An Organization Modelling Framework for Multi-Perspective Information System Design

Eric S. K. Yu
Dept. of Computer Science, University of Toronto
Toronto, Ontario, Canada M5S 1A4

Abstract

The design of information systems have traditionally been carried out primarily from a technical perspective. Recently, there has been increasing recognition that building truly effective systems requires taking into account a much broader set of perspectives, representing the viewpoints of various stakeholders, addressing their economic, political, institutional and other concerns. In order to support organizational analysis as an integral part of the overall information system development process, there is need for a practical and systematic framework to utilize insights from these broader perspectives to guide design. This paper outlines a conceptual modelling framework which attempts to provide a systematic way for organizing and using knowledge from multiple organizational analysis perspectives. The framework is made up of three types of models. The first represents an organizational configuration as a network of interdependencies among actors. The second models the make-up of social actors in terms of roles, positions, and concrete agents. The third uses a network of arguments to express the concerns that actors have about the current and potential organizational configurations from various perspectives. The framework is illustrated with empirical examples taken from the literature.

Categories and Subject Descriptors: K.4.3 [**Computers and society**]: Organizational impacts; D.2.1 [**Software engineering**]: Requirements/Specifications – *languages, methodologies*; D.2.10 [**Software Engineering**]: Design – *methodologies, representation*; H.5.3 [**Information Systems**]: Group and organizational interfaces – *organizational design, theory and models*; K.6.1 [**Management of Computing and Information Systems**]: Project and People Management – *systems development.*

Genenral Terms: Design, Human Factors, Management, Languages.

Additional Key Words and Phrases: conceptual modelling, knowledge representation, multiple perspectives, organizational analysis.

1 Introduction

The design of information systems (ISs) have traditionally been carried out primarily from a technical perspective. This narrow focus has been cited as a major source of problems when ISs are used in organizational settings, and various authors have argued that building truly effective systems requires taking into account a much broader set of issues (e.g. [17] [6]).

Information systems operate in rich organizational contexts. Suppose a new information system is to be developed for use in a hospital ward. Functionality and cost would likely be the most prominent issues discussed among system sponsors (e.g. hospital administrators) and developers. However, the eventual overall effectiveness of the system as embedded in its organizational context is determined by a much wider network of social actors, or "stakeholders". Stakeholders may or may not be direct users of the system, but can nevertheless have important impacts on the eventual outcome of system introduction. In a hospital ward IS, the set of stakeholders may include nurses and physicians of various specialties and skill levels, supervisory staff such as head nurses and medical directors, and, of course, patients, and even patients' relatives. Also included are support staff for the IS, as well as users and stakeholders of other ISs that are interconnected with the proposed IS, or share resources with it.

External organizations such as funding agencies, insurance organizations, educational and training institutions, and regulatory agencies can all have impact on or be impacted by new technology in the hospital ward.

The stakeholders of an IS may have a wide ranging set of concerns about an IS and how it might alter the work environment. For example, along with the introduction of a new IS, there may be changes in work roles [9] and in skill levels and opportunities for advancement; jobs may be enriched or extra burdens may be imposed [13]; there may be changes to the power structure [3] (e.g. when there is competition for resources, who is more likely to win or lose); there may be increased social pressure and discipline [17], or of potential conflict; individual freedom or privacy might be curtailed; and the possibility for negotiating future work arrangements may be enhanced or restricted. This array of issues is juxtaposed on top of the more explicit ones such as quality of care, and cost savings or profitability in the hospital.

Because stakeholders have so much at stake, they will in turn attempt to shape the way the system is used, misused, or disused, in the organization [16]. The overall effectiveness of an information system therefore could not be adequately understood without a deeper analysis of the organizational environment within which the technical system is embedded. To this end, researchers have borrowed broadly from the conceptual frameworks and theories developed in the social sciences [18].

Multiple Perspectives in Organizational Analysis. One feature of knowledge about the social world is that there can be many interpretations of a situation, depending on one's perspective. Each perspective highlights certain issues. For this reason, it is often helpful to employ more than one perspective in understanding an organization. A multi-perspective approach has been advocated in organizational analysis [24] [30] as well as in social analysis of computing [17][23]. [17] illustrates how four perspectives could be used together to analyse a number of computing situations in organizational environments – a rational perspective which emphasizes the formal goals of an organization such as profitability and productivity, a structural perspective which highlights uncertainty and deviations from standard procedures, an interactionist perspective which focuses on social meanings such as professional status or the symbolic value of state-of-the-art technology, and a political perspective which looks for coalitions of interests in search for autonomy and control. Together, these perspectives offer a rich picture of the cases studied.

From Organizational Analysis to System Design. System development is often taken to include requirements analysis, design, and implementation. As an important part of the requirements process, organizational participants explore different system options that developers might be able to offer, and endeavour to select options that appear to best address their needs and concerns.

Given that organizational analysis can help interpret and explain, after the fact, the effects of ISs on organizations, one would hope that the same insights could be used to assist developers and organizational participants to make better-informed choices about system alternatives. This is indeed an expressed objective in much of the social analysis of computing literature. Equipped with empirical skills and the theoretical foundations of various analytical perspectives, an organizational analyst would be able to probe deeper into the needs and concerns of participants. These stakeholders would then be better able to discuss and negotiate among themselves, and to search for appropriate technical options from developers.

While taking a wide range of issues from many perspectives into account may improve the likelihood of an effective system solution, the complexity of the task presents a considerable challenge. Unlike in after-the-fact analysis, where there is essentially one historical course of events to explain, the requirements process typically involves a large space of options generated from many layers of decisions. A multi-perspective analysis would require an assessment of the impact of an option on each issue from each analytical perspective, from the viewpoint of each stakeholder. Due to the large quantities of information involved in this process, important items or linkages can easily be overlooked. A systematic approach would appear to be warranted.

A Conceptual Modelling Approach. We follow an approach in Software Engineering and Information System Development based on *conceptual modelling* [5]. The premise is that, in order to organize and manage the large amounts of knowledge involved at each phase in software engineering, appropriate schemes for representing and using the concepts of the domain are essential [4]. Different conceptual models are appropriate for different phases in system development [25] [15]. Each phase deals with a different type of subject matter (e.g., humans interacting in a social world, versus machines operating on data structures), has different built-in assumptions, and thus requires a different type of model.

However, even though different types of models are required for different domains, there needs to be sufficient underlying commonality to allow a "mapping" across domains. The underlying mechanisms for organizing knowledge can be made common across these models. For example, by having a uniform representation of *concepts* in terms of attributes and attribute categories, and relating concepts along classification, generalization, aggregation and temporal dimensions can be made common across domains. This allows complex relationships among elements across domains to be managed, without the need to enforce a uniform ontology and common assumptions across these models.

In following this approach, one could have models of organization that are designed to meet the needs of organization modelling, as well as models of technical systems that focus on technical issues. One could then use the underlying conceptual modelling mechanisms to link items in one type of model to items in the other type of model, without imposing the world-view of one on the other.

Concepts for modelling technical systems have long been the subject of study in computing science. But what are the appropriate concepts for modelling organizations?

Some Requirements for an Organization Modelling Framework. Many authors have pointed to the pitfalls of mechanistic views of organizations [32] [17] [14] [36] [11]. Formal procedures do not describe how work is actually carried out in organizations [32]. They overlook the problem solving activities that are involved in getting work done. Organizational work is often open-ended [14], because social actors have limited knowledge about, and limited control over their environments. Social actors are best seen as being situated within a network or web of social relationships [17] [20]. Conflict is ubiquitous in a social environment – both within and among social actors – due, for example, to resource limitations or differences in values and interests [20] [11].

To accommodate these characteristics of the organizational domain, a modelling framework needs to adopt a view of a social actor that recognizes the actor's inherent autonomy within a set of social constraints. Since social actors are multi-faceted, each attempt to model an actor is from a particular perspective and will only produce a partial view. Analysis of social situations cannot be based on *a priori* boundaries. The scope of analysis needs to be determined during the analysis, by applying various criteria to delineate the networks of social linkages. Further, the problems, issues, alternatives, and criteria from multiple perspectives need to be brought together within one framework for comparison, understanding, and assessment, without necessarily attempting to resolve the conflicts or incompatibilities among them.

The Proposed Framework. The framework proposed in this paper is an attempt to provide a systematic way to organize the knowledge from multi-perspective organizational analysis so that they may be linked systematically to system design options and decisions.

The framework is made up of three types of models. The first, called an Actor Dependency (AD) model, represents an organizational configuration as a network of interdependencies among actors. The second, called an Agents-Roles-Positions (ARP) model, models the make-up of social actors in terms of roles, positions, and concrete agents. The third, called an Issue Argumentation (IA) model, uses a network of arguments to express the concerns that actors have about the current and potential organizational configurations from various perspectives.

In this paper, we focus on an informal presentation of the modelling concepts of the framework. These concepts are expressed through a graphical notation, and are explained by way of examples. A more formal characterization will be necessary for computational support within the context of software engineering tools and environments. A preliminary formalization of intentional dependencies was presented in [37].

We draw on theories of organization for organizational modelling concepts such as dependency [34] [28] and roles [35]. For the Issue Argumentation Model, we adopt and adapt concepts from a framework developed for dealing with non-functional requirements in software engineering [8] [27].

We use examples from a hospital setting to introduce concepts of the framework. These are loosely based on [2]. For illustrating multiple organizational analysis perspectives, we use an example from [17] about the use of a materials control information system in a manufacturing environment. To illustrate how the various parts of the framework work together, we use as example an ethnographic study from the literature about the introduction of a user-interface design tool called Trillium into a design team organization [3].

Limits to a Modelling Approach. A model highlights certain aspects of the real-world, inevitably omitting others. It offers a specialized vocabulary for expressing and reasoning about a selected domain, but has built-in assumptions and blindspots. In using models to describe organizations, we need to be mindful of their limitations. Some social analytical perspectives especially cautions against premature use of *a priori* categories in interpreting organizational situations.

The proposed framework formalizes selected concepts so as to be able to make some distinctions at an appropriate granularity, while leaving others for interpretation at an informal level. For example, we formalize certain notions of dependency, but leave concepts such as power, prestige, and autonomy as informal concepts to be argued about in a qualitative reasoning framework.

Organization of this paper. The rest of this paper is organized as follows. In section 2, we motivate and preview the key features of each of the three component models of the framework. In section 3, we present the details of the features in each of the models, using examples to illustrate the concepts. In section 4, we show how the various parts of the framework work together by applying the framework to a describe some experiences in introducing the Trillum design tool that was reported in the literature. In section 5, we discuss related work in both the social and the technical research areas. In section 6, we conclude by summarizing this research and points to future work.

2 Framework Overview

The proposed framework is made up of three types of models. An *Actor Dependency* model is used to describe organizational work patterns in terms of the network of dependencies among actors. An *Agents–Roles-Positions* model differentiates social actors into roles, positions and concrete agents. Finally, an *Issue Argumentation* model is used to express the issues and concerns that actors have. These three models are used together to express the concerns that agents have about work arrangements alternatives that may result from information system design options.

The Actor Dependency Model. An Agent Dependency (AD) model views an organization as a network of *intentional* dependencies among actors in a social environment. It is used to describe how work is organized among the members of an organization.

One way to model an organization is to list the activities of each member; and to draw work-flow diagrams showing the normal flow of events. Such models only present the "formal" (official) division of work. They tend to be highly abstract and idealized, and do not accurately describe how work actually gets done in the organization [32]. In particular, they do not take into account the often problematic nature of work. One could not tell from such a model what is likely to go wrong, and what might happen if and when something does go wrong.

In order to have models that can reflect the nature of organizations more accurately, we need to recognize that actors are ultimately autonomous beings. Not only are they able to solve or cope with problems in may different ways, they also have their own goals and aspirations. They explore possibilities and alternatives, using their experience and knowledge to guide action. Actors are *intentional* in that they are not just reacting to the factual world, but have conceptions of alternate realities, and pursue actions to bring about these alternate visions.

Many of the desires and aspirations of an actor cannot be fulfilled by relying on its own actions. In a social world, each actor *depends* on other actors for achieving some part of what it wants, and are in turn depended on by others. Actors have wants that are met by others' abilities, run tasks that are performed by others, and deploy resources that are furnished by others. These dependencies form a complex and intricate network of intentional relationships among actors that might be called the *intentional structure* of the organization.

When social actors participate in an organization, they are still free to act in many regards, but their freedom is qualified by the dependencies that they have on others and that others have on them. By using dependencies among actors as the main vehicle for modelling an organization, we would acknowledge the fundamental autonomy of actors. On the other hand, since a great deal of what a social actor does *is* constrained by its relationships with other social actors, the structure and content of their dependencies can reveal considerable detail for us to appreciate and express the differences among alternate ways of organizing.

The AD model distinguishes among four types of dependencies – **Resource-dependency**, **Task-dependency**, **Goal-dependency**, and **Soft-goal-dependency**. These indicate the nature of the intentional relationship between the two sides, including which side would deal with what contingencies. By following the chains of dependencies in an AD model, one could identify the possibilities that are open to an actor, as well as the threats and vulnerabilities that it faces. To indicate the strength of a dependency, the model distinguishes among **open** (uncommitted), **committed**, and **critical** dependencies.

The Agents-Roles-Positions Model. The Agents-Roles-Positions (ARP) model views social actors as being discernible into constituent "units" – **roles** that focus on specialized domains, **positions** that bring together related roles, and **agents**, which are actors with concrete, physical embodiment.

Instead of ascribing dependencies to a social actor as an undifferentiated unity, one could ascribe them to these sub-units, thus categorizing and structuring the large number of dependencies that a social actor would typically have. Each of these sub-units presents a partial view into the potential behaviour of the social actor. The behaviour of the social actor is characterized by the combined effect of all these views.

Partial views can be useful because they give a first-cut approximation to the behaviour of the social actor from a particular, more specific perspective. On the other hand, this simplified view can often be misleading because other aspects of the actor are not taken into account. In practice, it is usually impossible to get a complete picture of any social actor, from all possible angles. Treating a complex social actor as consisting of sub-units thus facilitates identifying and selecting those aspects that are most relevant to a particular situation.

An ARP model uses **association** links to model which positions cover what roles, which agents occupy which positions, and which roles are played by which agents. A simpler, aggregated notion of an **interdependency** link is used to hide the details of the dependency relationships, which can be revealed by reference to a corresponding Agent Dependency Model.

The Issue Argumentation Model. An Issue Argumentation (IA) model is a network of assertions (issues) linked by reasons (arguments). It is a reasoning structure which shows the relationships among a set of issues. The IA model is used to help organize the issues of concern to organizational participants regarding their work arrangements. By having an explicit model of the network of reasons, one can see why an actor might prefer one organizational arrangement over another, and what criteria these actors use to evaluate alternative arrangements.

The issues and concerns of actors may embody concepts from many different perspectives, reflecting their varied experiences. An organization analyst may choose to further analyse and interpret these according to various theoretical perspectives. Concepts as diverse as efficiency, uncertainty, prestige, and power may all be used to characterize one situation. The network of arguments must be able to accommodate the conflicts and incompatibilities that may be implicit or explicit in the full range of issues considered.

The IA model adopts a qualitative argumentation approach. It does not attempt to formalize the meanings of the issues. Instead it helps to clarify and manage the relationships among issues. Issues are treated as **contributing factors** toward other issues. Contributions can be **positive** or **negative**. A notion of **satisficing** is used to capture actors' judgements about whether a contribution is strong enough to address an issue. This is useful for registering actors' choices or decisions that may eventually lead to action.

In the combined framework, the issues and concerns expressed in an Issue Argumentation Model are used to argue about work arrangement *alternatives* that may arise from various information system design options. These alternatives are expressed in terms of dependencies and associations in Actor Dependency and Agents-Roles-Positions models.

3 Framework Details

In this section, we present each of the three types of models in the framework in detail.

3.1 The Actor Dependency Model

An Agent Dependency (AD) model consists of a set of **nodes** and **links**. Each node represents an "actor", and each link between two actors indicates that one actor depends on the other for something in order that the former may attain some goal. We call the depending actor the **depender**, and the actor who is depended upon the **dependee**. The object around which the dependency relationship centres is called the **dependum**.

An actor is an active entity that carries out actions to achieve goals by exercising its knowhow. A dependency is intentional if the dependum is somehow related to some goals or desires of the depender. By depending on another actor for a dependum, an actor (the depender) is *able* to achieve goals that it was not able to do without the dependency, or not as easily or as well. At the same time, the depender becomes *vulnerable*. If the dependee fails to deliver the dependum, the depender would be adversely affected in its ability to achieve its goals. In the AD model, the depender's internal goals and desires are not explicitly modelled.

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 [12]. 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 based on a notion of *non-functional* requirements in software engineering [8][27].

In a **Resource-dependency**, one actor (the depender) depends on the other (the dependee) for the availability of an entity (physical or informational). By establishing this 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.

Figure 1a shows the diagrammatic notation for indicating that a nurse depends on the pharmacy for a patient's medication. A carpenter's dependence on a tool from a tool shop, a driver's dependence on gasoline from a gas station, a retailer's dependence on information about a customer's credit worthiness from the credit card company, could all be modelled as resource dependencies.

A resource dependency is different from the usual notion of a "flow" in that the latter does not indicate the presence or absence of intentionality in the flow. A resource flow does not necessarily imply a resource dependency. For example, if a cashier would not suffer whether or not she the customer's credit information was available, then that information is a mere flow, not a resource dependency.

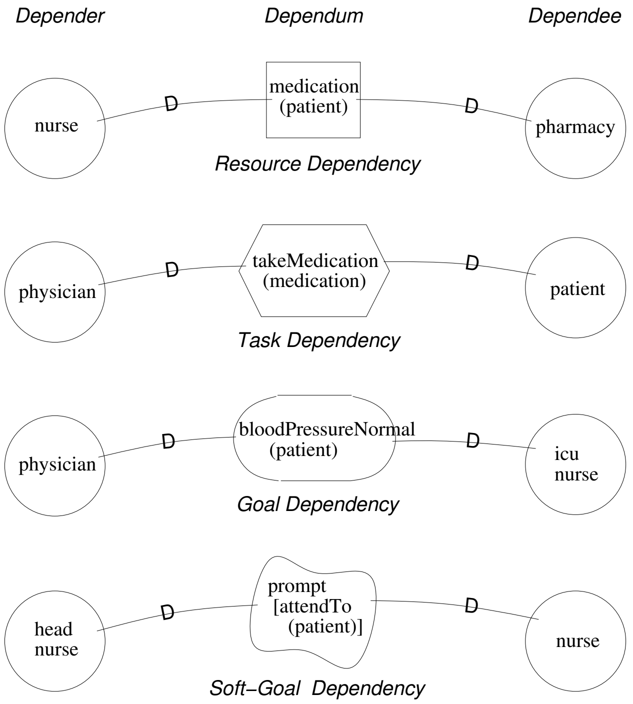

Figure 1: Four Types of Intentional Dependencies

In a **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 is vulnerable since the dependee may fail to perform the task.

When a physician asks her patient to take medication (e.g. four times a day, two hours after meals), she is depending on the patient to perform a task (figure 1b). If the patient does not take the medication (in the way specified), the physician's ability to cure the patient would be impaired. A carpenter's dependence on his apprentice's to run his errands (without explaining why), a passenger telling a driver which route to take (without indicating the destination), a store manager asking a clerk to stock the shelves in a certain way, these are examples of task-dependency.

Task specifications should be viewed as constraints rather than as the complete (and therefore adequate) knowhow for performing the task. This is one reason why a dependee may fail in performing the task. Another reason may be that the dependee decides not to perform the task, even when it is able to, e.g. if it decides there are more important things to do (which may be due to other commitments).

In a **Goal-dependency**, the depender depends on the dependee to bring about a certain state in the world. The dependee is given the freedom to choose how to do it. 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.

When a physician depends on an intensive-care unit nurse to keep a patient's blood pressure normal, this is a goal-dependency (figure 1c). The intensive-care nurse is trained to give the appropriate types of injections to maintain normal blood pressure. Leaving one's car with a repair shop saying "just get it fixed" is another example of a goal-dependency.

This notion of a goal is based on a clear-cut, black-and-white notion of goal achievement. The world is either in the stated condition or not. Often, there are goals that are not that sharply defined, but are goals nonetheless. They are not clear-cut because their meaning is not objectively known. There is no prior agreement between depender and dependee what constitutes the achievement of that goal. When this is the case, very often the goal is clarified during the process of trying to achieve the goal, by identifying alternatives, and the depender indicating which alternative to take. Often these alternatives are identified by the dependee, but the decision is taken by the depender. We call this a soft-goal-dependency.

In a **Soft-goal-dependency**, a depender depends on the dependee to perform some task that meets a soft-goal. The meaning of the soft-goal 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.

When a head nurse requires nurses in the ward to respond to patients *promptly*, the meaning of "promptly" is not clear-cut. A newcomer would have to find out in terms of what kinds of task should take priority. This is a Soft-goal-dependency. (figure 1d). (This type of dependency was called Goal-task-dependency in [37].)

These four types of dependencies also characterize how decisions fall on either side of the dependency. Under goal-dependency, the dependee is free to, and is expected to, make whatever decisions are necessary to achieve the goal (the dependum). Under task-dependency, the depender makes the decisions. The depender's goals are not given to the dependee. Under resource-dependency, the issue of decisions does not come up. A resource is the finished product of some deliberation-action process. It is assumed that there are no open issues or decisions to be addressed. Under soft-goal-dependency, the depender makes the final decision, but does with so with the benefit of the dependee's knowhow. The four types of dependencies thus indicate who will handle problems if and when they arise.

Given a non-intentional description of a relationship between two actors, eg. between a car owner and a repair shop with respect to the owner's car, 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. If, when a problem with the car arises, the two sides has no way of addressing the problem except to replace it with another car, then this is a Resource-dependency. If, when the car develops a problem, the owner tells the repair shop to "just get it fixed", this is a goal-dependency. If the owner tells the shop to "have it fixed economically", this is a soft-goal-dependency. For instance, if the car stalls easily, the shop might suggest overhauling the carburettor, or, as a stop-gap measure, one could raise the engine idle setting. The owner would select (and agree to) one alternative before the shop would start work. If the owner goes to the shop and asks to have the idle setting raised, without giving the shop the chance to suggest alternatives, this would be a task-dependency. The owner is telling the shop what to do, without making use of the shops knowhow for achieving the goal.

The set of nodes and links in an AD model form a dependency network. By following the *chains* of dependencies, an actor could use the network to explore the expanded possibilities that are accessible to it. From a vulnerability viewpoint, an actor could also use the dependency network to determine how it could be affected adversely by these dependencies.

Figure 2 shows a small example of an actor dependency network. A patient depends on a physician via a Goal-dependency on being cured. The patient must make himself available (Resource-dependency) for examination by the physician. The physician depends on the nurse to give medication to the patient (a Task-dependency). To do that, the nurse needs to have a prescription from the physician (Resource-dependency). The nurse gets the prescription filled by the pharmacy via a Task-dependency, and receives it via a Resource-dependency. The nurse gets the patients to take the medicine (Task-dependency). The patient depends on the nurse for the medicine (Resource-dependency).

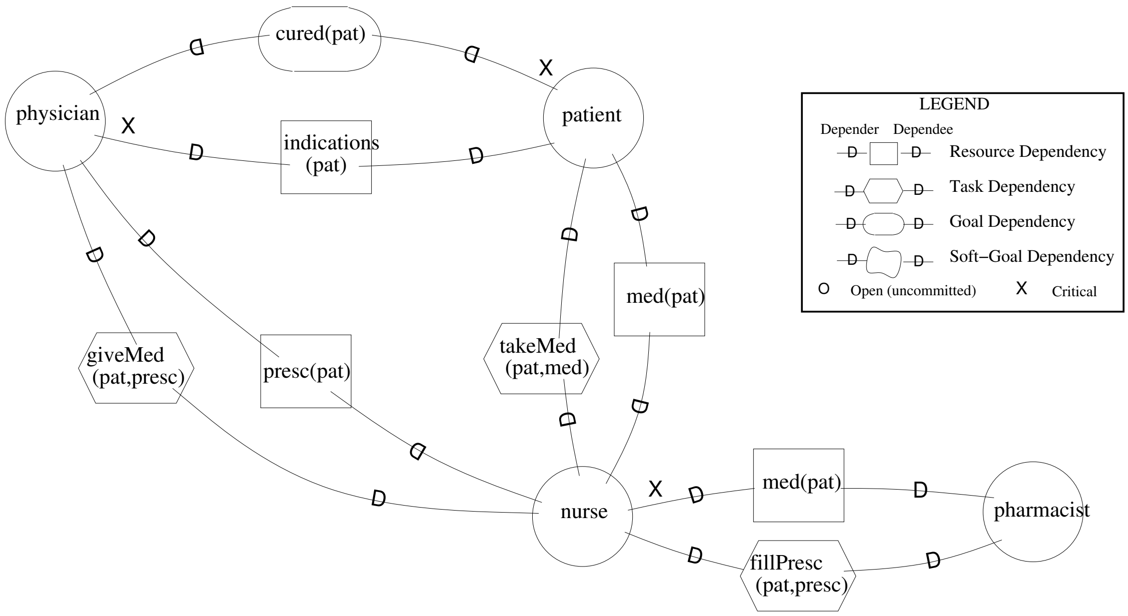

Figure 2: An Actor Dependency model – hospital example

It is important not to view these chains as mere flows. There is possibility of failure at each dependency, and we want to know its implication. For example, if the pharmacist sent the wrong medication, that could adversely affect the physician's ability to cure the patient. The type of dependency reveals whose goals might be affected by a problem, and indicates which side will deal with the contingency.

In a realistic setting, there are many more dependencies between actors. One can follow these chains very far afield. In practice, even though each dependency entails some potential contingencies, actors cannot constantly worry about all these vulnerabilities. One important way in which actors can be assured about its dependencies is by having *commitments*.

The notion of commitment has different significance on the two sides of a dependency. Given a commitment from the dependee, the depender would expect the dependee to try its best to deliver on the dependum, so that the depender could act as if the dependee will indeed deliver on the dependum. On the dependee side, in order to assure that it will be able to make good on its commitment, the dependee will in turn try to secure commitments from its own dependees.

Thus if the nurse is committed to giving medication to the patient, the physician will assume that the nurse will indeed do that, and handle whatever contingencies that may entail. For instance, the physician will not check whether the pharmacist has sent the correct medication, or whether the patient received the proper medication.

The model provides for the distinction between a **committed** dependee versus an **open** (uncommitted) dependee. Graphically, a dependency that has an uncommited dependee is marked with an "O" (for open) on the dependee side. An unmarked dependency link is a committed dependency.

That one can insulate the effect of failures through commitment is of course only an ideal. A commitment is not a guarantee. Actors are not always able to fulfill their commitments. When a depender can be significantly hurt by an unfulfilled commitment, it is likely to take stronger measures to assure itself of the viability of the dependency.

The model provides for three degrees of strength for a dependency on the depender side. An **open** dependency (marked by an "O" on the depender side) means that if the dependency fails, the depender would be affected to some extend, but not significantly. A **committed** dependency (unmarked) means that the depender would be hurt significantly if the dependency fails. Typically a commitment on the depender side means it has invested considerable resources (e.g. time and effort) in a course of action, which could not be reversed without loss. A **critical** dependency (marked by an "X" on the depender side) would indicate a goal could not be achieved due to the failure of the dependency. An interpretation of these distinctions in terms of plan failure was given in [37].

3.2 The Agents-Roles-Positions Model

An Agents-Roles-Positions (ARP) model consists of a set of nodes and links. Each node represents an **actor**, and each link represents an **association** or an **interdependency** between two actors.

Social actors have many dependencies. Grouping and categorizing dependencies as belonging to sub-units of a social actor can serve as a way of modelling the "internal" structure of an actor, while still relying primarily on an externally ascribed characterization of actors (based on dependencies with other actors).

We will continue to use the term **actor** generically to refer to any unit to which intentional dependencies can be ascribed. To model the sub-units of a complex social actor, we define three types of sub-units – agents, roles, and positions – each of which is an actor in a more specialized sense. We will use the term **social actor** when we want to emphasize that the actor in question is made up of a complex network of associated agents, roles and positions.

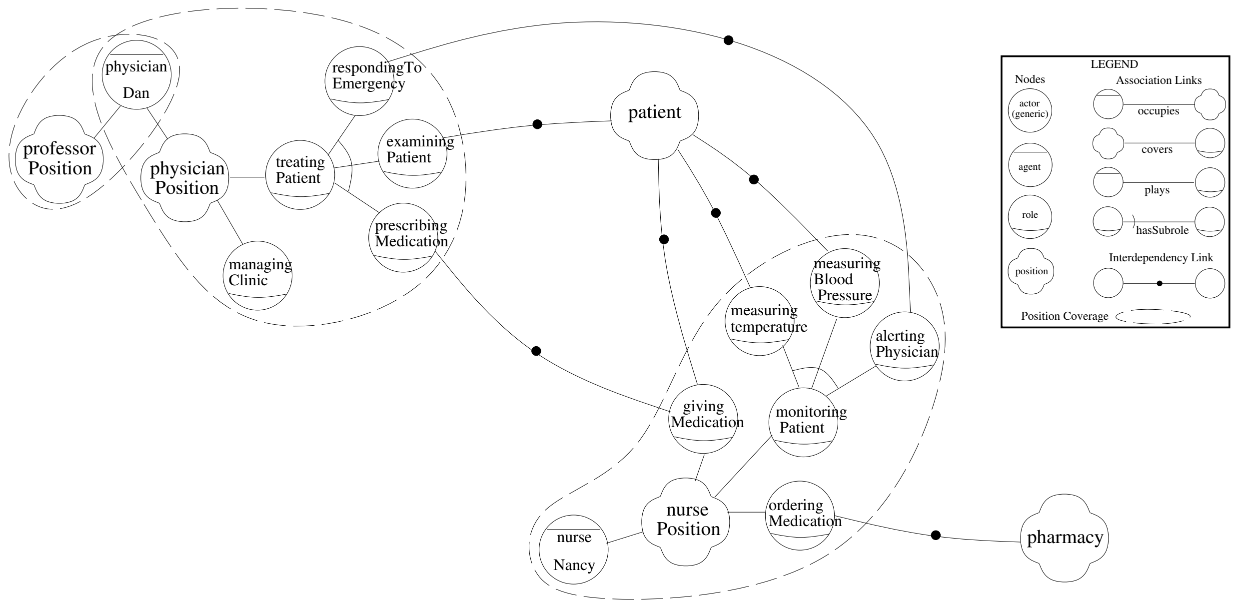

Figure 3: An Agents-Roles-Positions model – hospital example

A **role** is an abstract characterization of the behaviour of a social actor within some specialized context or domain of endeavour. Its characteristics are easily transferable to other social actors. A **position** is an abstract "place-holder" that mediates between agents and roles. It is a collection of roles that are to be played by the same agent. An **agent** refers to those aspects of a social actor that are closely tied to its being a concrete, physically embodied individual. Its characteristics are not easily transferable to other individuals, e.g. its skills and experiences. It has physical limitations.

Agents, roles, and positions are associated to each other via **association links**: an agent is said to **occupy** a position, a position is said to **cover** a role, and an agent is said to **play** a role. In general, the associations may be many-to-many.

By being associated, one actor "acquires" the characteristics of the other. The dependencies of one actor are induced onto the other. For example, all the expectations on a role are now also expectations on the agent who is playing that role. Since the dependencies as we have characterized them in the preceding section are intentional, the association between actors also become intentional. An association with all its induced dependencies we call the **intentional association** between two actors.

The ARP model offers another type of link which may be used independently of association links. An **interdependency** is a less detailed way of indicating the presence of dependency relationships between two actors. An interdependency link simply indicates that there is some intentional dependency between the two actors. Each interdependency link may have one or more underlying dependencies, in either or both directions. When used in conjunction with the Actor Dependency Model, each interdependency link may be expanded into its constituent dependencies, with details about their direction, type, dependum, and commitment and criticality characteristics.

An intentional association implies the presence of an interdependency link, but the converse does not hold. Typically, we use an interdependency link in an ARP model to represent the relationships across social actors, leaving the details of the dependencies to an AD model.

Each of the three kinds of actors – agents, roles, and positions – can have **sub-parts**. For example, a role can have a **sub-role**. These are treated as associations as well since they can also induce dependencies. Since sub-roles may be delegated, an agent's playing one role does not imply that it also plays all its sub-roles. Similarly, the coverage of a position must be explicitly stated. In the graphical notation, we use a dotted-line boundary to show the coverage of a position or an agent.

An ARP model provides a way to separately identify those dependencies that are attributable to a role, as opposed to those that are attributable directly to a concrete agent. Performance measures, for example, are typically characteristics of a concrete agent. Training and education background pertain to an agent, since they accompany the agent when he/she moves to another position. Qualification requirements, however, pertain to positions and roles. Roles typically have dependencies that relate to the "tasks" or "functions" that an agent may become associated with or disassociated with, for example, by taking on or moving away from a position.

When we separate out these "components" of a social actor, each component is a partial description of the social actor. Each component gives some hints about how the social actor might behave, or is expected to behave, in some specialized, narrow context. Such hints are useful because they give a first-cut approximation, thus simplifying the picture. But we also need to recognize its limitations, because they can often be misleading. It does not take the other aspects of the social actor into account. The behaviour of the social actor can only be understood in its totality.

Each association from agent to position to role can be problematic. For example, while there may be certain expectations on a role, when this role is covered under a position, the combination of roles may result in conflicts that makes those expectation less likely to be fulfilled. Similarly, a position may entail expectations that a concrete agent with specific abilities, knowledge, and aspirations may not exactly match. Figure 3 shows an example of an ARP model in a hospital setting.

When analysing actual organizations, it will be helpful to be able to make further distinctions between instance versus class, and be able to specialize and generalize class descriptions. These features have been well studied in conceptual modelling (e.g. [26]). We will not elaborate on them here.

3.3 Issue Argumentation Model

An Issue Argumentation (IA) model consists of a set of **nodes** and **links**. Each node represents an **issue**, and each link represents the relationship that one issue is a **contributing factor** towards the other.

When social actors face potential change in their work arrangements, they would have *concerns* about how they might be affected. These issues and concerns may alternatively be called interests, stakes, problems, complaints, aspirations, etc. We use *issue* as the generic term. An issue is expressed as a statement. It is not a factual statement about the world, but rather, an *intentional* or mental state, i.e. a relationship between an actor and a possible state of affairs in the world, rather than the actual state of affairs in the world. Examples of mental states are goals, beliefs, and commitments.

We will focus mainly only on the interpretation of these issues as *goals* (as in [27]). The statement or assertion describes a situation that an actor would like to come true. In other words, issues are viewed as goals or problems that need to be addressed or solved. In the context of the present framework, a solution usually is a reference to an organizational configuration (represented as elements of an AD or ARP model). The actions that an actor would pursue are those that would bring about a change in the organizational configuration (i.e. technological and/or organizational changes).

Usually a solution cannot be arrived at in a single step of reasoning from a problem statement. More likely, chains and networks of reasons are involved. Suppose an information system is about to be introduced into a hospital ward. A nurse might be concerned about the portability of the information that the system will produce. Why would the nurse want the information to be portable? – because she needs to use the information constantly. Why would she need the information constantly? – because she needs to constantly plan her activities for the rest of the shift. Why does she need to plan? – because that is the way to become an effective nurse. Why does nursing need to be effective? – because we want good quality care for the patients. But portability is likely to conflict with having information that is up-to-date. Furthermore, portability by itself is not enough to make the nurse's shift plannable. On the other hand, according to these nurses, if they have an overview of the ward, then that would be enough for them to plan their work and achieve effective coordination.

Note that this type of reasoning cannot be captured well by conventional logic. None of these factors are necessarily conclusive, and most of it involve subjective criteria. A physician, or another nurse, may evaluate the same issues differently, or bring in other issues that would alter the overall assessment of the alternatives. The goals in this type of reasoning are mostly "soft goals" (as discussed under the AD model). The interpretation of these goals are not hard and fast, nor objectively defined. The reasoning does not follow a simple true/false logic. All along the reasoning chain, and at various junctures in the network, each actor is making a series of judgements about the combined effects of many positive and negative factors.

Despite the softness of these goals, their meanings are not entirely open. Since actors eventually have to choose an alternative and take action, it they are ever to achieve their goals, they must stop considering ever more alternatives and contributing factors. An actor who acts after taking *enough* factors into account would be adopting a *satisficing* solution rather than an optimal one [31]. In deciding to pursue a goal in a particular way after having considered a particular set of issues, the actor has in effect given a specific interpretation to an otherwise soft goal. By recording what issues an actor has taken into account, and what it considers to be "enough", one can have a systematic treatment of a network of reasoning that is based on soft goals [27].

In this kind of reasoning, the progress from goals to solutions is in general not a simple top-down process, but may proceed in all directions. New issues will continue to arise as (prospective) solutions are generated. Problems are seldom fully solved. Instead, one might consider certain areas within a network of issues as having been adequately addressed, while acknowledging that new problems and issues may have been created in other areas.

The IA model provides a number of link types for differentiating among the different types of relationships that might exist between issues (nodes). A link may carry four kinds of information:

- the direction of the contribution – from which node to which node. In the graphical notation, we use arrowheads that point from the solution towards the goal.
- the sense of the contribution – whether the contribution is a positive or a negative one. Graphically, we use a "+" or a "–" annotation adjacent to the arrowhead.
- the extent or degree of the contribution – Following [27], we only distinguish between enough contribution versus not enough (partial) contribution. These are called **sup** and **sub** links in [27], connoting above and below. Graphically we use "Δ" and "∇" (for sup and sub respectively) in conjunction with the "+" and "–" signs. The "+" sign is assumed if omitted.
- how multiple links contribute towards a node – either as a combined effect, called **AND** (graphically a single line arc across the links), or as separate effects, called **OR** (graphically a double line arc).

To illustrate the various link types in the IA model, we will continue with the hospital ward information system example. Suppose nurses in the ward are considering two alternative proposals for a new information system. One provides interactive commands for nurses to query information about patients. The other, called a "worksheet system", focuses on producing a one-sheet printout which contains essential information about the entire ward for a particular shift (example setting adapted from [2]). The issues that the nurses bring to assess these alternatives may be arranged in a network as in figure 4.

The **und** link type (for undetermined) is used to indicate that one issue affects another in some way, but the sense of the contribution (whether positive or negative) is not known. For example, the nurses do not know whether the query-based system will help them reduce work and save time, or whether it will do the contrary. But they do believe the system will have some impact on these issues of concern to them. The *und* link is represented by an unannotated arrow in the IA network model.

A **positive** link indicates that one issue contributes positively to another, but the extent is not known. The nurses believe that the work sheet system will help them reduce work and save time, but they do not know by how much. A **negative** link represents a negative contribution of unknown extent. For example, the nurses are concerned that the standardization of work brought about by the worksheet system might contribute negatively to their goal of retaining individuality as persons.

A **+sub** link represents a positive contribution that is partial in extent, i.e. not enough to say that the goal has been met. Having patient information that is portable is one factor that will contribute towards make a nurse's shift more plannable, but by itself, portability is not sufficient to make the shift more plannable. Effective coordination contributes partially to better cooperation among a group of nurses, but other factors, such as team spirit, may also be required to bring about cooperation.

A **+sup** link indicates a positive contribution that is enough to address the goal. For example, the nurses believe that having an overview of the ward is enough to make their work in a shift more plannable. Having overview is also enough for achieving effective coordination. Further, the goal of having overview can be adequately addressed by having a worksheet system.

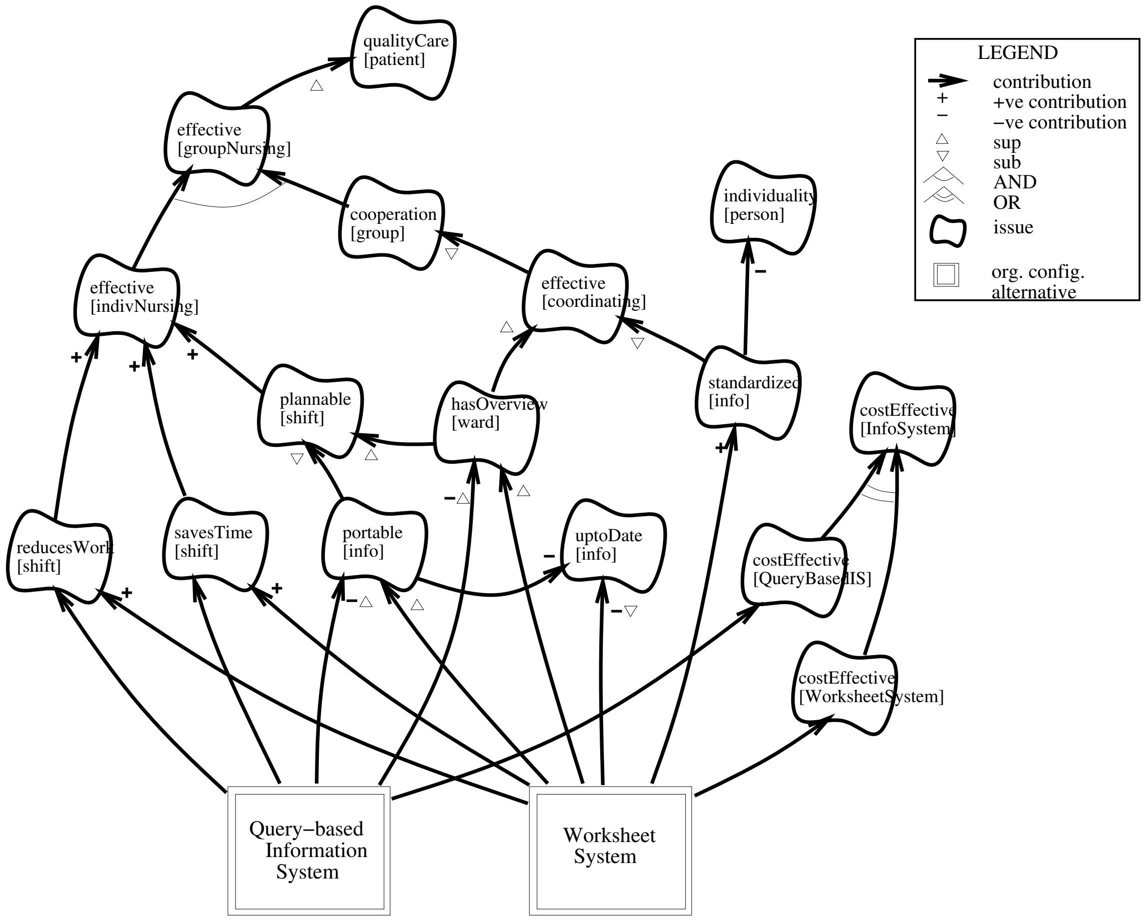

Figure 4: An Issue Argumentation model – hospital ward IS example

The −**sub** and −**sup** links are defined analogously. Printed worksheets are hard to keep up-to-date. But this negative contribution is only partial. Nurses do not mind, for example, incrementally doing updates by hand on the sheets. On the other hand, the goal of having overview has a *−sup* link from the query-based system, which means that the negative contribution was to an extent that makes the alternative unacceptable.

The **AND** link indicates that several contributions together make enough of a contribution to address a goal. For example, the goal of effective group nursing would be adequately addressed by the combined contributions of individual nurses being effective, and of having cooperation among the group of nurse. The **OR** link indicates that each of the individual contributions among several is enough to address a goal. If the goal is to have a cost-effective information system in the hospital ward, this could be met adequately either by having a query-based system that is cost-effective, or a worksheet system that is cost-effective.

4 An Example

In the preceding sections, we have used examples in a fragmented way to illustrate each of the features of the proposed modelling framework. We now illustrate how the various elements of the framework come together by applying the framework to a single coherent real-life example.

The example setting concerns the introduction of a design tool called Trillium into the machine interface design community of a large American corporation [3]. Trillium is a computer-based design environment used in the creation of the layout and logic of interaction for Control/Display interfaces on machines such as copiers and printers. [3] describes how the new technology altered the patterns of work relationships in the organization. The organization evolved in a way that resulted from an interplay of many factors, and did not follow the straight forward path that was originally intended. Further, the way the tool was used in the organization impacted the demands and requirements on the tool itself, and influenced its evolution.

From the description in [3], one could recognize many of the key concepts offered by the proposed modelling framework. Each of the models in the framework helps highlight certain aspects of the organizational setting.

Dependence is one of the key concepts used to explain many features of the organization. Organizational members depended on each other for achieving goals such as producing good user interfaces on time. Dependence is understood *intentionally* in that goal achievement would be affected if the dependencies did not hold up.

The concepts of *roles, positions,* and *agents* also provide useful distinctions in understanding how the organization responded to the new technology. Before Trillium was introduced, the organization consisted of three (types of) positions – program planners, designers, and programmers (figure 5). The designer position was occupied by individuals (agents) with training in industrial design or human factors psychology, while the programmer position was occupied by computing professionals. A major part of the organizational impact of Trillium was how it altered and rearranged work roles that are associated with positions, and the types of dependencies among them.

Issues and concerns motivated (some) organizational actors to introduce the new technology, and led them to respond to the technology in the way they did. For program planners, the main issue was to improve the user interface development process. This led the designers to explore different ways of relating to the programmers, with issues of power and control as important undercurrents. Issues of professional self-image affected how agents were willing or unwilling to accept learning the new technology. When the issue of usability of the Trillium tool was overcome by creating a new position, requirements issues on the tool shifted from usability to better functionality and speed performance.

[3] described how two different groups evolved along different paths in response to the introduction of Trillium. We will only use one of these to illustrate our framework.

To better appreciate how each of the three types of models contribute to an overall treatment of the example, we first present the configurations as stages that the organization evolved through, using AD models (figure 6). We then incorporate elements from IA and ARP models to form a combined model (figure 7).

Three Stages of Evolution. The AD model of figure 6a elaborates on the dependencies that appear as interdependency links in the ARP model of figure 6. Program planners depend on the designers to design the user interface (UI) for a new product (e.g. a copier), and on the programmers to produce implementions that will run on the target machine. We model these as Goal-dependencies since the dependees are expected to use

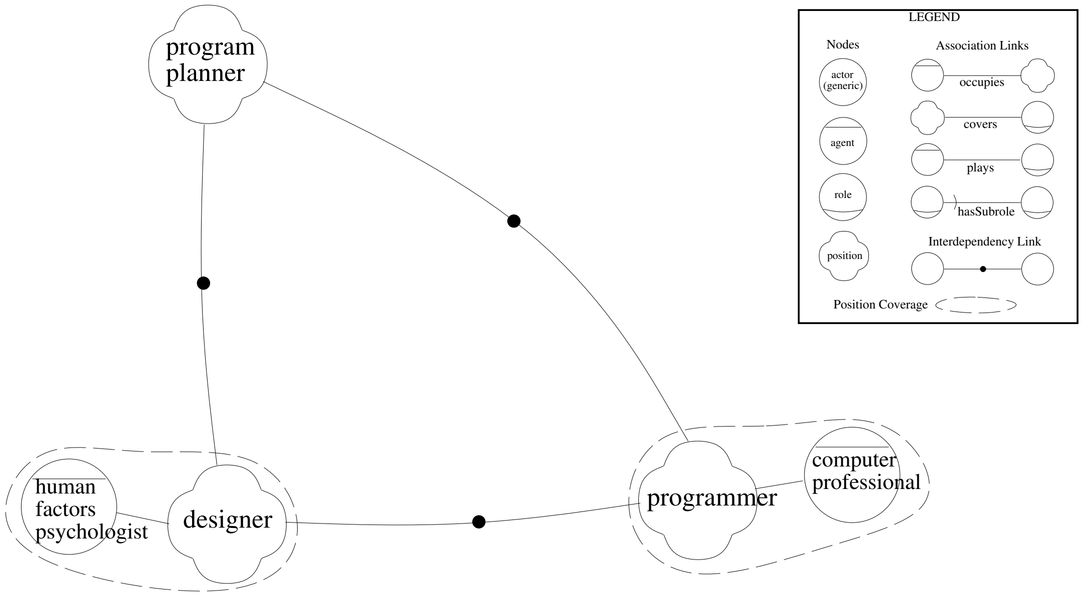

Figure 5: An Agents-Roles-Positions model of the initial configuration (before Trillium)

their knowhow freely to achieve the goals – namely, that there will be a UI, and an implementation of it. There are also accompanying soft goals, such as that the UI design be a good one, and that the implementation be completed on time for product introduction.

The nature of the dependence between designers and programmers turned out to be of crucial interest in this organizational setting. Since designers are committed to producing good user interfaces, they are dependent on programmers to bring their design concepts into concrete realization. If designers were able to describe their designs precisely (e.g. as in a formal notation), they could adopt a Goal-dependency relationship with the programmers. The dependum would be something like "meet(UI.imp, UI.spec)", where there are reasonably clear-cut definitions and criteria for deciding whether an implementation meets the spec.

This was not the case. Consisting of informal diagrams and descriptions, the specs were often incomplete and open to interpretation. Designers expected programmers to consult them during the implementation process, until they were sufficiently satisfied that the implementation meets the spec. Under our framework, this would be a Soft-Goal-dependency. (The choice of terms such as "meet", "conform", and "emulate" are only for mnemonic purpose; what really matters is the dependency type.)

Figure 6b shows the intended organizational configuration with the Trillium tool introduced into the organization. From the designers' viewpoint, the main feature to note is that the relationship between designers and programmers is now a Goal-dependency. The idea was to use the Trillium tool to build a prototype that would embody the working behaviour of the UI design. The programmers would then be asked to make the implementation produce the same behaviour as the prototype. Designers would become users of the Trillium tool, via a resource dependency on the software engineers who designed and supported the tool (the "supporter" position).

Aftering trying out this work arrangement for a few months, designers were still experiencing difficulty with using the tool. The organization subsequently evolved into the configuration of figure 6c. This involved the creation of a new position called a "Trillium implementor". This position was filled by persons (agents) who were trained as designers but had become specialists in using the Trillium tool. Trillium implementors work with designers via a Soft-goal-dependency to produce a prototype. Designers are then able to depend on programmers to provide an implementation via a Goal-dependency. An unanticipated development was that, once Trillium implementors were able to produce prototypes, the program planners also wanted to use them for the purpose of project reviews.

Actors' Reasoning About Trillium and Organizational Change The above models (figure 6a, b, and c) do not show the actors' reasoning that led the organization to evolve in the way it did. We now try to express

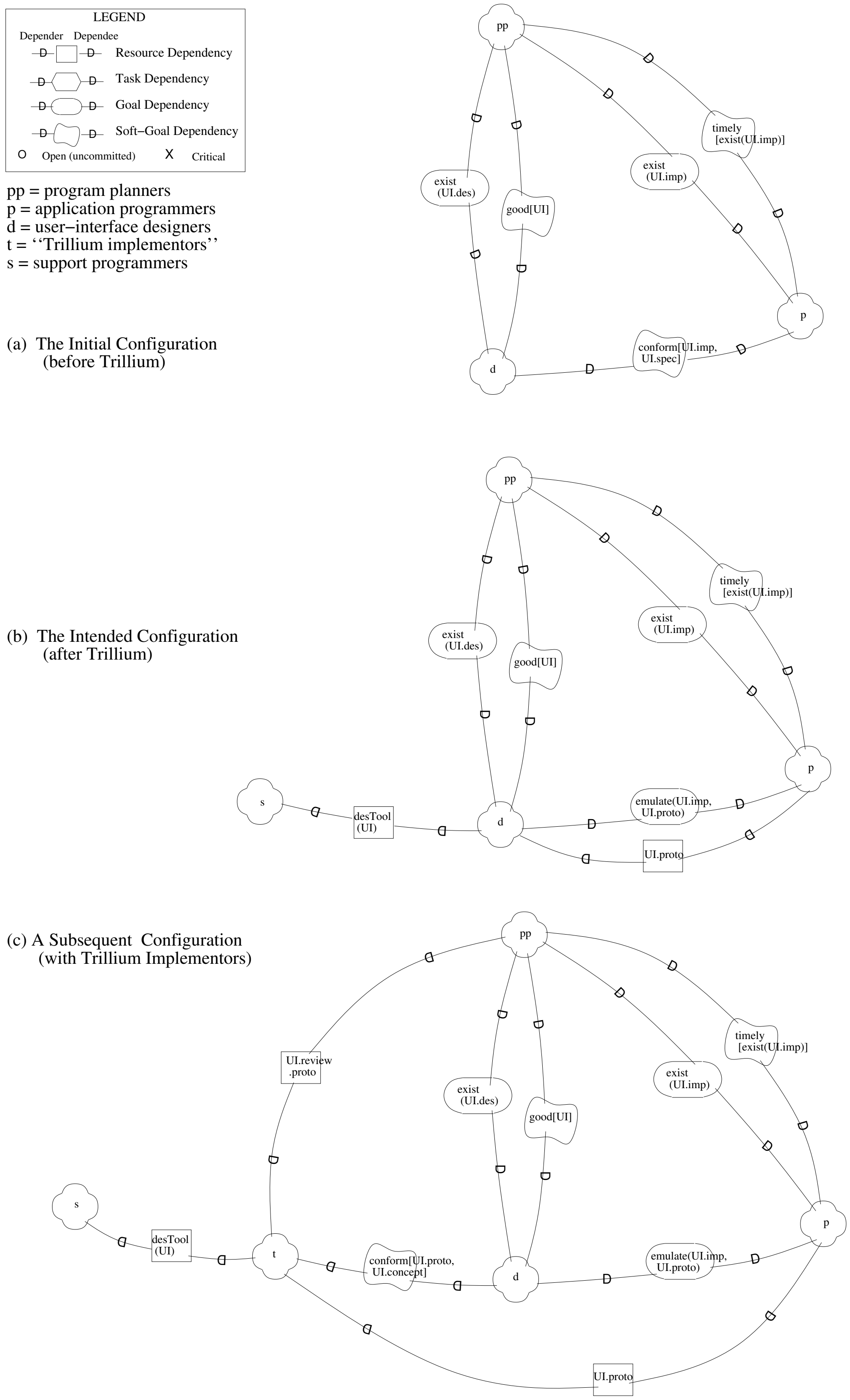

Figure 6: Actor Dependency models at three stages of evolution

the reasoning given in [3] in a combined model.

The combined model (figure 7) has elements from all three types of models. IA elements (heavy lines) are used to show the issues and arguments about alternative organizational configurations. These configurations or work patterns are shown as combinations of AD model elements (intentional dependencies) and ARP model elements (interdependencies and associations, especially sub-role relationships, to show the changes in roles). The coverage boundaries (dotted lines) also serve to indicate the actor viewpoint from which issues are perceived.

Let us focus first on the configuration alternatives (lighter lines) in this combined model. One of the roles that program planners have is to develop user interfaces. This role has as sub-parts planning, designing, and implementing. The latter two are delegated to designers and programmers respectively, via Goal-dependencies. The way in which the designers' roles evolved in response to the introduction of Trillium was the focal point of this case example. Designing requires doing the actual design and getting the design implemented. Three alternatives for getting designs implemented were attempted or used. The first two are by way of direct dependencies on programmers (Soft-goal and Goal-dependencies respectively). The third one involved a Goal-dependency on programmers to emulate a prototype. This requires getting a prototype produced. One way is for the designer to do design using the tool. Another was to delegate the prototype building to specialized Trillium implementors.

Let us now turn to the IA elements (heavy lines) that express actors' reasoning about these configurations. Program planners has the role of improving the development process. To the designers, one factor that can contribute to a better development process is for them to have better control over the process, in particular, on how their designs eventually appear in the final product. Because they are not able to specify their designs precisely and completely, they would want to be able to iterate on what the programmers have implemented, and alter the design until they (the designers) are satisfied – a Soft-goal type of dependency.

Although this was what the designers would be satisfied with (the +sup link from "conform" to "control"), it was not the relationship they were getting. By appealing to their timeliness commitment to the program planners, the programmers were able to make it difficult for designers to make changes to the UI design once the programmers were given the spec. In effect, the programmers were operating on a *de facto* Goal-dependency relationship with the designers. The programmers were free to make decisions about how the final interface looked like, usurping the control that the designers needed to have to ensure their conception of a good design. The programmers succeeded in enjoying a powerful position with respect to the designers (the +sup link from "meet" to "power"), with the opposite effect on the designers' side.

With the Trillium tool, the designers hoped to be able to give the programmers a much more precise representation of what they wanted, in the form of a prototype that would demonstrate the working behaviour of the UI design. Given this more precise representation, the designers would be satisfied with a Goal-dependency relationship with the programmers (+sup from "emulate" to "preciseRep"), since it would be clear whether a finished implementation is correctly emulating the behaviour that is embodied in the working prototype. This would give the designers the control that they wanted (+sup link), and has the added benefit of shifting power from the programmers back to the designers (+ve link).

When the designers attempted to follow this route, they found that the usability of the tool was unacceptable (-sup link). Further, they felt that learning to use the tool conflicted (-ve link) with their self-image of what designers are supposed to spend their time on, and distracted them from their proper role of doing actual designing.

The alternative to this was to have some members of the team become specialized expert users of the Trillium tool. Under the team approach, the designers would focus on producing the conceptual design, while delegating the prototyping to the Trillium implementors. This delegation still needed to be based on an Soft-goal-dependency, since design concepts needed to be tried out iteratively on the prototype and refined. However, communicating with Trillium implementors was considerably easier (+ve link from "conform[UI.proto, UI.concept]" to "ease[comm]") than communicating with programmers (-ve link from "conform[UI.imp, UI.spec]" to "ease[comm]"). This is because the Trillium implementor position was occupied by agents who have the same background and training as those occupying the designer position.

Since Trillium implementors had as their primary role that of using Trillium to produce prototypes, they perceived efforts spent learning to use Trillium as furthering their professional expertise (+ve link) and were able to overcome the usability problem that the designers had. The demands on the support programmers who provided the Trillium tool shifted from usability to better functionality and speed performance.

Once the prototyping process became viable with the team approach, program planners began to use the

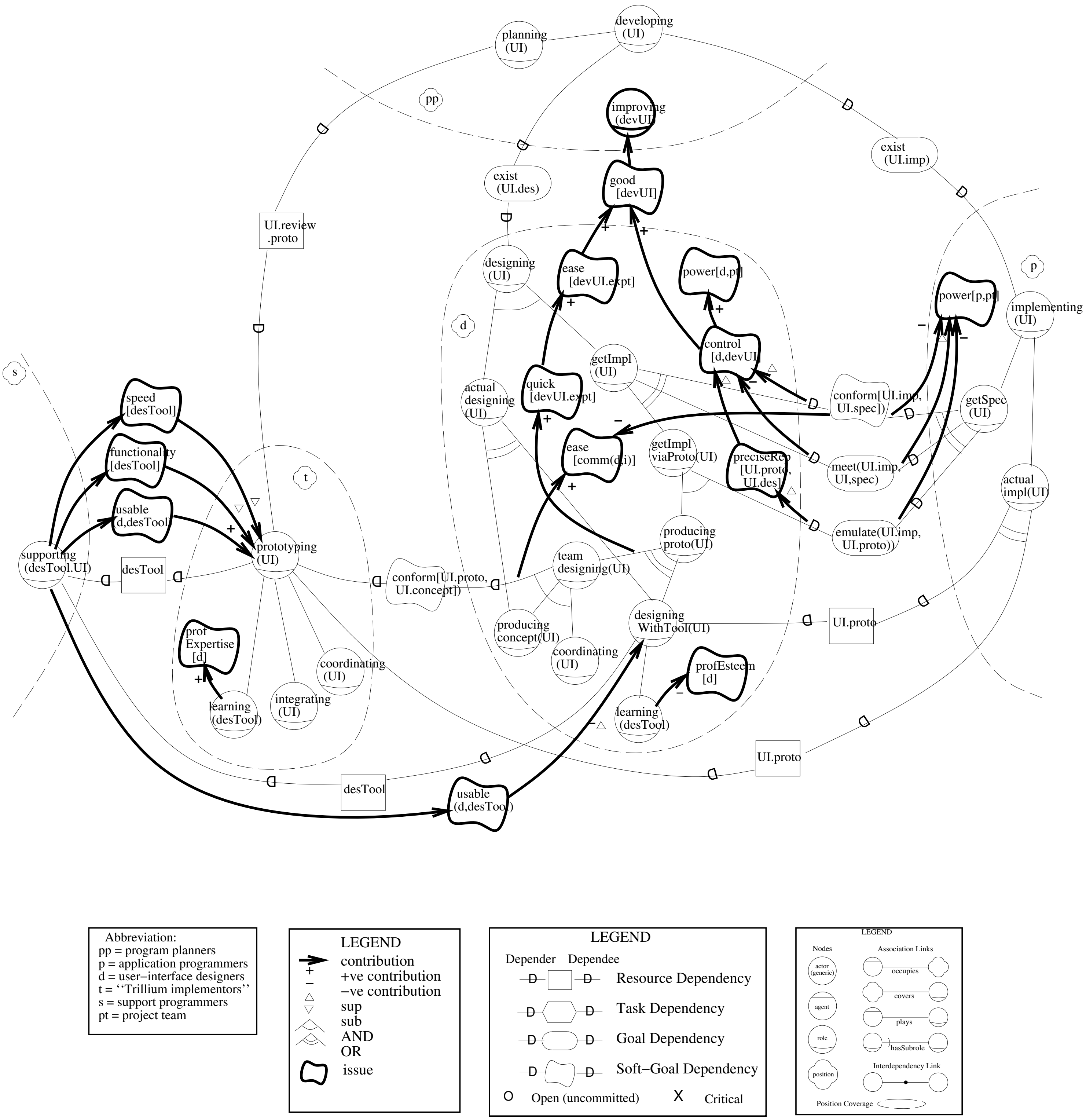

Figure 7: A Combined Model for the Trillium example

prototypes for reviewing and evaluating UI designs at an earlier point in the development process. This also allowed designers to make convincing arguments about innovative design concepts, and were able to argue for better allocation of resources to their efforts, further improving their power within the organization (not shown in model). By being dependees to program planners as well as to designers, Trillium implementors were able to consolidate their positions in the organization.

Observations on the Application of the Modelling Framework to the Trillium Example. Although the analysis in [3] was not explicitly multi-perspective, the issues and concerns that were raised included key concepts that would be emphasized by several major analytical perspectives (e.g. as discussed in [17]). Program planners' desire for better quality, faster development cycle would be dominant issues from a rational perspective. Designers and programmers were quick to see, from a political perspective, the implications of the new technology on control and distribution of power in the organization. The meaning and perception of the new technology as something incompatible with the proper role of a designer could be interpreted from a symbolic interactionist perspective. Finally, the program planners' use of the prototypes as a resource to reduce project management uncertainty would fit well into a structural contingency perspective.

What this framework provided is a systematic way to first identify and display the organizational configuration alternatives (the AD and ARP models), and then arrange the issues and concerns of actors into a network of arguments (the IA model). Issues from various perspectives interact with each other, through positive and negative contributions, leading to adoption of an alternative when actors felt that their issues and concerns would be adequately addressed by pursuing that alternative.

5 Related Work and Discussion

In this section, we compare the concepts and features in the proposed framework with previous work in social studies as well as in the technical literature in computing science.

The Actor Dependency Model. Notions of dependence have been widely used in conceptions of organizations. [34] identified three types of interdependence: pooled, sequential, and reciprocal. [28] used resource dependence as a central concept for analysing relationships between organizations. The network concept in [19] has the same spirit as a dependency network. The notion of production lattice also suggests a network of dependencies. Intentionality in implied by the discussion about success and failure. The notion of common object [22] is similar to our notion of a dependum. However, goals or tasks were not identified as potential common objects. Dependencies are not explicitly formulated as being intentional.

We have focused on a notion of *intentional* dependency as a way of formalizing a more precise relationship between depender and dependee. By adapting models of agents in artificial intelligence [10] [33], an intentional notion of dependency can be characterized using intentional modal operators such as goal, belief, and ability [37].

Most discussions of dependency, both in the social analysis literature [28] [19] and in the technical literature [7], emphasize resource dependency. We use the four-fold distinction among Resource-, Task-, Goal-, and Soft Goal-dependency to reflect the problematic nature of work situations surrounding any dependency, and to identify which side of a dependency is expected to handle problems should they arise.

Commitment has been recognized as an important concept in the social sciences [1] and in the technical literature [10] [33]. By building a notion of commitment into the Actor Dependency model, we echo the view of an organization as a network of commitments [36]. In doing so, we note that there needs to be two notions of commitment, one for the depender (addressed in [1] and [33]) and one for the dependee (addressed in [10] and [36]).

The Agents-Roles-Positions Model. The concepts in this model were not motivated originally to capture some phenomena in the social world. Rather, they were intended to provide an extension to the Actor Dependency model, so that groupings of dependencies and how they can be associated with different types of actors at different times could be modelled. Nevertheless, the concepts in the model appear to be compatible with the basic ideas behind the corresponding concepts in the social science literature (e.g. [35]). Agents (more often referred to as individuals or actors) play (or take) roles, and are occupants (or incumbents) in social positions. These relationships correspond to association links in the ARP model. A role is a set of expectations on an actor's behaviour. This seems to be compatible with our characterization of role as a set of intentional dependencies.

The Issue Argumentation Model. In the social science literature, issues and arguments are usually presented in narrative text. Box-and-arrow models (with '+' and '–' annotations) are sometimes used, usually to indicate the causal or theoretical relationships among variables [35], rather than to represent the intentional reasoning of particular actors. Models of formal reasoning when applied to social studies tend to be deductive and rationalistic. Such model are difficult to accommodate multiple perspective reasoning.

Models of argumentation have been adapted for use in software engineering to capture arguments about design alternatives and to support decision making [29] [21]. [8] and [27] has further specialized these frameworks by making the distinction between functional and non-functional goals, and by incorporating a notion of satisficing.

The Issue Argumentation model in the proposed framework is an application of a subset of the Non-Functional Requirements framework to the organization modelling domain. The notion of a soft goal in the Actor Dependency Model is also based on this formulation of non-functional goals.

6 Conclusions

There is increasing recognition that effective IS design requires taking into account a broad range of organizational analysis perspectives. In order to support organizational analysis as an integral part of the overall information system development process, there is need for a practical and systematic framework to utilize insights from these broader perspectives to guide design.

In this paper, we have presented a set of modelling concepts which may serve as a basis for employing a conceptual modelling approach to link organizational analysis to technical system design. We have focused on *intentional dependency* as a key concept in modelling organizational configurations of social actors. The concepts of *roles, positions,* and *agents* were used to differentiate the many facets of social actors. *Issues* and concerns of actors such as power, prestige, and autonomy are argued about informally within a qualitative reasoning framework, thus allowing potentially incompatible concepts from different analytical perspectives to be managed and assessed. The framework was tested against an example drawn from the literature which reports the ethnographic study of the introduction of a technical system into an actual organization.

In order for the framework to be sufficiently developed for practical use, many steps remain to be taken. Among these, we intend to further elaborate on the prospective uses of these models. Given a model of a particular organization, what interesting properties can be derived from the model. This will require a more precise characterization of the modelling concepts. The expressive adequacy and usefulness of these concepts will also need to be tested on a broader range of organizational settings.

Acknowledgements

Sincere thanks are due to Prof. John Mylopoulos for his encouragement and guidance, to Lawrence Chung for many helpful discussions, and to Prof. Fred Lochovsky for many earlier discussions about organizational computing. Special thanks are due to Jeanette Blomberg, who kindly discussed the Trillium case example with the author.

References

[1] H. S. Becker, Notes on the Concept of Commitment, *American Jounral of Sociology*, vol 66, July 1960, pp. 32-40.

[2] G. Bjerknes, T. Bratteteig, The Memoirs of Two Survivors, or, Evaluation of a Computer System for Cooperative Work, *Proceedings of the Conference on Computer-Supported Cooperative Work*, 1988, pp. 167-177.

[3] J. L. Blomberg, The Variable Impact of Computer Technologies on the Organization of Work Activities, *Proceedings of the Conference on Computer-Supported Cooperative Work*, pp. 35-42, 1986, also in *Computer-Supported Cooperative Work – A Book of Readings*, I. Greif, ed., pp. 771-781.

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

[5] M. L. Brodie, J. Mylopoulos, J. W. Schmidt, eds., *On Conceptual Modelling: Perspectives from Artificial Intelligence, Databases, and Programming Languages*, Springer-Verlag, 1984.

[6] J. A. Bubenko, Next Generation Information Systems: an Organizational Perspective *Intl. Workshop on Development of Intelligent Information Systems*, Niagara-on-the-Lake, Ontario, Canada, April 21-23, 1991, pp. 22-31.

[7] 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.

[8] L. Chung, Representation and Utilization of Non-Functional Requirements for Information System Design. *Advanced Information Systems Engineering, Proc. 3rd Intl. Conf. CAiSE, Trondheim, Norway,* R. Anderson etal. (eds.), Springer-Verlag, pp. 5-30, 1991.

[9] A. Clement, C. C. Gotlieb, Evolution of an Organization Interface: The New Business Department at a Large Insurance Firm, *Trans. Office Information Systems*, 1977.

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

[11] L. Gasser, Social Conceptions of Knowledge and Action: DAI Foundations and Open Systems Semantics, *Artif. Intell.,* 47, Jan. 1991, pp. 107-138.

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

[13] J. Grudin, Why CSCW Applications Fail: Problems in the Design and Evaln of Organizational Interfaces, *Proc. Conference on Computer-Supported Cooperative Work* 1988, pp. 85-93.

[14] C. Hewitt, Offices Are Open Systems, *Trans. Office Information Systems*, vol. 4, no. 3, July 1986, pp. 271-287.

[15] 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.

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

[17] R. Kling, W. Scacchi, The Web of Computing: Computer Technology as Social Organization *Advances in Computing,* vol. 21, 1982, pp. 1-90.

[18] R. Kling, Organizational Analysis in Computer Science, *The Information Society,* vol. 9, no. 2, 1993.

[19] 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.

[20] R. Kling, Cooperation, Coordination and Control in Computer-Supported Work *Communications of the ACM,* vol. 34, no. 12, December 1991, pp. 83-88.

[21] J. Lee, SIBYL: A Qualitative Decision Management System, *Artificial Intelligence at MIT: Expanding Frontiers*, P. Winston, ed., with S. Shellard, Cambridge, MA: MIT Press, June 1990.

[22] T. W. Malone, K. Crowston, What is Coordination Theory and How Can It Help Design Cooperative Work Systems? *CSCW 90 Proc.* , 1990, pp. 357-370.

[23] L. Markus, Power, Politics, and MIS Implementation, *Communications of the ACM*, vol. 26, no. 6, June 1983, pp. 430-444.

[24] G. Morgan, *Images of Organization*, Sage Publications, 1986.

[25] 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.

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

[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] J. Pfeffer, G. Salancik, *The External Control of Organizations: A Resource Dependence Perspective*, Harper and Row, 1978.

[29] C. Potts, G. Bruns, Recording the Reasons for Design Decisions, *Proc. 10th International conference on Software Engineering* , 1988, pp. 418-427.

[30] W. R. Scott, *Organizations: Rational, Natural, and Open Systems*, 2nd ed., Prentice Hall, 1987.

[31] H. A. Simon, *The Sciences of the Artificial*, 2nd ed., MIT Press, 1981.

[32] 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.

[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] J. D. Thompson, *Organizations in Action,* McGraw-Hill, 1967.

[35] J. H. Turner, *The Structure of Sociological Theory*, 4th ed., Dorsey Press, 1986.

[36] T. Winograd, F. Flores, *Understanding Computers and Cognition*, Reading, MA: Addison-Wesley, 1987.

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