails.

Although a Strategic Dependency Model provide hints about why a process is structured in a certain way, it does not sufficiently support the process of suggesting, exploring, and evaluating alternative solutions. That is the role of the second model.

Strategic Rationale Model

Figure 4 shows the Strategic Rationale Model for the claims handling role in the redesign problem. Again, the graph has four node types—goal, task, resource, and soft goal—but it also describes the reasoning behind each actor's relationships with other actors.

Process components. A process is often depicted as a collection of activities with entity flows among them (as in a work-

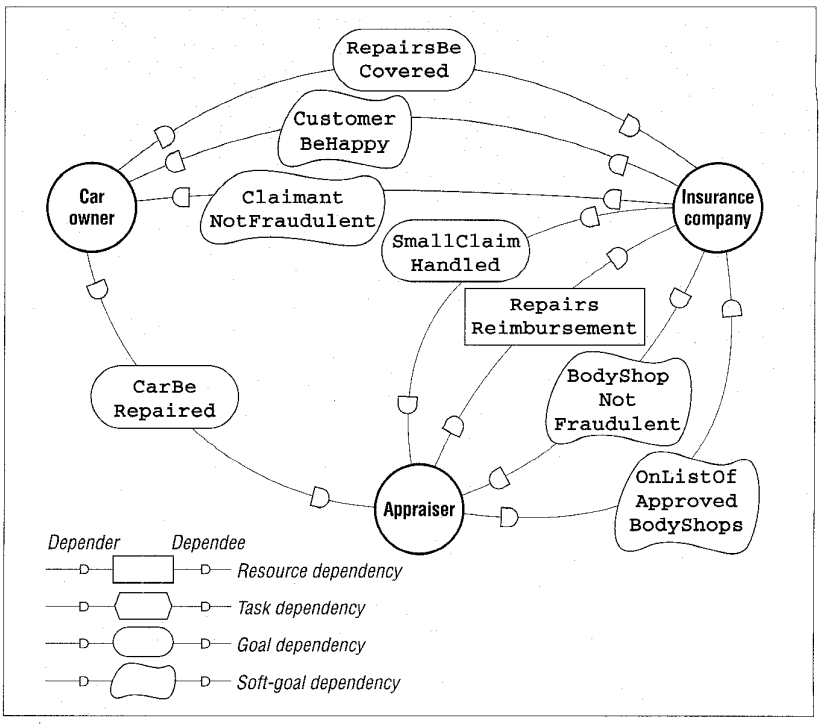

Figure 3. Strategic Dependency Model for redesign proposal "Let the body shop handle the claims."

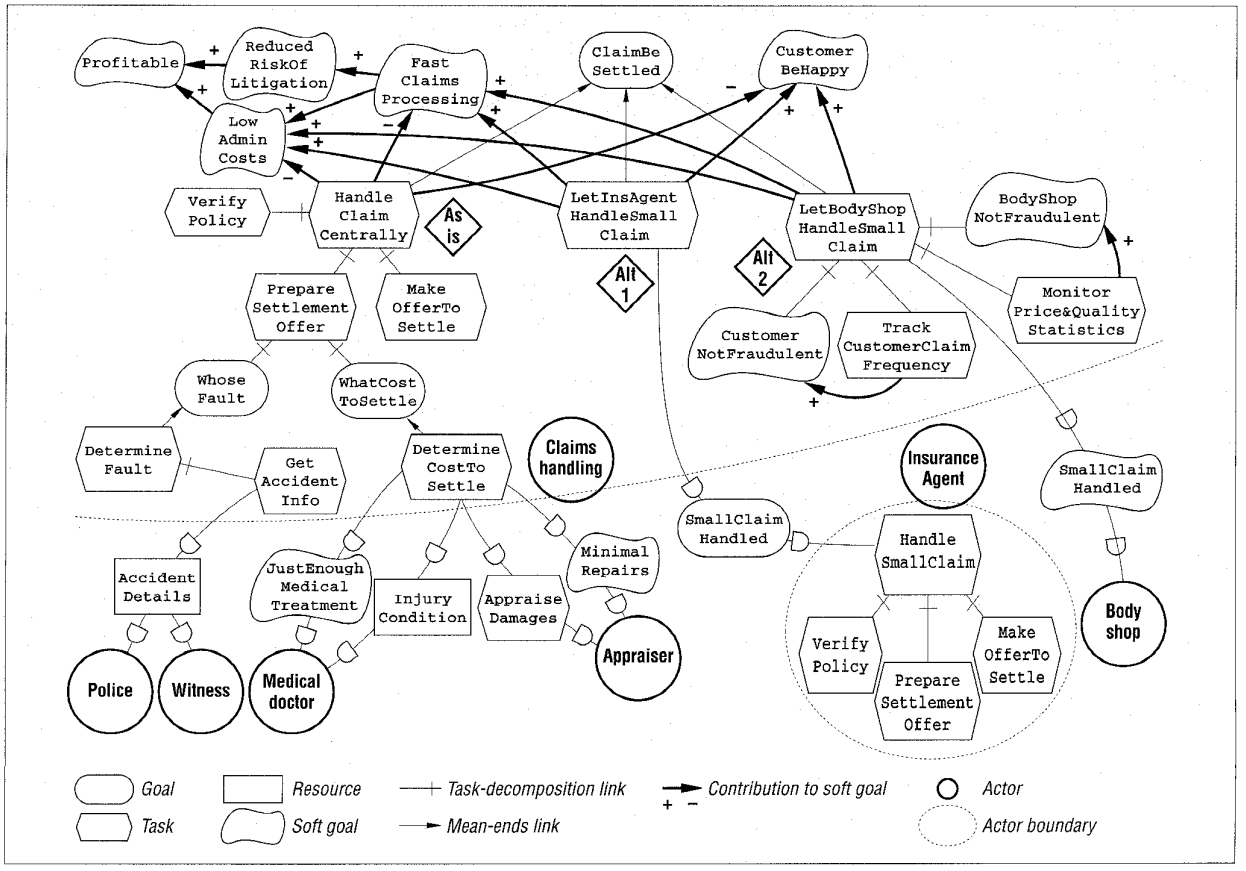

Figure 4. Strategic Rationale Model to support reasoning about reengineering the claims-handling process.

flow analysis). A claims-handling process includes activities such as verifying policy coverage, collecting accident information, determining who is at fault, appraising damages, and making an offer to settle. In the Strategic Rationale Model, we arrange these into a hierarchy of *means-ends* and *task decomposition* links.

A means-ends link indicates a relationship between an end and a means for attaining it. For example, `HandleClaimCentrally` is one way to achieve `ClaimBeSettled`. Figures 1 through 3 represent three ways to achieve the same goal: As is, Alternative 1, and Alternative 2. "As is" represents the approach in Figure 1; the alternatives represent the redesign approaches in Figures 2 and 3, respectively. Each approach contributes to the soft goals in different ways, such as lowering administrative costs or keeping customers happy.

By explicitly representing means-ends relationships, the Strategic Rationale Model provides a systematic way to explore the space of possible new process designs.

A task-decomposition link describes the components of a task in terms of its subgoals, subtasks, resources, and soft goals. For example, `HandleClaimCentrally` consists of the subtasks `VerifyPolicy`, `PrepareSettlementOffer`, and `MakeOfferToSettle`. Some task decompositions may involve dependencies on external actors.

Goal-based reasoning. In seeking ways to redesign a business process, goals are good places to look for improvement. An ambitious redesign must both discover and rethink high-level goals. The redesign team discovers high-level goals by asking *why* questions. Once the team identifies the high-level goals, they can seek alternatives by asking *how else* the organization can accomplish those goals. In the sample problem, the redesign team wanted to consider radical solutions. They first identified the high-level goal "Claims must be settled" and then began to think about how that goal could be accomplished other than to handle claims centrally.

As Figure 4 shows, each alternative has different implications for several soft goals, such as `CustomerBeHappy`, `FastProcessing`, and `Profitable`. Although you can measure and quantify some soft goals, a qualitative approach is suitable at this stage of exploring alternatives. Contributions to soft goals can be positive or negative, and are judged to be adequate or inadequate. (Our treatment of soft goals is based on a framework developed by Lawrence Chung for dealing with nonfunctional requirements in software engineering.[6])

Means-end links and soft-goal correlations can be viewed as instantiations of generic methods and rules. Consequently, process reengineers can use a knowledge base of generic business-process design knowledge in the form of methods and rules to suggest new solutions and to identify related goals during process design.[2]

ConGolog framework

The *i** models support strategic reasoning in exploratory design stages, but for validation and verification, process reengineers need a more precise characterization of the proposed processes. For example, a reengineer might want to simulate a process to observe its behavior under different conditions.

CONGOLOG'S MODELING CONSTRUCTS MAKE IT SUITABLE AS A DECLARATIVE MODELING LANGUAGE FOR BUSINESS PROCESSES.

Most existing process models are deficient in that they cannot deal with partial descriptions of world states. Such models describe processes only in terms of operations that produce new output states, given completely specified input states. Analysis of these models involves stepping through a process by instantiating it with a full description of a world state. However, business processes have uncertainties and are frequently open-ended or only partially specified. In modeling such processes, it is important to be able to specify and analyze *partial* models of business processes in the context of *partial* models of the world.

To address this need, we are adapting a logical framework for representing and reasoning about actions and processes.[7,8] The ConGolog (for concurrent Golog) framework addresses incompleteness in business-process specifications. The analyst models primitive actions by specifying their preconditions and postconditions in a logical language. Given a specification for a set of primitive actions, the analyst can then specify complex processes using various control structures, including nondeterministic choice and concurrent execution.

Figure 5 is an example of a ConGolog specification for a process that determines the cost to settle an insurance claim. Note how ConGolog specifications include declarations of pre- and postconditions for the primitive actions as well as procedures.

ConGolog's modeling constructs make it suitable as a declarative modeling language for business processes. The constructs include not only sequencing (';'), conditional (if-then) and iterative constructs (while <condition> do...) but also concurrent activity ('||'), nondeterministic choice (choose), and others. We have implemented a ConGolog interpreter in Prolog, which supports the simulation of a modeled process. Noteworthy features of the framework include the following:

- The primitive actions appearing in ConGolog processes can be arbitrary actions that change the world state (filing a claim-settlement report, for example), rather than just assignments to variables as in ordinary simulation languages like Simula; this works because the user declares the action's pre- and postconditions.
- The tests in ConGolog processes can refer to any world condition modeled; the interpreter automatically maintains the world model required to simulate the processes on the basis of the pre- and postconditions.
- The framework incorporates a solution to the so-called frame problem (discovering through a computationally tractable procedure all things that the specified process does not change). In the context of business reengineering, it relieves the redesigner from having to specify what aspects of the world don't change when an action is performed.

Analysts can evaluate process specifications in the framework using a variety of sophisticated reasoning methods. They can also use the framework to prove, simulate, and validate process properties. Finally, the framework includes a model of what agents know and how they can acquire additional

information through knowledge-producing actions, such as communicating with users or other agents or searching the environment for information.

Analysis and design tools

Apart from offering an inadequate ontology for organizational modeling, traditional organizational modeling techniques offer few tools to support the analysis of organizational models. In designing the *i** framework, we kept in mind the need for such tools. Our toolset, which is still under construction, simplifies business reengineering by assisting in the validation and verification of an organizational model and the exploration of design alternatives.

The toolset consists of tools to analyze strategic relationships, redesign strategic relationships, perform qualitative reasoning, validate a model, and verify a process.

```
precondition consultClaimFile(claim):
   ClaimMade(claim) and ClaimFileOpened(claim)

causes consultClaimFile(claim):
   KnowWhoIs(agent, claimant(claim))

% other preconditions and causes declarations left out

procedure determineCostToSettle(claim)
   consultClaimFile(claim);
   % concurrently get vehicle and medical appraisals
   ⟨  % pick a vehicle appraiser
      choose v [VehicleAppraiser(v)?;
         request(v,doVehicleAppraisal(claim))
      ]
   ||
      if ClaimInvolvesMedicalExpenses(claim) then
         % pick a medical appraiser
         choose m [ MedicalAppraiser(m)?;
            request(m,doMedicalAppraisal(claim))
         ]
   ⟩;
   consultMedicalAppraisalReport(claim);
   consultVehicleAppraisalReport(claim);
   fileCostToSettleReport(claim)
end procedure
```

Figure 5. Example in ConGolog of a complex process specification.

Strategic relationships analysis. This tool, based on the Strategic Dependency Model, lets analysts construct, refine, and analyze the network of strategic dependencies among actors: on the one hand, they open up *opportunities* (by letting actors achieve goals that are otherwise not achievable or not achievable to the same extent), but on the other hand, they bring *vulnerabilities* (because the dependee actors might not deliver). Analysis with this tool focuses on opportunities and vulnerabilities, and patterns of dependencies based on the concepts of *enforcement*, *assurance*, and *insurance*. For a more detailed analysis of dependency relationships, analysts can further differentiate actors into physically embodied *agents*, the *roles* they play, and the *positions* they occupy.[1]

The tool also includes a graphical user interface for presenting and manipulating the model.

For example, an analyst might use the tool to construct and analyze the claim-processing model of Figure 1 (or its alternatives in Figures 2 and 3), noting goals not being achieved, tasks not being accomplished, or resources not being furnished. The tool can also note long chains of dependencies that suggest vulnerabilities, or dependency patterns that define conflict-of-interest situations.

Strategic relationships redesign. This tool, based on the Strategic Rationale Model, explicitly supports the means-ends reasoning behind the design of business processes. The basic idea of the tool is that you can obtain an understanding of the *why* behind process elements (or steps) by following (querying) their links to process design goals (*up* the means-ends hierarchy), extending the Strategic Rationale Model when appropriate. Alternatively, given some design goals, you can explore alternative ways for achieving them (*down* the means-ends hierarchy). A knowledge base of generic means-end knowledge (methods for reducing errors, preventing fraud, and so on) and some knowledge-structuring mechanisms, such as classification and generalization, might simplify the process of traversing the means-end hierarchy. You could also use *correlation rules*[6] to help detect cross-impacts among goals and identify design tradeoffs. We expect this tool to be highly interactive and iterative.

One of the challenges in constructing a tool like this is to collect a representative body of means-ends knowledge in business process redesign to illustrate the tool's practical utility. We have taken a first step toward meeting this challenge by collecting methods to achieve security, accuracy, and performance soft goals in the context of nonfunctional requirements for information system design.[6]

Qualitative reasoning support. This tool, adapted from work by Lawrence Chung,[6] is designed to make it easier to analyze a collection of interdependent soft goals. In particular, given a soft-goal dependency graph and a list of *satisficed* and *unsatisficeable* soft-goal nodes, the tool can label other soft goals as satisficed, unsatisficeable, or otherwise, depending on the types and destinations of the interdependencies for each soft goal. (We adapted the terms "satisficed" and "satisficeable" from work by Herbert Simon,[9] who coined the word "satisficing." The terms mean that the soft goal has been met to a sufficient, satisfactory degree.) The algorithms for propagating labels across the soft-goal dependency graph use qualitative reasoning techniques.[6]

The qualitative component of design reasoning is essential in supporting business process redesign. While quantitative measures are important in evaluating finished designs, they are of limited use *during* the iterative design phase, when many competing (or complementary) goals must be traded off and new alternatives explored. There is virtually no support for this type of analysis in existing tools.

Process model validation. This tool helps confirm that a process model is consistent

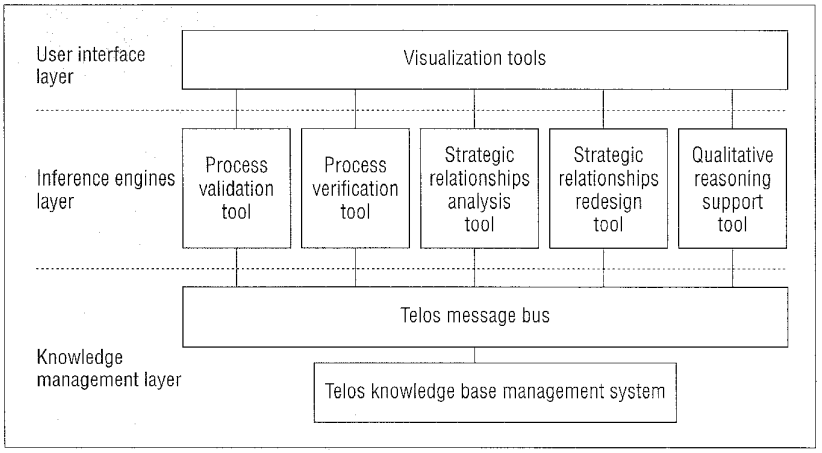

Figure 6. Tool-integration architecture.

with the modeler's understanding of the process. The tool validates the process model by letting the user *simulate* process execution. Given a description of the conditions in effect at the beginning of a business process, the tool answers queries about the world state as the process proceeds. The tool offers a declarative language for process specification and can simulate processes even when a process or its initial state are only partially specified.

This tool, under development, derives from the ConGolog interpreter. Given a process specification and a partial description of an organizational state, the simulation tool will answer questions about the state of the organization during and after the process has been carried out. Such symbolic simulations require a sophisticated theoretical foundation for reasoning about action, including handling the frame problem.

Process verification. Along similar grounds, the goal of the process verification tool, also being developed, is to help verify that a specified process satisfies given properties—in particular, state constraints (analogous to integrity constraints in databases). Given a set of process specifications and a set of constraints that the process must maintain, the tool will suggest strengthened specifications to ensure that those constraints are maintained.

Like the validation tool, this tool must address long-standing AI dilemmas, such as the frame problem (how to specify actions without enumerating all the properties of the world that are not affected) and the ramification problem (how to compute the indirect effects of actions). Although these two tools are based on the same theoretical foundation, we expect them to use different inference engines, employ different algorithms, and handle different classes of problems.

Integration issues. Although each tool now exists separately as a research prototype, we plan for them to work synergistically through a knowledge management layer. Figure 6 illustrates the proposed integration.[10]

The integrated architecture will employ a *message bus*, whose function is to support information interchange among tools that, although developed independently, can benefit from information sharing. The bus we plan to use, the *Telos Message Bus,* was developed in the context of a software-engineering research project. In its current form, the bus uses an extensible object model, Telos,[3] along with a message-based communication protocol system called *mbus*. In addition, the integration architecture includes a repository, which uses the Telos knowledge-base management system to store sharable information among the integrated tools. We have already implemented the repository in C++, using a commercial object-oriented database system as its persistent storage manager.

AS ORGANIZATIONS STRIVE TO keep up, there will be a greater demand for modeling and analysis tools that can help capture and analyze the strategic relationships among business work units and external players, such as customers, suppliers, and business partners. Adopting a knowledge representation and reasoning approach to organizational modeling and reengineering makes it easier to connect business processes and underlying business objectives. It also provides the formal foundation for several different analysis tools that can help analysts explore alternatives and lend credibility to the product of their work. Models constructed in terms of our approach also make it easier to develop information systems to support the reengineered business processes.

Our current effort is to complete the integration of the toolset and integrate the *i** models and the ConGolog framework into a single framework, which we plan to call Tropos. Among other things, Tropos will use a context mechanism to support the management of multiple actor viewpoints and to keep track of design alternatives. We are also exploring the use of verification techniques.[11]

Efforts are also underway to explore methodological issues in applying the modeling and reasoning techniques to several case studies, including one involving procedures for power-plant operators.

Acknowledgments

This research is funded in part by the Information Technology Research Centre of Ontario and by the Natural Sciences and Engineering Research Council of Canada.

References

1. E. Yu and J. Mylopoulos, "Understanding 'Why' in Software Process Modeling, Analysis, and Design," *Proc. 16th Int'l Conf. Software Engineering*, IEEE CS Press, Los Alamitos, Calif., 1994, pp. 159–168.
2. E. Yu, *Modeling Strategic Relationships for Process Reengineering*, doctoral dissertation, Dept. of Computer Science, Univ. of Toronto, 1995.
3. J. Mylopoulos et al., "Telos: Representing Knowledge About Information Systems," *ACM Trans. Information Systems*, Oct. 1990.
4. P.R. Cohen and H.J. Levesque, "Intention is Choice with Commitment," *Artificial Intelligence*, Vol. 42, No. 3, Mar. 1990.
5. M. Hammer and J. Champy, *Reengineering the Corporation: A Manifesto for Business Revolution*, HarperBusiness, New York, 1993.
6. K.L. Chung, *Representing and Using Non-Functional Requirements*, doctoral disserta-

tion, Dept. of Computer Science, Univ. of Toronto, 1993.

7. Y. Lespérance et al., "Foundations of a Logical Approach to Agent Programming," in *Intelligent Agents—Volume II: Proc. 1995 Workshop on Agent Theories, Architectures, and Languages*, Springer-Verlag, Berlin, 1996, pp. 331–346.
8. H. Levesque et al., "GOLOG: A Logic Programming Language for Dynamic Domains," *J. Logic Programming*, to be published in 1996.
9. H.A. Simon, *The Sciences of the Artificial*, 2nd ed., MIT Press, Cambridge, Mass., 1981.
10. J. Mylopoulos et al., "Towards an Integrated Toolset for Program Understanding," *Proc. CASCON 94*, IBM Canada, Toronto, 1994, pp. 19–31.
11. D. Plexousakis, *On the Efficient Enforcement of Integrity Constraints in Temporal Deductive Knowledge Bases*, doctoral dissertation, Dept. of Computer Science, Univ. of Toronto, 1996.

Eric S.K. Yu is an assistant professor in information systems at the University of Toronto. He is also a principal investigator on a project to develop models for supporting business process analysis and redesign. His research interests include organization modeling, strategic reasoning, information systems analysis and design, software engineering, and applications of knowledge-based technologies. Yu received a PhD in computer science from the University of Toronto. He is a member of the ACM, the AAAI, and the IEEE Computer Society. Readers can contact Yu at the Dept. of Computer Science, Univ. of Toronto, 6 King's College Rd., Toronto, Ontario, Canada M5S 3H5; eric@cs.toronto.edu; http://www.cs.toronto.edu/~eric.

John Mylopoulos is a professor of computer science at the University of Toronto. His research interests include knowledge representation and conceptual modeling, covering languages, implementations, and applications. He is currently leading several research projects, including one on modeling and analyzing business processes. Mylopoulos received a PhD from Princeton University in electrical engineering in 1970. He is a fellow of the AAAI, a board member of the Very Large Databases Endowment, and a corecipient of the best paper award of the 1994 International Conference on Software Engineering.

Yves Lespérance is an assistant professor in computer science at York University, Toronto. His current work focuses on the development of logic-based tools for modeling and designing intelligent agents. These tools are now being used for applications in business-process reengineering, intelligent software agents, and robot control. Lespérance received his PhD in computer science from the University of Toronto in 1991. He is a member of the ACM, the AAAI, the Canadian Society for Computational Studies of Intelligence, and the French Association for Artificial Intelligence.