An Actor Dependency Model of Organizational Work – With Application to Business Process Reengineering

Eric S. K. Yu and John Mylopoulos

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

ABSTRACT

In developing information systems for use in an organization, one often needs to understand the reasons that underlie established work patterns and practices. Because organizational actors depend on each other for goals to be achieved, tasks to be performed, and resources to be furnished, reasons for work patterns can be revealed by examining the dependencies among actors. We present a model which characterizes a work organization in terms of the network of dependencies among organizational actors. Actor dependencies are taken to be intentional – they expand or restrict an actor's ability to pursue goals. The network of actor dependencies constitutes the intentional structure of the organization. We use examples from business process reengineering to motivate and illustrate the model.

KEYWORDS

Organization model, organization analysis and design, business process reengineering, workflow, requirements engineering.

INTRODUCTION

To build effective organizational computing systems, one needs to have a good understanding of the organizational environment in which the systems are intended to operate. Research in software engineering and information system development have increasingly recognized the need to *model* the environment (e.g., [3, 17, 2]). Models employing knowledge representation techniques have been developed to structure and manage the knowledge about environments (e.g., [14, 10, 25]). These models, however, have focused primarily on capturing the "whats" – what operations or activities are performed; what work products are produced – but not the "whys" – the intentions, motivations, and rationales that underlie the "whats".

When organizational participants explore and discuss with analysts and designers different ways in which computing systems can help improve their work processes, each design alternative embodies an implicit set of assumptions about work relationships. For instance, if an invoice is missing in accounts payable, who would be concerned, and why? Who is expected to track down a missing invoice? And why is there a need for invoices in the first place? These kinds of knowledge about a work environment is typically not found in models that are currently used to help develop organizational information systems. Yet these are important aspects of an organizational environment. A model which makes such *intentional* relationships among actors in an organization explicit can help clarify what a particular design entails, and is more likely to lead to a successful development effort and an effective organization.

Recent interest in business process improvement and innovation in the business and management communities (e.g., [12, 16, 11]) underscores the need to understand organizations at an intentional level. As business environments are changing rapidly, organizations must adopt new ways of working, and take advantage of the capabilities of new technologies [29, 20]. Too often, computers are simply used to automate existing ways of doing business. Hammer's concept of reengineering, in particular, emphasizes the need to keep asking Why? and What-if? questions. If one were to achieve the full potential of organizational computing technologies, one must challenge the reasons behind established work patterns and practices, and "obliterate" those that are there for reasons that are no longer valid [15].

In this paper, we propose a model of organizational work based on a notion of *actor dependency.* We view organizational participants as inherently autonomous actors who deploy resources (such as physical objects, information, and knowhow) to carry out actions in pursuit of goals. When actors participate in an organization, their otherwise autonomous behaviour become constrained by their interdependencies: organizational

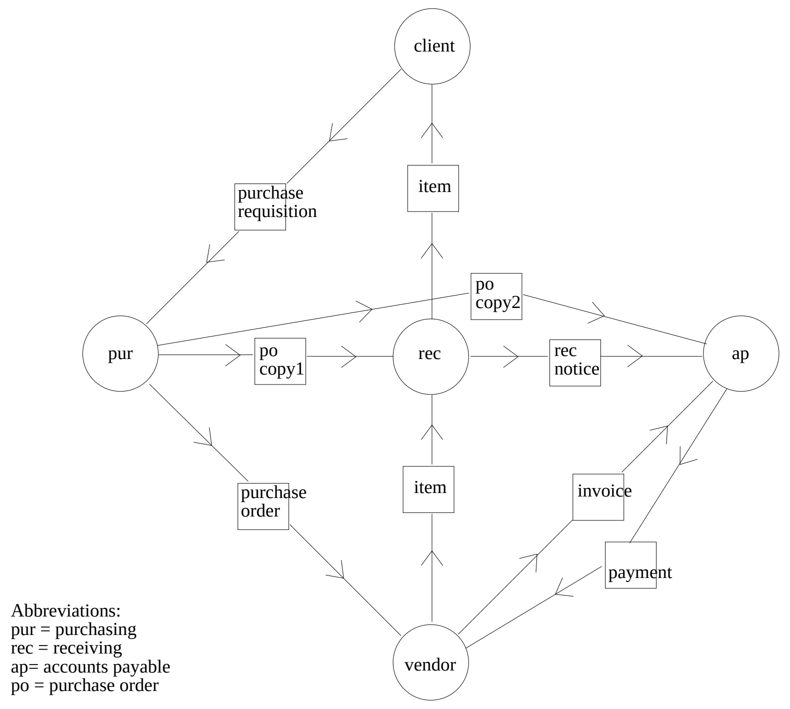


Figure 1: Workflow model of a goods acquisition process

actors depend on each other for goals to be achieved, activities to be performed, and resources to be furnished. A dependency extends an actor's capabilities, but it also makes the actor vulnerable. The network of dependencies among actors provides an intentional account of the relationships among actors, and the reasons that underlie the workflow structure. We call the network of intentional dependencies the *intentional structure* of the organization. The model is presented informally, using examples from business process reengineering.

Section 2 of this paper motivates the notion of intentional structure by comparing it with existing goal-based notions of organizational work. Section 3 presents the features of the Actor Dependency model using examples. In section 4 we use business process reengineering to illustrate how the Actor Dependency model could be applied. In section 5, we discuss the contribution of the model and the larger framework of which it is a part. We conclude in section 6 by summarizing this work and discussing future work.

THE INTENTIONAL STRUCTURE OF ORGANIZATIONAL WORK

A familiar and intuitive way to model a work organization is in terms of the flow of work products from one work unit (e.g., a department or a person) to another. For example, a typical workflow for acquiring goods in a business organization might be as represented in Figure 1. A more detailed model would show activities performed within each unit, with intermediate products as inputs and outputs of activities. Workflow models show the entities and activities involved in a work process, but not the reasons for their existence or relatedness.

A natural way to explain why certain entities or activities are needed in an organization is by seeing them as means to ends. For instance, entities and activities in the goods acquisition process can be traced to the goal that the client wants to have an item purchased. In terms of classical AI concepts, these objects can be linked to goals in a goal graph [28] (Figure 2). Each entity or activity could then be traced to the goals that it is a means for. Conversely, from a goal reduction viewpoint, goals are reduced to subgoals, and eventually to actions that can be performed. When fully reduced, they form a plan. The workflow model of Figure 1 could be interpreted as a plan to achieve the overall goal of purchasing an item.

Although this kind of goal graph does reveal, to some degree, the intentionality behind the work, it does not accurately reflect the way work actually gets done. Empirical research (e.g., [31, 13]) has indicated that a large part of organizational work has to do with addressing problems. Because of the open-endedness of organizational work situations, the exact kinds of problems that

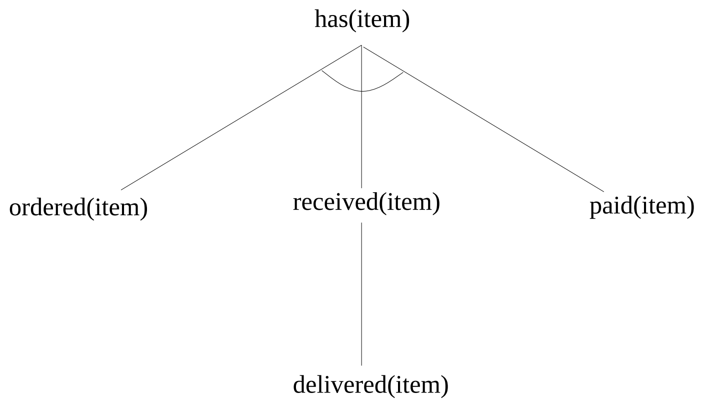


Figure 2: A Goal Graph for Goods Acquisition

will arise cannot be fully anticipated. The traditional notion of plans can only reflect a standard procedures view of the organization. In reality, actors interact with each other in order to accomplish work, and use their knowhow to deal with problems. The recognition of the problematic nature of organizational work has led researchers to develop technical systems that support the problem solving activities of organizational actors (e.g., [1, 34, 7, 9]).

A model aimed at helping organizational participants and stakeholders understand the implications of an organization design (e.g., a particular way of introducing computing technology) need not capture the actual problem solving knowledge (such as those required for building technical systems to support problem solving). Instead, what one would like to be able to tell, from the model of a particular design is: where in the work structure can problems be expected to arise; who is expected to deal with these problems when they arise; and whose interests would be hurt if the problems are not dealt with. It is therefore through the interaction of the actors involved, including the way problems are handled, that actors contribute towards the overall workings of the organization. The *network* of intentional relationships among actors constitutes the intentional structure of the organization.

AN ACTOR DEPENDENCY MODEL OF ORGANIZATIONAL WORK

An Actor 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 some "object" 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**. Figure 3 shows an Actor Dependency model for a goods acquisition process.

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* [24]. If the dependee fails to deliver the dependum, the depender would be adversely affected in its ability to achieve its goals.

Each link has four attributes: the direction of the dependency (i.e., which actor is the depender and which the dependee), the type of the dependency, the dependum object itself, and an indication of the "strength" of the dependency.

We distinguish among four types of dependencies, based on the type of the dependum. In world modelling, it has been found useful to distinguish among three basic ontological categories: entities, activities, and assertions [14]. Entities are used to model objects in the world. These can be physical or informational. Activities produce changes in the world. An assertion expresses a state or condition about the world. From these basic categories, we get three types of intentional dependencies: Resource-dependency, Task-dependency, and Goal-dependency. A fourth type, called a "Soft-goal-dependency", is a hybrid of goal- and task-dependency. The four types of dependencies also characterize how decisions fall on either side of the dependency, and which side will handle problems if and when they arise.

In a **Goal-dependency**, the depender depends on the dependee to bring about a certain state in the world. The dependee is free to, and is expected to, make whatever decisions are necessary to achieve the goal (the dependum). The depender does not care how the dependee goes about achieving the goal.

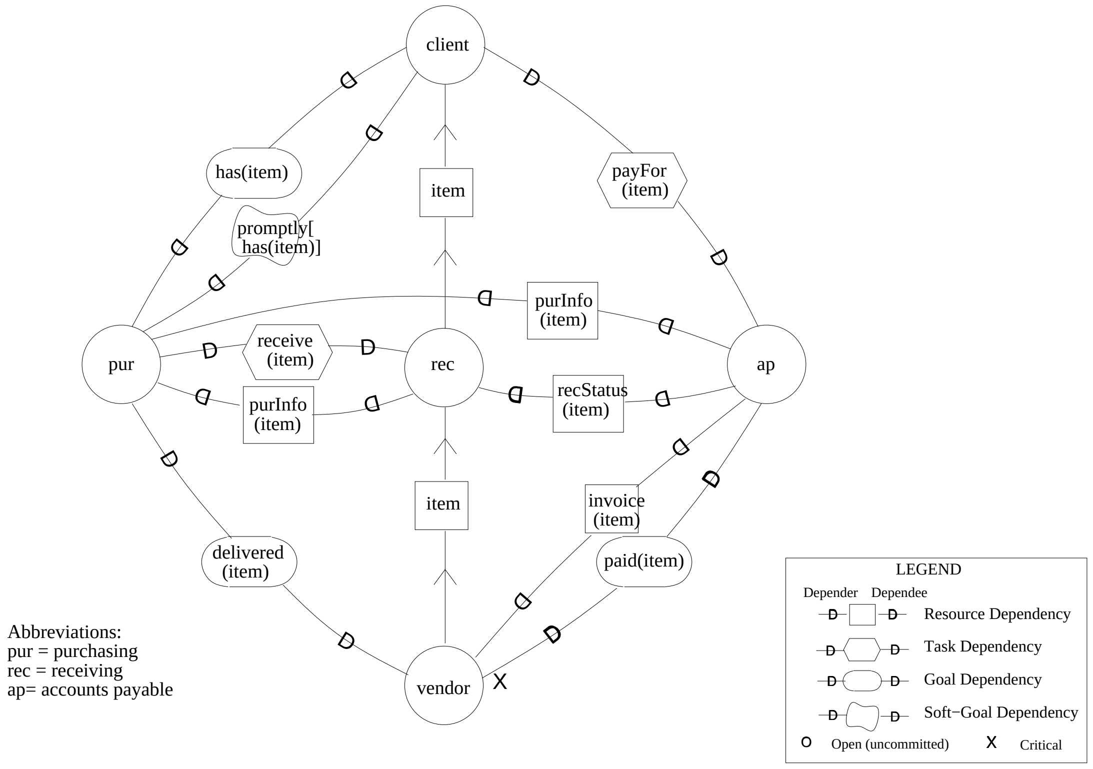


Figure 3: Actor Dependency model of a goods acquisition process

In the goods acquisition example, the client just wants to have the item, but does not care how the purchasing specialist obtains price quotes, or which supplier he orders from. Purchasing, in turn, just wants the vendor to have the item delivered, but does not care what mode of transportation is used, etc. A goal-dependency relationship allows a depender to have a goal achieved even when it does not have the knowhow and/or resources to achieve the goal. With a goal-dependency, the depender gains the ability to assume that the condition or state of the world will hold, but becomes vulnerable since the dependee may fail to bring about that condition.

In **Task-dependency**, the depender depends on the dependee to carry out an activity. A task-dependency specifies *how* the task is to be performed, but not *why*. The depender makes the decisions. The depender's goals are not given to the dependee.

In our example setting, Purchasing's dependency on Receiving is a task dependency because Purchasing relies on Receiving to follow procedures such as: accept only if the item was ordered. Similarly, the client wants Accounts Payable to pay only if the item was ordered and has been received. A task dependency is typically used when a desired product is not readily available (e.g., easily perishable), or when no tangible product is involved (e.g., go to some place). The depender has control over how the task is performed (e.g., how the product is made). By using a task-dependency, the depender is able to have a task performed without engaging in it personally, but is vulnerable since the dependee may fail to perform the task.

In **Resource-Dependency**, the depender depends on the dependee for the availability of an entity (physical or informational). Under resource-dependency, the issue of decisions does not come up. A resource is usually the finished product of some deliberation-action process. It is assumed that there are no open issues or decisions to be addressed. The production process is not viewed as problematic by the depender in a resource dependency. There is no decision to be made.

Accounting's dependencies for information from Purchasing, Receiving, and the vendor before it can issue payment are examples of resource dependencies. A resource can be physical or informational. By establishing a resource-dependency, the depender gains the ability to use this entity as a resource. At the same time, the depender becomes vulnerable if the entity turns out to be unavailable.

In a **Soft-Goal-Dependency**, a depender depends on

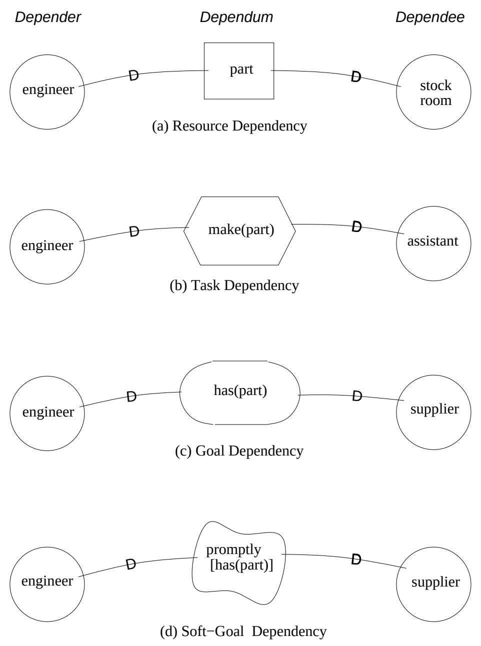


Figure 4: Four types of intentional dependencies

the dependee to perform some task that meets a soft-goal. A soft-goal is one whose meaning is specified in terms of the methods that are chosen in the course of pursuing the goal. As in a goal-dependency, a depender gains the ability of having the goal condition brought about, but becomes vulnerable in case the dependee fails to bring about that condition. The difference here is that the conditions to be attained are elaborated as the task is performed. Under soft-goal-dependency, the depender makes the final decision, but does with so with the benefit of the dependee's knowhow. For example, if the vendor wants to be paid *promptly*, the meaning of promptly needs to be further specified, perhaps in terms of specific steps that the payer would have to take. This interpretation of a soft-goal is based on a framework for dealing with non-functional requirements in software engineering [27, 5].

It is often possible to choose which type of dependency to establish for a given dependum object, by modifying the type of the object. The example in Figure 4 is an illustration. The scenario is one in which an electronics design engineer is in need of some parts for building a laboratory prototype. If the parts are obtainable from a stockroom, then the engineer's dependency on the stockroom for parts would be a resource dependency. The only kind of contingency is that the stockroom may run out of stock. If the electronics engineer asks an assistant to make a part for her by telling the assistant how to do it, this would be modelled as a task dependency. If the engineer wants a part from a supplier, but does not care how (e.g., how it is delivered), it would be a goal-dependency. If the engineer wants a part from a supplier requesting "prompt delivery", it would need further elaboration or qualification, e.g., deliver by courier. This would be a soft-goal dependency.

Given a non-intentional description of a "flow" of an electronic part from some actor to the engineer, one could not say, without further knowledge about the relationship, which dependency type would be appropriate for modelling the relationship. One would have to know, given certain types of uncertainties (problems that require decisions), which side would come up with the options, and which side would make the decision. As an organization evolves, it is quite possible for a relationship to shift from one type of dependency to another.

The model provides for three degrees of strength of dependencies: Open (uncommitted), Committed, and Critical. These apply independently on each side of a dependency. Graphically, we use an "O" for open, unmarked for committed, and "X" for critical.

On the depender side, an **open** dependency means that if the dependency fails, the depender would be affected to some extent, but not significantly. A **committed** dependency (unmarked) means that the depender would be hurt significantly if the dependency fails. For example, the depender might have invested considerable resources (e.g., time and effort) in a course of action, which could not be reversed without loss. A **critical** dependency would indicate that some goal of the depender could not be achieved if the dependency fails.

On the dependee side, an **open** dependency means that the dependee is able to achieve the goal, perform the task, or furnish the resource, but there is no commitment. A **committed** dependee will try its best to achieve the goal, perform the task, or furnish the resource. The analogous form of a critical dependency on the dependee side would suggest a need to guarantee success, which is usually hard to achieve.

AN APPLICATION TO BUSINESS PROCESS REENGINEERING

An Actor Dependency model captures the intentional relationships among actors in an organization. By following the chains of dependencies, one could explore the expanded possibilities that are open to an actor. From a vulnerability viewpoint, one could also use the model to determine how an actor could be affected adversely by its dependencies. We have argued that this type of intentional model is important for understanding an organization design. We now illustrate this with some examples from business process reengineering.

The concept of reengineering is advocated as one way to achieve dramatic improvements in organizational performance by fundamentally redesigning the work organization as new information technology is introduced [15, 11]. It is argued that information technology often is not able to deliver significant benefits because it is simply used to automate existing (and often outdated) business practices. Instead, one should question the appropriateness of the work process itself, with respect to today's business environment. This is summarized in the slogan: Don't automate, obliterate [15]. To do this, one needs to keep asking Why? and What-if? questions about existing work practices.

Workflow models cannot provide answers to Why? and What-if? questions because the intentional content is missing. Suppose one asks: Why do we need Purchasing in the goods acquisition process? The workflow model of Figure 1 could only indicate that purchasing is there to "process" purchase requisition forms. The model is equally unhelpful for answering what would happen if Purchasing were bypassed ("obliterated"). It offers little help in identifying alternatives to having a purchasing department.

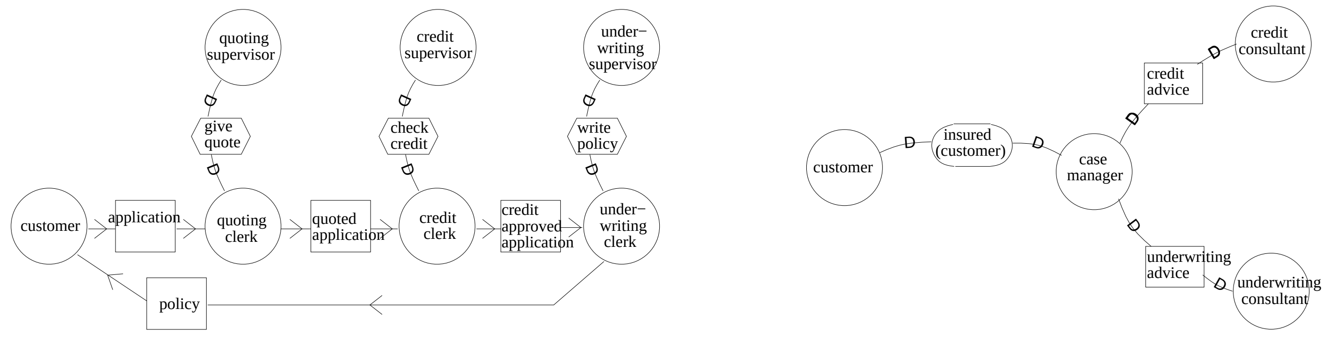
In contrast, from an Actor Dependency model, one could tell who depends on whom, and for what. In Figure 3, the client depends on purchasing for meeting the goal of having an item. Purchasing in turn depends on the vendor to meet the goal of having the item delivered, and on the Receiving department to perform the procedurally defined task of receiving the item. At each point in a chain of dependencies, one can infer from the model how the actor's goal-seeking behaviour may be enhanced or restricted, based on the type and strength of the dependencies that it has. It is this deeper knowledge that is necessary to help judge potential targets for obliteration.

To answer the question: "What if the Purchasing department is removed from the goods acquisition process?", we observe from the model that the client depends on Purchasing to achieve the goal of having an item, and is therefore vulnerable with respect to this same goal. The client's ability is enhanced through this dependency because the goal can be achieved even if the client does not have the knowhow or the resources to pursue it on his own. To bypass Purchasing, the client would have to acquire the knowhow and have the needed resources (e.g., time and effort) to doing purchasing on his own.

As reported in [15], one company reengineered its goods acquisition process by following this line of reasoning. The traditional purchasing process was full of paperwork, errors, and delays. For small purchases, it was not uncommon for the purchasing process to cost more than the item. The company recognized that expert systems technology could be used to provide the knowhow and resource support for simple purchases. Now, except for large or strategic orders, company employees would order most items directly from pre-approved vendors through the system without the help of human purchasing specialists.

The Ford Motor Company took a different approach. It reengineered its goods acquisition process by eliminating invoices. Instead of paying when an invoice is received, Ford now pays when it gets the goods. A large part of accounts payable work consists in reconciling disagreements among purchase orders, receiving documents, and invoices. Information technology could have been deployed to help investigate invoicing errors, and to automate document flow, but that would not have provided the radical improvement Ford had aimed for. By redesigning the business process, invoices, and hence invoicing errors, were eliminated altogether (in the spirit of the zero-defect approach to process quality [8]). The remaining reconciliation (between purchase orders and receiving reports), which is much simpler, could now be handled mostly by computer, which also generates the payment cheque.

In this case, the intentional dependency between vendor and Accounts Payable – that the vendor wants to have the item paid for – is unchanged before and after reengineering. But realizing that the invoice is only a



(a) before (b) after

Figure 5: "Organize around outcomes, not tasks"

means to this end, Ford was able to find an alternate, and much more effective, means to the same end, and took the radical step of obliterating the age-old practice of invoice processing. The Actor Dependency model of Figure 3 would have made this reasoning more perspicuous. With conventional methods that focus on workflow analysis (Figure 1) and no explicit support for answering Why? and What-if? questions, one is more likely to end up with the less effective, "automation" approach.

A number of rule-of-thumb principles have been proposed to guide reengineering efforts (e.g. see [15, 32]). We use two of these principles to further illustrate how the Actor Dependency model can help bring out the distinctive features of different organizational configurations by making their intentional structures explicit.

"Organize around outcomes, not tasks".
One common way to organize work is to group similar tasks into units with a specialized function, such as order entry, credit checking, assembly, or shipping. The problem with this structure is that while workflow passes from unit to unit, no one is responsible for the overall process from end to end. While each person is accountable to a supervisor in his/her own functional unit, problems that arise in between units tend to fall through the cracks (e.g., files misplaced, delayed, or lost in transit). One insurance company (Mutual Benefit Life) reengineered its policy application process into a configuration in which a single person acts as a case manager and handles a customer's application from beginning to end. Using computer support, the case manager performs all the tasks associated with the application. For difficult cases, she would seek help from specialist consultants, but only for advice. The decisions remain with the case manager.

The Actor Dependency model of Figure 5a reflects the traditional functional organization. Each processing unit (the clerks) is a dependee in a task dependency, but are only related to each other by non-intentional workflow. In the reengineered configuration (Figure 5b), there is a single goal dependency from the customer to the case manager. The case manager depends on the consultants' advice as a resource, since the case manager is the one who makes the decisions and takes action.

"Put the decision point where the work is performed, and build control into the process".
In hierarchically structured organizations, decisions are made in the higher levels, while the ensuing tasks are executed by the lower levels. This type of structure is often plagued with problems of delay, error, and miscommunication. Reengineering recommends to allow the person performing the work to make the decision. Computer networks and shared databases can be used to enable one to access information and knowhow so that one person can encompass a much broader scope of work. The case manager in the insurance company exemplifies this principle. She can best decide how to meet customer needs because she is closer to the customer.

Using an Actor Dependency model, the traditional hierarchical delegation chain may be modelled as a string of task dependencies (Figure 6a). Under the reengineered configuration, the relationship between the case manager and her superior is modelled as a goal dependency (Figure 6b). She has the freedom to make decisions regarding how to meet the goal. Control is built-in because it is the outcome that matters to the depender, not the detailed activities of the dependee.

These examples from business process reengineering help illustrate the need to understand a work organization at an intentional level. Without the deeper knowledge

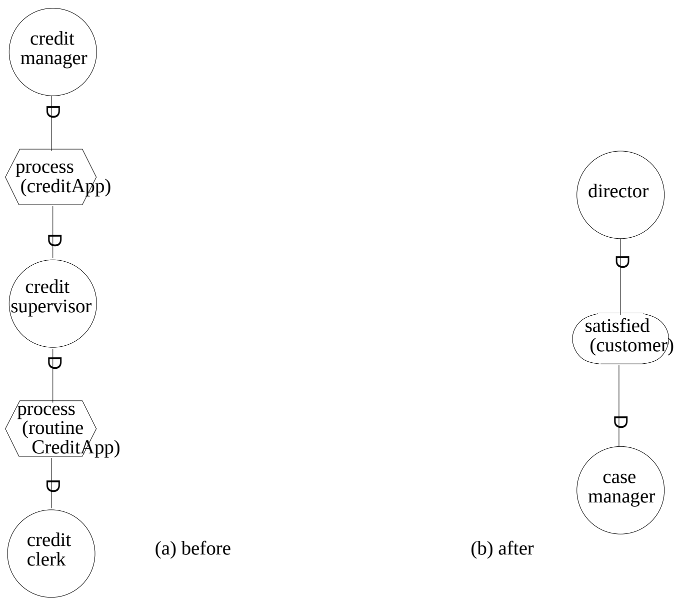


Figure 6: "Put the decision point where the work is performed, and build control into the process."

about intentional structure, one could not easily break away from current practice to a new conceptualization of the work process.

DISCUSSION

We have presented the basic features of the Actor Dependency model. We now discuss its use in the broader context of organization redesign and information system development.

The Actor Dependency model is one of two main components in a framework aimed at supporting requirements engineering for organizational information systems [37]. The purpose of the Actor Dependency model is to provide an appropriate representation of an organizational configuration (of human and computer elements). The other main component, called the Issue Argumentation model, provides a representation of the issues and concerns that stakeholders may have about current and various proposed organizational configurations ("designs"). In considering design alternatives, many issues can arise. The business process improvement and reengineering literature focuses on issues such as turnaround time, manpower savings, and quality of service to customers. For successful information system and organizational change, other issues such as power, cultural values, and conflict will also need to be considered (e.g., [19, 21]). The Issue Argumentation model uses an argumentation structure to manage and support stakeholders' reasoning about various designs, following work in design rationale [22] and in non-functional requirements in software engineering [5, 27].

The Actor Dependency model serves as the subject matter for the arguments in the Issue Argumentation model. For example, a stakeholder might be concerned that a redesign putting him in a goal-dependency relationship with another member (say, instead of a task-dependency in the current work arrangement) would reduce his power and control in the work group. The other member may welcome the increased responsibility, but have the concern that she may not be able to meet the commitment without additional resources. An organization model which does not capture the intentional relationships between actors – such as conventional workflow models – would not be able to differentiate alternate designs to a degree that is required for stakeholders to express their concerns. An illustration of how the Actor Dependency model is used in conjunction with the Issue Argumentation model has been presented in [36]. A third component of the framework differentiates actors into agents, roles, and positions to deal with more complex organizational relationships.

The organization modelling framework of [37], of which the Actor Dependency model is part, follows a conceptual modelling approach to software engineering and information system development, which emphasizes the need to represent and utilize pertinent knowledge to support each phase of development and on-going evolution [26]. The organization modelling framework aims to add to this line of research ([14, 25, 18]) by elaborating on the link between organization redesign and technical system development.

In accordance with the conceptual modelling approach, we have endeavoured to seek a formal grounding for the concepts of the model. Intentional models of agents have been developed in AI (e.g., [6, 33, 23]) using modal operators for belief, goal, ability, and commitment. We have adapted some of these concepts for characterizing actor dependencies. A preliminary set of axioms for the model were proposed in [35].

A formal characterization of the model will serve to clarify the semantics of the model features, and will help human users of the model resolve ambiguities in interpretation. For computational support, general inference procedures for the formalism are likely to be intractable. For practical applications, we intend to identify specialized tractable algorithms for computing selected properties that are of particular interest. For instance, various types of conflict resulting from a confluence of dependencies on an actor would be of interest. There may be conflicting goals, opposing tasks, and contention over resources. Also of interest are special patterns of dependency networks, such as reciprocal dependencies or loops [4]. A reciprocal dependency can be viewed positively, as in an exchange relationship. It can also be viewed as a method of control to enforce commitment, through countervailing power, since reciprocal dependency implies reciprocal vulnerability. In a different context, a dependency loop could be indicative of a conflict-of-interest situation: for example, if a person in charge of a budget has a dependency on a beneficiary of the budget, then the risk of collusion for kickbacks exists. These kinds of properties, once recognized in the Actor Dependency model, can be used as support for arguments in the Issue Argumentation model for or against particular redesign proposals.

Actor Dependency modelling is intended to complement, not to replace, existing types of organization modelling. Entity Relationship-based organization models (e.g. [30]) or more elaborate semantic data models which focus on passive and static relationships will continue to be useful for characterizing the structure of information and data. SADT or other techniques that focus on activity and process will still be needed for representing the "surface structure" of the routine dynamics of an organization. It is when one is facing organizational change, as one digs below the surface into the whys and wherefores, that the Actor Dependency model becomes helpful for representing and reasoning with this deeper knowledge. As argued in this paper, these are the situations faced in IS requirement engineering and business process reengineering.

Researchers in the office/organizational information systems (OIS) area have long adopted this more dynamic, open-ended, problem-solving view of organizational work, and have developed more sophisticated and flexible information systems by embedding knowledge representation and reasoning into the system (e.g. [1, 34, 7]). The Actor Dependency model, on the other hand, while based on similar assumptions about the nature of organizational work, is aimed at helping organizational actors identify the types of information technology components that might be most appropriate for supporting their work, and, in general, would not be embedded in the application system. The concept of actor dependency modelling, however, need not be limited to the domain of information system requirements. The need to identify and assess opportunities and vulnerabilities among interdependent actors arises in many contexts, such as negotiation and strategic decision making. When the Actor Dependency model, with its associated framework is implemented as a generic decision support tool (as opposed to a requirements engineering tool), it is itself a candidate technology component for incorporation into the organizational fabric for consideration during the redesign of work.

CONCLUSION

We have argued that, in order to adequately characterize a work organization when considering potential work support technologies, it is important to identify the intentional relationships among actors. The Actor Dependency model presented in this paper is an attempt to provide a way to model these relationships. We have illustrated how the model could be used in the context of organization redesign, using example settings from the literature on business process reengineering. The adequacy of the particular set of features proposed in the model will need to be tested in practice.

To incorporate the model into a knowledge-based, software engineering environment, a number of steps will need to be taken. The semantics of the model will need to be characterized formally. Algorithms for assisting with reasoning about the model will need to be identified. Knowledge structuring mechanisms such as classification, generalization, aggregation, and time (such as those offered in Telos [25]) will also be necessary.

A model is but one ingredient in the overall process of understanding and redesigning an organization and its technological support. The methodological implications of the proposed model remain open issues. For example, will the model make elicitation more difficult – because it probes deeper to get at rationales; or easier – because asking "why" helps structure the inquiry? Will stakeholders be reluctant to divulge their dependency relationships, or will they participate actively in order to secure their interests? How can a model of intentional structures be maintained and kept up to date? These and many other issues will need to be investigated before the model can be pragmatically applied.

Business environments are undergoing rapid change. Recent advances in organizational computing technology offer many opportunities for greater organizational effectiveness. It is hoped that the proposed model will contribute towards a conceptual framework, and eventually tools and methodologies, with which organizations can systematically examine and assess the different ways in which modern organizational computing technologies can help improve organizational effectiveness.

ACKNOWLEDGMENTS

The authors would like to acknowledge helpful comments from Lawrence Chung, Brian Nixon, Carson Woo, and anonymous referees. The first author would also like to thank Prof. Fred Lochovsky for helping him gain a deeper understanding of organizational computing through many discussions.

References

[1] G. R. Barber, *Office Semantics*, Ph.D. Dissertation, Dept. of Elec. Eng. and Comp. Sci., M.I.T., 1982.

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

[3] J. A. Bubenko, Information Modeling in the Context of System Development, *Proc. IFIP,* pp. 395-411, 1980.

[4] C. Castelfranchi, M. Miceli, and A. Cesta, Dependence Relations Among Autonomous Agents, *Decentralized A.I. - 3 (Proc. 3rd European Workshop on Modeling Autonomous Agents in a Multi-Agent World)*, Elsevier, 1992.

[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, submitted for approval.

[6] P. R. Cohen and H. J. Levesque, Intention is Choice with Commitment, *Artif. Intell.,* 42 (3), 1990.

[7] W. B. Croft and L. S. Lefkowitz, A Goal-Based Representation of Office Work, *Office Knowledge: Representation, Management, and Utilization,* W. Lamersdorf (ed.), Elsevier, 1988, pp. 99-124.

[8] P. R. Crosby, *Quality is Free*, MacGraw-Hill, 1979.

[9] P. de Jong, *Ubik: A Framework for the Development of Distributed Organizations*, Ph.D. Dissertation, Dept. of Elec. Eng. and Comp. Sci., M.I.T., 1989.

[10] E. Dubois, J. Hagelstein, E. Lahou, F. Ponsaert and A. Rifaut, A Knowledge Representation Language for Requirements Engineering, *Proc. IEEE*, 74 (10), pp. 1431 -1444, Oct. 1986.

[11] T. H. Davenport, *Process Innovation: Reengineering Work Through Information Technology,* Harvard Business School Press, Boston, Mass., 1993.

[12] The Search for the Organization of Tomorrow, *Fortune Magazine*, May 18, 1992, pp. 93-98.

[13] L. Gasser, The Integration of Computing and Routine Work, *Trans. Office Info. Sys.*, vol. 4, no. 3, July 1986, pp. 205-225.

[14] S. J. Greenspan, *Requirements Modelling: A Knowledge Representation Approach to Software Requirements Definition,* Ph. D. Thesis, Dept. of Comp. Sci., Univ. of Toronto, 1984.

[15] M. Hammer, Reengineering Work: Don't Automate, Obliterate, *Harvard Business Review*, July-August 1990, pp. 104-112.

[16] H. J. Harrington, *Business Process Improvement: the breakthrough strategy for total quality, productivity, and competitiveness*, MacGraw-Hill, 1991.

[17] M. Jackson, *System Development*, Prentice-Hall, 1983.

[18] M. Jarke, J. Mylopoulos, J. W. Schmidt, Y. Vassiliou, DAIDA: An Environment for Evolving Information Systems, *ACM Trans. Information Systems*, vol. 10, no. 1, Jan 1992, pp. 1-50.

[19] P. Keen, Information Systems and Organizational Change, *Comm. of ACM*, vol. 24, no. 1, January 1981, pp. 24-33.

[20] P. Keen, *Shaping the Future: Business Design Through Information Technology*, Harvard Business School Press, Boston, Mass., 1991.

[21] R. Kling, Defining the Boundaries of Computing Across Complex Organizations, *Critical Issues in Information Systems Research*, R. J. Boland Jr., and R. A. Hirschheim, eds., Wiley, 1987.

[22] J. Lee and K.-Y. Lai, What's In Design Rationale? *Human-Computer Interaction* vol. 6, no. 3, 1991, pp. 251-280.

[23] Y. Lesperance, *A Formal Theory of Indexical Knowledge and Action,* Ph.D. Thesis, Univ. of Toronto, also, Tech. Rept. CSRI-248, Comp. Sys. Res. Inst., Univ. of Toronto, Feb. 1991.

[24] T. W. Malone, Modeling Coordination in Organizations and Markets, *Management Science*, vol. 33, 1987, pp. 1317-1332.

[25] J. Mylopoulos, A. Borgida, M. Jarke, M. Koubarakis, Telos: Representing Knowledge about Information Systems, *ACM Trans. Info. Sys.*, 8 (4), 1991.

[26] J. Mylopoulos, Representing Knowledge About Information Systems, *Intl. Workshop on Development of Intelligent Information Systems*, Niagara-on-the-Lake, Ontario, Canada, April 21-23, 1991, pp. 94-96.

[27] J. Mylopoulos, L. Chung, B. Nixon, Representing and Using Non-Functional Requirements: A Process-Oriented Approach, *IEEE Trans. Soft. Eng.*, 18 (6), June 1992.

[28] N. Nilsson, *Principles of Artificial Intelligence*, Tioga Press, 1980.

[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] A.-W. Scheer, *Enterprise-Wide Data Modelling: Information Systems in Industry*, Springer-Verlag, 1989.

[31] L. Suchman, Office Procedures as Practical Action: Models of Work and System Design, *ACM Trans. Office Information Systems*, vol. 1, no. 4, October 1983, pp. 320-328.

[32] D. Tapscott, A. Caston, *Paradigm Shift – The New Promise of Information Technology*, McGraw Hill, 1993.

[33] B. Thomas, Y. Shoham, A. Schwartz, and S. Kraus, Preliminary Thoughts on an Agent Description Language, *Intl. J. Intell. Sys.*, Vol. 6, 1991, pp. 498-508.

[34] C. Woo, *An Object-Oriented Model for Supporting Office Work*, Ph.D. Thesis, Dept. of Comp. Sci., Univ. of Toronto, 1988.

[35] E. Yu, Modelling Organizations for Information Systems Requirements Engineering, *Proceedings of First IEEE Symposium on Requirements Engineering*, San Diego, Calif., 1993, pp. 34-41.

[36] E. Yu, An Organization Modelling Framework for Multi-Perspective Information System Design, in *Requirements Engineering 1993: Selected Papers*, J. Mylopoulos et al., eds., Univ. of Toronto Dept. of Comp. Sci. Tech. Rpt. DKBS-TR-93-2, July 1993.

[37] E. Yu, *An Organization Modelling Framework for Information Systems Requirements Engineering*, Ph.D. Thesis, Dept. of Computer Science, Univ. of Toronto, forthcoming.