Agent Orientation as a Modelling Paradigm[1]

Eric Yu
University of Toronto
http://www.fis.utoronto.ca/~yu

ABSTRACT

Agent orientation is emerging as a new paradigm for constructing software systems. Agent-oriented systems are expected to be more powerful, more flexible, and more robust than conventional software systems. This paper argues for a shift towards agent orientation at the level of requirements engineering, quite separately from the concept of agent as a software construct. Requirements analysis relies on models to characterize systems and their environments in order to understand user needs and stakeholder wants, assess the viability of alternative systems proposals, and to arrive at a definition of the system to be developed. The success of any software system depends crucially on the effectiveness of the requirements process. The complexity of today's systems and our expectations on them have grown to a point where a social paradigm for modelling and analysis has become more appropriate than the traditional mechanistic paradigms. A requirements modelling approach centred around a concept of *agent* can offer powerful new ways for characterizing and analyzing the interactions and relationships among the many semi-autonomous entities in the system and in the environment. Based on observations about the changing needs of requirements modelling, we propose six properties that are desirable for a concept of agent as a modelling construct -- intentionality, autonomy, sociality, contingent identity and boundaries, strategic reflectivity, and rational self-interest.

1 Introduction

Agent orientation is emerging as a new paradigm for constructing software systems. New kinds of systems are now being developed based on the concept of software agent. According to one definition (Jennings et al., 1998), software agents are situated -- they sense the environment and perform actions that change the environment; they are autonomous -- they have control over their own actions and internal states, and can act without direct intervention from humans; and they are flexible -- responsive to changes in the environment, goal-oriented, opportunistic, and take initiatives; they are also social -- they interact with other artificial agents and humans to complete their tasks and help others. Agent orientation is offering an exciting new way of thinking about what software is and how it should be constructed. Compared to the dominant software paradigm of the day, namely object orientation, agent orientation offers a higher level of abstraction for thinking about the characteristics and behaviours of software systems. It can be seen as part of an ongoing trend towards greater interactivity in conceptions of programming and software system construction (Newell, 1982, Bobrow, 1992, Wegner, 1997).

While developing more powerful and robust software capabilities is clearly fundamental to the enterprise of software and information systems, it is also clear that we need effective techniques for determining the needs and requirements for particular application settings, so that *the right system* will be built to meet those needs. The success of any software system depends crucially on the requirements and how well they are addressed during the development process. The requirements engineering community has been actively developing new concepts and techniques

[1] A revised version of this paper appears in *Wirtschaftsinformatik*. 43(2) April 2001. pp. 123-132.

(van Lamsweerde, 2000, Nuseibeh & Easterbrook, 2000). This paper argues that agent orientation can be as significant a paradigm shift for requirements engineering as for software construction.

Models and languages are crucial for requirements engineering. They allow the right kinds of knowledge to be expressed, in order to support the right kinds of analysis and reasoning. As the context and needs of requirements engineering change, advances in modelling and languages are also needed to respond to those changes. Traditional modelling techniques reflect a mechanistic worldview in that they focus on specifications of behaviour that are known in detail and fully controllable. Today's systems and their environments are more varied and dynamic, accommodate more local freedom and initiative, and have to cope with limited knowledge and control.

An appropriate concept of *agent* can be developed to serve as the central construct in a new kind of modelling and analysis to respond to today's needs. Much like the concepts of activity and object that have played pivotal roles in earlier modelling paradigms, the agent concept can be instrumental in bringing about a shift to a much richer, socially-oriented ontology that is needed to characterize and analyze today's systems and environments.

In section 2, we examine some factors that are motivating a shift in the kind of modelling and analysis that is needed for requirements engineering in today's environments. In section 3, we propose six properties that a concept of agent should have as a construct for requirements modelling. These properties are distinct from those associated with software agents, because of the distinctive nature of requirements modelling. Section 4 briefly reviews the ***i**** modelling framework which exemplifies some of the desired properties of agent-oriented modelling. Section 5 concludes with a look at related work and open research issues.

2 The Changing Needs of Requirements Modelling

Requirements engineering aims to uncover and analyze relevant facts and relationships in the world so as to help propose new systems (or modifications to existing systems) that will solve problems or exploit opportunities for stakeholders. In the past, the emphasis has been on the capture, analysis, and specification of what users and clients want in a system. Consistency, completeness, and validation of requirements have been the dominant objectives. Today, the interaction between systems and their environments have become much more complex and intricate. Advances in systems technologies have resulted in many new ways in which human organizations can take advantage of technologies. These need to be explored and analyzed during the requirements stage. The requirements process needs to go beyond elicitation and elaboration of details. It needs to support the exploration of potential alternatives, understanding their implications, and relating them to higher-level goals for stakeholders (Bubenko, 1995, van Lamsweerde 2000).

In this section, we highlight a number of ways in which the context for requirements engineering has been changing. The combination of these forces suggests that a move towards a social paradigm for modelling and analysis would be desirable and beneficial.

1) Technology as enabler

Information technology has taken on much greater significance in organizations today. Systems are not used merely to automate well-established tasks and processes -- they are now used to re-design business processes, re-invent organizations, and even to create entirely new businesses and industries (Hammer, 1990, Davenport, 1992). Systems professionals are expected to help organizations explore and discover innovative ways to leverage technology, to meet business goals, to build competitive advantage, and to challenge conventional wisdom and vision. Requirements analysis therefore needs to become much more interactive and participatory. Instead of taking requirements as given, requirements analysts need to understand more about business goals and the organizational environment. They need to work with organizational participants to uncover hidden assumptions and rationales, to develop and explore the space of possible options, and to analyze their implications (Bubenko, 1995).

2) Networked systems and organizations

Distributed systems technologies have created systems that can be partitioned in many different ways. Networking allows disparate and widely distributed components to be brought back together to form systems. This flexibility in systems topology is being paralleled by similar developments in human organizations. Organizations have become less hierarchical, making more use of lateral structures such as teams, and have become more distributed and networked (van Alstyne, 1997). Market-based arrangements offer a even greater degree of flexibility and dynamism. Organizations use outsourcing, sub-contracting, and virtual enterprises to adapt to changing needs. Technical system functionalities have been dissected and re-packaged into commercial-off-the-shelf (COTS) packages, application service providers, application frameworks, etc. Network-centric computing creates end-user services on-the-fly using component services and resources dynamically assembled from over the network (Shaw, 2000).

Consequently, requirements engineering now has to deal with systems and environments that are much more dispersed, fluid, contingent, and even ephemeral. The single boundary between system and environment has now become a thicket of connections and inter-relations across a multitude of elements on each side because each of those elements may have its own dynamics. Furthermore, social structures are often changed substantially as systems are introduced or modified. Responsibilities, accountability, authority, ownership, and control are redefined and redistributed. New patterns of power and influence may emerge. Development efforts that do not take these into account often fail.

3) Increased interdependency and vulnerability

Since systems and their environments are now likely to be made up of many semi-autonomous units, each potentially conceived, built, operated, and maintained by different groups or organizations, there are many interdependencies. For example, in healthcare, there are numerous systems and subsystems which depend on each other for clinical support, logistics, administration and billing, quality analysis, research, etc. Requirements analysis needs to support ways for identifying dependencies (both direct and indirect) and analyzing their impacts, including the impact of changes. Dependencies imply vulnerabilities. Each dependency has potential benefits and liabilities. Successful operation is contingent upon a viable network of interdependencies. Support for analyzing vulnerabilities and ways for mitigating them are needed. One would like to be able to state requirements on interdependencies and be able to reason about them, e.g., which parts of a system should or should not depend on which other parts in what ways, and why this is good or bad. The problem of “feature interaction” in telecommunications systems is an example of consequences of unanticipated interdependencies.

4) Limited knowledge and control

Because of the distributed and networked nature of systems and their surrounding organizational environments, the different parts of a system or the environment typically have limited knowledge about, and control over, other parts. Traditional requirements techniques tend to assume that the development organization (with the support of the sponsor) has full control over what goes into a system. This is no longer true when capabilities and authorities are spread out over different jurisdictions. For example, telecommunications network management crosses over many administrative domains and ownership boundaries. Unlike in systems conceived and designed under a single coherent framework, knowledge and control cannot be taken for granted. Requirements techniques need to be able to express what is knowable and what is controllable across system components as well as environmental elements, and to reason about the desirability of such relationships.

5) Openness and uncertainties

Openness, or open-endedness, refers to the unpredictable nature of real-life activities and events (Gasser, 1991). Traditional requirements analysis relies heavily on the modelling of *processes*. Through activity diagrams, event sequence charts, etc., one describes or prescribes what would or should happen under various known conditions. Real-life, however, is more chaotic (Suchman, 1987). When there are many parts of the system and environment over which one has little control or knowledge, it is hard to anticipate all contingencies and be able to know in advance what responses are appropriate. For example, in supply chain management, unexpected events can cause delays, wastages, and loss of revenue or customers. Information systems need to be flexible enough to deal with these contingencies. Process models that pre-determine all the steps are likely to be poor approximations of reality. Dealing with the dynamics of systems and interactions with their environments is therefore essential to requirements engineering. The challenge is to be able to describe and prescribe processes despite openness and uncertainties.

6) Cooperation

When authority and control are distributed, systems and environments are not merely interacting with each other. The vocabulary of interaction is inadequate for conveying the higher-level association that is needed to get disparate and autonomous parts to produce successful joint results. The idiom of cooperation is more appropriate as it implies the need for mutual agreement. Agreement may be achieved through recognition of mutual benefits arising from the cooperation. The potential for successful cooperation may be assessed through the analysis of the goals and beliefs of the two parties. Requirements techniques are needed to support the analysis of various aspects of cooperation, including synergy and conflict among goals, how to discover shared goals, and even how goals may be changed (Jarke & Kethers, 1999).

7) Boundaries, Locality, and Identity

Pervasive connectivity is removing or reducing barriers to the physical flow of information. However, there are many other reasons to set boundaries and even to re-erect barriers, for example, to assert local authority and autonomy. In healthcare, it is technically feasible to have fully integrated systems that allow any system or user to access information from any other system. Yet there are needs to set up appropriate boundaries due to privacy and other concerns, which must be considered together with legal, economic, administrative, and technical issues. There are boundaries that delineate extents of knowledge about the external world, and the reach of control. Requirements models and techniques need to be able to represent and analyze various kinds of boundaries from the very physical to the various forms of "logical" boundaries and identities.

3 The Agent as a Modelling Construct

Current modelling notations and languages that focus primarily on objects and activities are inadequate for coping with the new challenges and complexities. Recent research in modelling and analysis has begun to address the intentional and social dimensions (Mylopoulos, 1998). Concepts such as goal and agent are increasingly used in requirements frameworks. Based on the factors identified in section 2, we aim to formulate a notion of agent that can serve as a central construct in an agent-oriented approach to modelling and analysis. We propose that the agent as a modelling construct should have the following properties: intentionality, autonomy, sociality, contingent identity and boundaries, strategic reflectivity, and rational self-interest. To make the discussion concrete, we may think in terms of the development of a meeting scheduling system to support a distributed project team. We consider how agent-oriented modelling goes beyond conventional system analysis techniques.

In the remainder of this section, the term "agent" refers to the modelling construct, as opposed to, for example, software agents as a concrete artifact. The agent concept would be used to model phenomena involving hardware, software, and humans, some of which are inside, others outside, of the system developer's control. Some of these properties have counterparts in the software artifact conception of agent, but they have different connotations when applied to the requirements modelling context, as outlined in the following.

3.1 Intentionality

1. Agents are intentional.

Intentional concepts such as goals, beliefs, abilities, commitments, etc., provide a higher-level characterization of behaviour. One can characterize an agent in terms of its intentional properties without having to know its specific actions in terms of processes and steps. Explicit representation of goals allows motivations and rationales to be expressed. They allow "why" questions to be asked and answered. Beliefs provide for the possibility that an agent can be wrong in its assumptions about the world, and mechanisms to support revisions to those assumptions.

Because they are high-level, intentional descriptions may not be sufficient as specification for system construction. However, despite the high-level abstraction, they may already contain the kinds of information that are needed to distinguish important alternatives at initial stages of system definition. For example, a meeting scheduling system may be described as having the ability to set up meetings with designated participants. For the system to function, it requires commitment from invitees to respond with their available time slots. This description, with the implication (through a hierarchy of goals and sub-goals) that the system has the ability to come up with agreeable meeting times, distinguishes it from a calendar management system that only records meeting times arranged without assistance from the system. Starting from this initial functional characterization, one would also be led to probe the non-functional requirements, such as how quickly and reliably the system is expected to come up with meeting times that are desirable for various participants, and thus compare its viability and vulnerabilities with the less capable calendar management system.

Such analysis, suggested as crucial in section 2, can be done well before non-intentional specifics (such as sequences of message exchanges among participants and their systems) are elaborated on. A comparable example from the real-time domain is in a telecom system that needs to assemble and coordinate resources and capabilities to respond to a multi-media service request.

2. Agent intentionality is externally attributed by the modeller.

To say that an agent has intentionality is not to say that it has implementational mechanisms that make it intentional. From a modelling point of view, intentionality may be attributed to some entity if the modeller feels that the intentional characterization offers a useful way for describing and analyzing that entity. For example, some entity that is treated as an agent during modelling may end up being implemented in software that has no explicit representation and manipulation of goals, etc.

3. Agency provides localization of intentionality.

Multi-agent modelling allows different goals, beliefs, abilities, etc., to be attributed to different agents. An agent can be thought of as a locality for intentionality. Instead of having a single global collection of goals, belief, etc., these are allocated to separate agents. The agent concept provides a local scope for reconciling and making tradeoffs among competing intentionality, such as conflicting goals and inconsistent beliefs.

4. Agents can relate to each other at an intentional level.

The intentionality of agents are insulated from each other to a large extent, due to the assumption of autonomy. Nevertheless, they do influence and constrain each other. Relationships amongst agents can be characterized at an intentional level, without being reduced to non-intentional interactions. For example, one can say that the meeting scheduler depends on invitees to respond quickly (without having to say what message exchanges take place along what channels), in order for the meeting scheduling to be effective. Note that it is possible to have intentional relationships among agents that have no direct physical interactions (actions and their responses). For example, interactions may be mediated through other agents, or, there may be pre-established understanding among the agents.

3.2 Autonomy

1. An agent has its own initiative, and can act independently. Consequently, for a modeller and from the viewpoint of other agents, its behaviour is not fully predictable. Agents are not fully knowable, nor fully controllable.

The agent concept provides a way of accounting for the uncontrollable, unknowable, and unpredictable. In a modelling context, the presumption is that the behaviour of some piece of the system or environment is unknown until (or unless) it is specified. Unlike in the construction of artificial agents, the requirements analyst is not burdened with the task of generating the behaviour. The analyst merely has to characterize it and use that characterization to arrive at some conclusions or decisions. In traditional requirements modelling, incompleteness of knowledge about the behaviour of the system and of the relevant environment presents serious impediments to analysis. The new realities outlined in section 2 suggest that one needs to be able to do analysis despite incomplete knowledge.

2. The behaviour of an agent can be partially characterized, despite autonomy, using intentional concepts.

Openness of behaviour can be circumscribed by identifying where the freedoms of action lie, by defining the space of possible actions. It is possible to chart the areas of freedom through intentional modelling. The modeller can use goals and methods for achieving goals to demarcate spaces of possible actions for agents to take. Some courses of actions may lead to goal achievement, while others do not. Because of autonomy, agents are free to undertake whatever course of action they choose, and to live with the associated consequences. Intentional models therefore allow the idea of choice and freedom to be expressed. Stating something as a goal implies that there can be different ways for achieving it. Specifying one particular way restricts the choice and narrows the range of freedoms.

3.3 Sociality

1. An agent is characterized by its relationships with other agents, and not by its intrinsic properties alone.

It is by now widely recognized that requirements have to do with relationships between a system and its environment, and not a characterization of a system in isolation. Given the distributed nature of systems today, they are better characterized as being interspersed among environmental elements. System and environment can be treated as networks of agents with interdependencies among them. By adopting intentional modelling, these networks of dependencies can be modelled and reasoned about at a high level of abstraction.

2. Relationships among agents are complex and generally not reducible.

Social agents typically participate in multiple relationships, with a number of other agents, at the same time or at different times. In mechanistic systems as portrayed in most traditional requirements models, relationships are narrowly focused around intended functions. Agent-oriented modelling entertains a complexity of relationships similar to those in human organizations and societies. Requirements arise from many sources. They are inherently diverse and may be based on incompatible conceptual vocabularies, meanings, and value systems.

3. Conflicts among many of the relationships that an agent participates in are not easily resolvable.

There may be conflicts or potential conflicts arising from the multiple relationships that an agent engages in. In traditional approaches, competing demands need to be reconciled in order for requirements to be defined, then frozen for system development and implementation. In a more fluid and open environment, the demands of various agents may keep changing and may not be fully knowable. Agents may also build new relationships with other agents and dissolve existing ones. The management of conflicts is an ongoing one. Therefore it becomes necessary to maintain an explicit representation of the competing interests and their conflicts.

4. Agents tend to have multi-lateral relationships, rather than one-way relationships.

For example, social agents may have reciprocal dependencies and expectations on each other. Agent A can expect agent B to deliver on a commitment because B has goals and interests that A can help fulfil or meet. Reciprocity can be indirect, mediated via other agents. In general, social relationships exist as networks and patterns of relationships that involve multi-lateral dependencies. In mechanistic artificial systems, where one designer oversees interaction among parts, it is more common to see master-slave relationships that go one-way.

5. Agent relationships form an unbounded network.

There are no inherent limits on how far the impact of dependencies may travel in a network of agents. In considering the impact of changes (e.g., the introduction of computerized meeting scheduling), one may ask: Who else would be affected? Who will benefit, who will be hurt? Who can help me improve my position? These questions may lead to the discovery of agents not previously considered.

6. Cooperation among agents cannot be taken for granted.

Because agents are autonomous, the likelihood of successful cooperation is contingent upon many factors. Cooperative arrangements may not be stable. In mechanistic systems, collaboration or even integration of systems is usually not regarded as problematic, as the systems would not "resist" since they do not have self-initiative. It is a technical matter to be accomplished through the ingenuity and competence of the designer in charge of the systems. When autonomous agents are involved, an important part of the analysis is to determine the viability of cooperative arrangements and dependencies.

7. Autonomy is tempered by sociality.

Given autonomy, an agent can behave in totally arbitrary ways. However, an agent that exists within a social network of expectations and commitments has behaviour that are confined by them. The agent can still violate them, but will suffer the consequences. The behaviour of a socially situated agent is therefore largely predictable, although not in a precise or minute way. By the same token, a social agent is partially knowable and controllable due to its social connected-ness.

3.4 Identity and Boundary

1. There can be abstract agents, as well as physical agents.

A concept of agent for requirements modelling should not be tied to that of a physical agent. An abstract agent is an entity that exhibits coherent behaviour, according to the judgement of the modeller. The need to distinguish essential or "logical" aspects of a system from physical aspects has long been a basic tenet of requirements analysis. Agent-oriented modelling should allow agents along a range of physicality and abstractness to be described, as well as relationships among them. Social agents frequently create new abstractions such as teams, roles within teams, positions within organizations, etc., to help define each others responsibilities and expectations, and to guide action.

2. The boundaries, and thus the identity, of an agent are contingent and changeable.

The external relationships of an agent also serve as its conceptual boundaries, and they are not fixed a priori. For example, when a task is re-allocated from one agent to another, the boundary of responsibility shifts. Various notions of boundary need to be expressible, as well as changes in the boundaries.

3.5 Strategic Reflectivity

1. Agents can reflect upon their own operations.

Requirements analysis is a reflective process. It is reflective in that the analyst is detached from the routine operation of the system and its environment, treats it as a subject of analysis, reasons about it, passes judgement on it, redefines and alters it with the aim of improving it. An agent can therefore appear in the "operational world" that is being modelled, as well as participate in the requirements analysis and decision making process, in the "development world". When an agent enters the development world, it is engaging in reflection. It reasons about its own condition in the operational world, and proposes changes to it. To support the requirements process, both the agent as it appears in the operational world, and in the development world, need to be modelled.

2. Development world deliberations and decisions are usually strategic with respect to the operational world.

The introduction or modification of systems alters the strategic relationships among agents in the environment. Some agents will gain while others may lose -- in terms of capabilities, power, or even just convenience. To understand the forces at work, one needs to be able to model and analyze strategic interests. When a new system is defined, it alters the space of possible actions for each agent. It creates a new set of opportunities and constraints. The freedoms that agents have are realigned. Therefore decisions and actions in the development world set the stage for decisions and actions in the operational world. It is therefore important, during requirements analysis, to determine opportunities and vulnerabilities in various proposals and configurations, to analyze the outcomes from the perspectives of the strategic interests of stakeholders.

3.6 Rational Self-interest

1. An agent strives to meet its goals.

An agent is presumed to make decisions about what actions to take in order to best serve its interests. This applies to both the operational world and the development world. The agent construct serves as a focal point for capturing self-interest. The assumption of rational self-interest provides a convenient idealization for characterizing agents whose behaviour are otherwise unpredictable, because of autonomy, and may therefore appear to be incoherent.

2. Self-interest is in a context of social relations.

Self-interest does not mean an agent can act without regard for others. This is due to its social relationships with other agents. For example, it is usually in the agent's interest to live up to its commitments.

3. Rationality is bounded and partial.

The modeller attributes rationality to the agent in order to draw inferences about its behaviour. However, the inferences are limited by incomplete knowledge and bounded resources. The agent construct provides for the exercising of rationality within a local scope. The notion of abstract agent is a way of creating new scopes of rationality that transcend physical ones.

4 *i** - An Agent-Oriented Modelling Framework

i* (Yu, 1997, 1995) is an agent-oriented modelling framework motivated by some of the considerations in the preceding sections. While it does not address all the issues in full, it is briefly reviewed here as an illustration of a possible first step towards an agent-oriented modeling paradigm. References to concepts in Section 3 are given where appropriate.

In ***i****, the central modelling construct is that of the intentional actor. It has intentional properties such as goals, beliefs, abilities, and commitments (Section 3.1). Actors are autonomous, and are not fully knowable or controllable (Section 3.2). They are externally characterized in terms of their relationships with other actors. Actors depend on each other for goals to be achieved, tasks to be performed, and resources to be furnished. By depending on others, an actor may be able to achieve goals that are difficult or impossible to achieve on its own (Section 3.3). On the other hand, an actor becomes vulnerable if the depended-on actors do not deliver. Actors are strategic in the sense that they are concerned about opportunities and vulnerabilities, and seek rearrangements of their environments that would better serve their interests (Section 3.5).

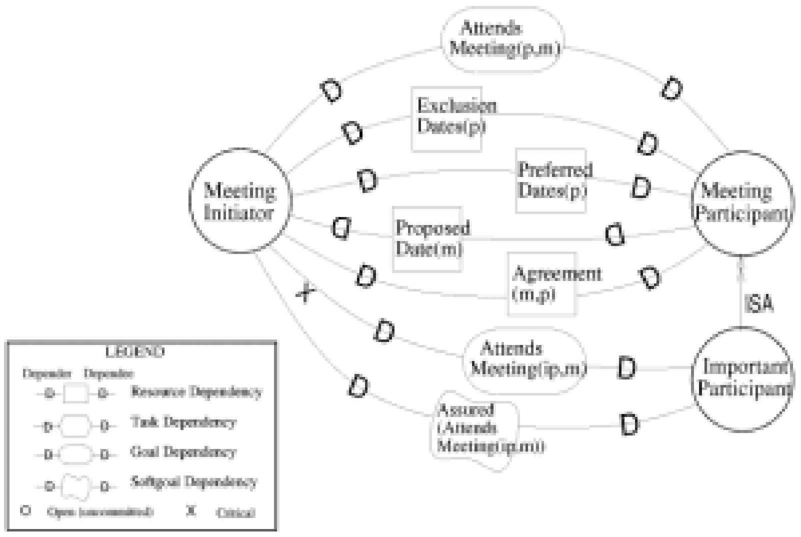

Figure 1. Strategic Dependency model for meeting scheduling, without computer-based scheduler

A small example concerning a meeting scheduling system will be used to illustrate. Figure 1 shows a Strategic Dependency (SD) model for meeting scheduling, without computer-based support. The meeting initiator depends on meeting participants p to attend meeting m. The dependency is intentional (Section 3.1) in that if some participant does not attend the meeting, the meeting initiator may fail to achieve some goal (not made explicit in the SD model), or at least not succeed to the degree desired. This is the reason for wanting to schedule the meeting in advance. To schedule meetings, the initiator depends on participants to provide information about their availability -- in terms of a set of exclusion dates and preferred dates. (For simplicity, we do not separately consider time of day or location.) To arrive at an agreeable date, participants depend on the initiator for date proposals. Once proposed, the initiator depends on participants to indicate whether they agree with the date. For important participants, the meeting initiator depends critically (marked with an ``X" in the graphical notation) on their attendance, and thus also on their assurance that they will attend.

The kinds of freedom and constraints (Section 3.2) are indicated by the types of dependency between dependers and dependees. The meeting initiator's dependency on participant's attendance at the meeting (`AttendsMeeting(p,m)`) is modelled as a *goal dependency*. This means that it is up to the participant has the freedom to decide how to attain that goal. An agreement on a proposed date `Agreement(m,p)` is modelled as a *resource dependency*. This means that the participant is expected only to give an agreement. If there is no agreement, it is the initiator who has to find other dates (do problem solving). For an important participant, the initiator critically depends on that participant's presence. The initiator wants the latter's attendance to be assured (`Assured[AttendsMeeting(p,m)]`). This is modelled as a *softgoal dependency*.
It is up to the depender to decide what measures are enough for him to be assured, e.g., a telephone confirmation. A softgoal is a goal whose criteria for satisfaction is not sharply defined a priori, and is subject to a "satisficing" mode of reasoning by the stakeholders (Simon, 1996). The softgoal concept is adapted from a framework for dealing with non-functional requirements in software engineering (Chung, 1993, Chung et al, 2000). These types of dependency relationships between intentional actors cannot be expressed or distinguished in non-intentional models that are used in most existing requirements modelling frameworks.

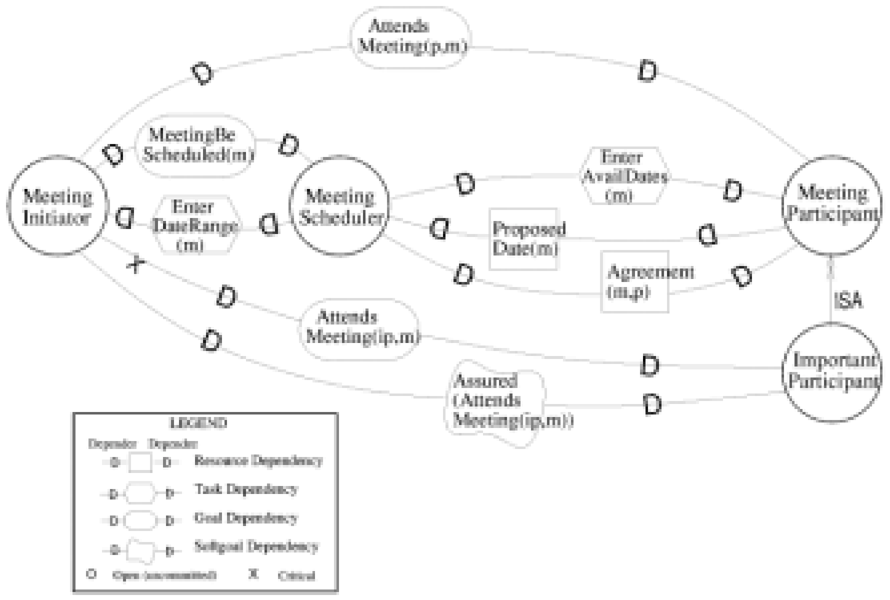

Figure 2. Strategic Dependency model for meeting scheduling with computer-based scheduler

Now consider how the introduction of a Meeting Scheduling System might change the strategic relationships (Figure 2). The meeting initiator delegates much of the work of meeting scheduling to the meeting scheduler. The initiator no longer needs to be bothered with collecting availability information from participants, or to obtain agreements about proposed dates from them. The meeting scheduler also determines what are the acceptable dates, given the availability information. The meeting initiator does not care how the scheduler does this, as longer as the acceptable dates are found. This is reflected in the goal dependency of `MeetingBeScheduled` from the initiator to the scheduler. The scheduler expects the meeting initiator to enter the date range by following a specific procedure. This is modelled via a *task dependency*., indicating a restriction on the initiator's freedom of action (Section 3.2 - freedom and constraints).

Note that it is still the meeting initiator who depends on participants to attend the meeting.

It is the meeting initiator (not the meeting scheduler) who has a stake in having participants attend the meeting. Assurance from important participants that they will attend the meeting is therefore not delegated to the scheduler, but retained as a dependency from `MeetingInitiator` to `ImportantParticipant`.

The SD model models the meeting scheduling process in terms of intentional relationships among agents, instead of the flow of entities among activities. This allows analysis of opportunity and vulnerability. For example, the ability of a computer-based meeting scheduler to achieve the goal of `MeetingBeScheduled` represents an opportunity for the meeting initiator not to have to achieve this goal himself. On the other hand, the meeting initiator would become vulnerable to the failure of the meeting scheduler in achieving this goal. (Section 3.5 - strategic opportunities and vulnerabilities).

The Strategic Dependency model provides an important level of abstraction for describing systems in relation to their environments, in terms of intentional relationships among them. This allows the modeller to understand and analyze new or existing organizational and systems configurations even if internal goals and beliefs are not known.

When actors reflect upon the merits or drawbacks of various alternative configurations, their strategic interests and concerns are made known to the modeller, and are therefore accessible for analysis (Section 3.5 - strategic reflectivity). In the ***i**** framework, the Strategic Rationale (SR) model provides a more detailed level of modelling by looking ``inside" actors to model internal intentional structures and relationships. Intentional elements (goals, tasks, resources, and softgoals) appear in the SR model not only as external dependencies, but also as internal elements linked by means-ends relationships and task-decompositions (Figure 3). The graphs show how the rational pursuit of self-interest by each actor can be modeled and analyzed (Section 3.6).

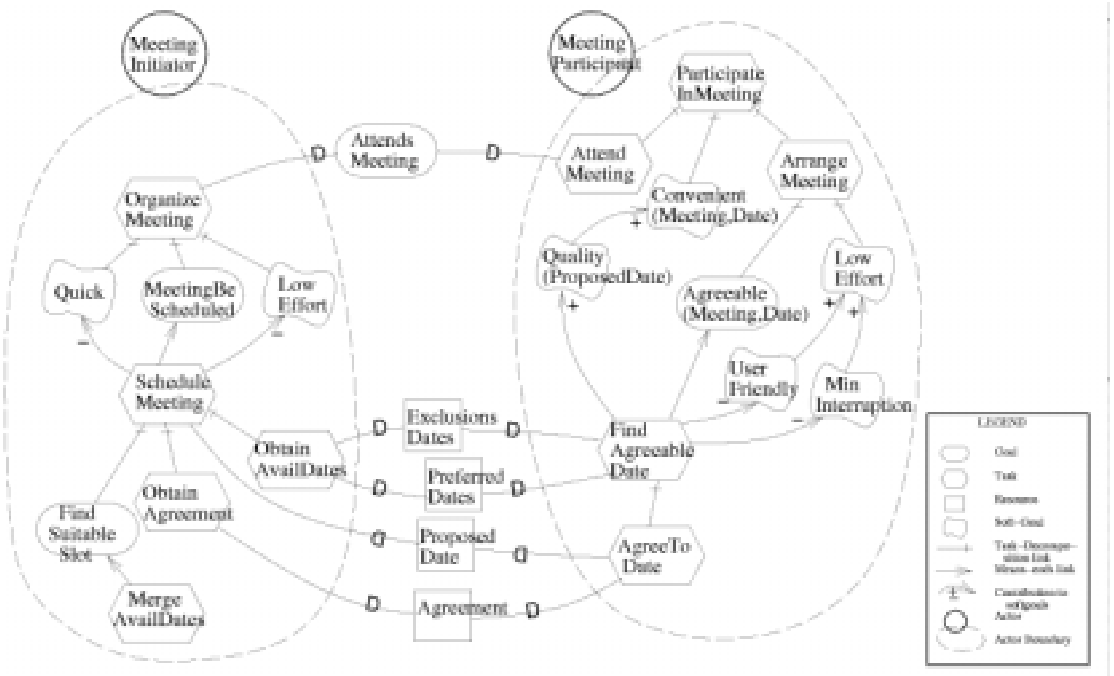

Figure 3. Strategic Rationale model for meeting scheduling, before considering computer-based meeting scheduler

For example, for the meeting initiator, an internal goal is that of `MeetingBeScheduled`. This goal can be met (represented via a means-ends link) by scheduling meetings in a certain way (a task), consisting of (represented via task-decomposition links): obtaining availability dates from participants, finding a suitable date (and time) slot, proposing a meeting date, and obtaining agreement from the participants.

These elements of the `ScheduleMeeting` task are represented as subgoals, subtasks, or resources depending on the type of freedom of choice as to how to accomplish them (analogous to the SD model). Thus `FindSuitableSlot`, being a subgoal, indicates that it can be achieved in different ways. On the other hand, `ObtainAvailDates` and `ObtainAgreement` refer to specific ways of accomplishing these tasks. Similarly, `MeetingBeScheduled`, being represented as a goal, indicates that the meeting initiator believes that there can be more than one way to achieve it (Figure 4).

`MeetingBeScheduled` is itself an element of the higher-level task of organizing a meeting. Other subgoals under that task might include equipment be ordered, or that reminders be sent (not shown). This task has two additional elements which specify that the organizing of meetings should be done quickly and not involve inordinate amounts of effort. These qualitative criteria are modelled as softgoals. These would be used to evaluate (and also to help identify) alternative means for achieving ends. In this example, we note that the existing way of scheduling meetings is viewed as contributing negatively towards the `Quick` and `LowEffort` softgoals.

On the side of the meeting participants, they are expected to do their part in arranging the meeting, and then to attend the meeting. For the participant, arranging the meeting consists primarily of arriving at an agreeable date. This requires them to supply availability information to the meeting initiator, and then to agree to the proposed dates. Participants want selected meeting times to be convenient, and want meeting arranging activities not to present too many interruptions.

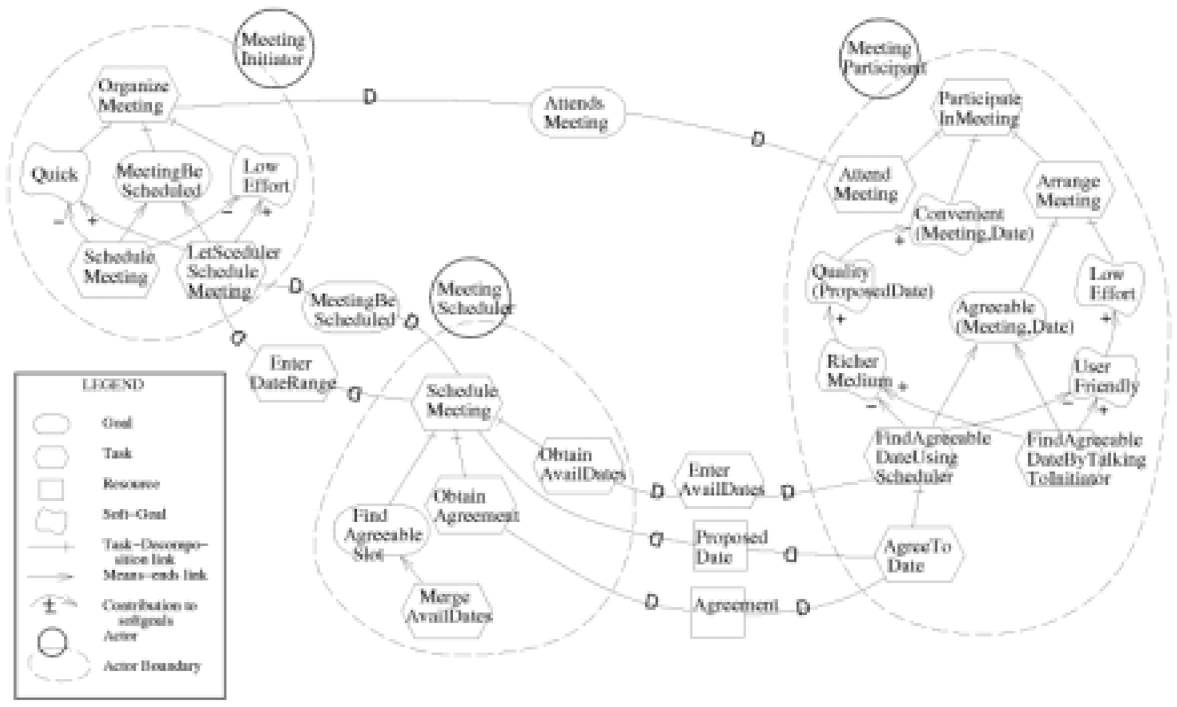

Figure 4. Strategic Rationale model for a computer-supported meeting scheduling configuration

The SR model thus provides a way of modelling stakeholder interests, and how they might be met. Stakeholders evaluate various alternatives with respect to their interests. *Task-decomposition* links provide a hierarchical description of intentional elements that make up a routine. The *means-ends* links in the SR provides understanding about *why* an actor would engage in some tasks, pursue a goal, need a resource, or want a softgoal. From the softgoals, one can tell *why* one alternative may be chosen over others. For example, availability information in the form of exclusion sets and preferred sets are collected so as to minimize the number of rounds and thus to minimize interruption to participants.

The ***i**** models offer a number of levels of analysis, in terms of ability, workability, viability and believability. These are beyond the scope of this paper and are discussed further in (Yu, 1997, 1995). The term agent is used in ***i**** to refer to physical actors, while actor is used as the more generic term. A role is an abstract actor, and a position is a collection of roles that are typically covered by a (physical) agent (Yu & Mylopoulos, 1994).

The ***i**** framework has been incorporated into a number of recent modelling frameworks and methodologies (Briand et al., 1998, Jarke & Kethers, 1999, Petit, 2000, Castro et al., 2001, Perini et al., 2001).

5 Discussion and Conclusions

In this paper, we have argued for the need to develop agent orientation as a modelling paradigm, quite separately from the use of agents as a software technology. The nature of systems and the characteristics of their environments have changed. A more flexible, higher-level set of constructs are needed to deal with a world operating more on social principles than on mechanistic rules. We have identified a number of properties that are desirable for a concept of agent that is suitable for the purpose of modelling and analysis. The list of properties is not claimed to be in any sense exhaustive or complete. It is rather intended to consolidate and to elaborate on the directions along which an agent-oriented paradigm might develop. Some of the properties may appear similar to those of software agents, but we have argued that they need to be reinterpreted and reformulated to suit the context and needs of modeling.

Agent orientation for modelling and requirements engineering is a relatively recent development. There has been a number of frameworks and approaches each with different emphases and orientations. The KAOS framework (Dardenne 1993, van Lamsweerde 2000) offers a methodical approach for goal-oriented requirements engineering. The agent is an essential construct, following the earlier work on Composite Systems Design (Feather 1987, Feather et al. 1991). The formal framework provides a strong foundation for goal-based reasoning and analysis in agent-oriented modelling. Goals are fully reduced to non-intentional operations by the time they are assigned as responsibilities to agents. The openness of agent actions is therefore not considered when generating or evaluating alternatives. Agents interact with each other non-intentionally, so they do not have rich social relationships.

The Albert II language (Du Bois 1995) is an agent-oriented language for expressing requirements for real-time systems. Agents have limited knowledge of each other, and have contractual obligations expressed in terms of internal and cooperation constraints. Agents are not intentional and do not have goals. The language focuses on specification and is not concerned with the examination of alternatives for meeting goals. CASL (Cognitive Agents Specification Language) (Shapiro & Lespérance 2000) and the underlying ConGolog language (Lespérance et al. 1999, Koubarakis & Plexousakis 2000) has been used to model multi-agent business processes. They are based on the situation calculus and emphasize verification and validation. Simulations can be performed despite incompleteness of state specifications. They do not consider openness of agent behaviour or intentional relationships among agents.

The EKD approach (Bubenko et al., 1998) to requirements engineering is goal-driven, with an emphasis on understanding the business objectives behind system requirements. Modelling is done in terms of six interconnected submodels: the Goal model, the Concepts model, the Business Rules model, the Business Process model, the Actors and Resources model, and the Technical Components and Requirements model. The approach contains many of the concepts needed for agent-oriented modelling, but does not explicitly deal with issues of agent autonomy and sociality.

Action-Workflow is a notation and method for modelling cooperative work (Medina-Mora et al., 1992). The basic unit of modelling is a workflow between a *customer* and a *performer*. The customer-performer relationship is characterized in terms of a four-phased loop, representing the stages of proposing, agreeing, performing, and accepting. Each phase involves different types of communication acts which can be analyzed using Speech Acts theory. This framework has a stronger orientation to deal with the social nature of agents, especially their reliance on commitments and the potential for breakdowns. Intentional structures such as goals or means-ends relationships are not explicitly represented, so there is no support for reflection or shifting boundaries of responsibilities.

Other related modelling and analysis frameworks and techniques include those for managing multiple viewpoints (Finkelstein & Sommerville, 1996), inconsistencies (Ghezzi & Nuseibeh, 1998) and for supporting traceability (Jarke, 1998) and negotiation (Robinson & Volkov 1999). All of these are relevant aspects for agent orientation. Some scenario-oriented methods also include agents in their approaches (Jarke & Kurki-Suonio, 1998). Agent-oriented extensions have been proposed for UML (Bauer et al., 2000), but the current emphasis is on the treatment of interaction protocols. Intentional concepts have not yet been incorporated. Methodologies for analysis and design in the multi-agent software community (e.g., Wooldridge et al, 2000) have been focusing on the software system and are not concerned with requirements per se.

The ***i**** framework is closest in spirit to the vision of agent-oriented modelling proposed in this paper, but it only begins to address the multiplicity of issues outlined. Some areas that are particularly lacking include temporal aspects, viewpoints and negotiation support, and support for limited forms of rationality. The connection between intentional models and the non-intentional world of objects and actions also need to be elaborated on. There have been work in this area, (e.g., Yu et al., 1995, Petit, 2000), but many open issues remain.

While the agent-oriented approach to requirements modelling and analysis can be used regardless of the type of software technology to be used eventually to construct the system, there are clearly advantages if the implementation is also based on agent-oriented concepts (see e.g., Wagner et al, 2000). Efforts are underway to develop an agent-oriented approach to software development that is guided by a concept of agent from the modelling level (Mylopoulos & Castro 2000).

Acknowledgements

Financial support from the Natural Sciences and Engineering Research Council of Canada, Communications and Information Technology Ontario, and Mitel Corporation are gratefully acknowledged.

References

Bauer, B., Müller, J.P., & Odell, J. (2000). An Extension of UML by Protocols for Multiagent Interaction. Proceedings of the Fourth International Conference on Multi-Agent Systems (ICMAS-2000), pp. 207-214.

Bobrow, D.G. (1991) Dimensions of Interaction: AAAI-90 Presidential Address. AI Magazine 12(3): 64-80.

Briand, L., Kim, Y.-M., Melo, W., Seaman, C., & Basili, V. (1998). Q-MOPP: Qualitative Evaluation of Maintenance Organizations, Processes, and Products. Software Maintenance: Research and Practice, (10):249-278.

Bubenko, J. (1995). Challenges in Requirements Engineering. 2nd Int. Symp. on Requirements Engineering. March 1995, York, England.

Bubenko, J., Brash, D., & Stirna, J. (1998). EKD User Guide. Available at ftp://ftp.dsv.su.se/users/js/ekd_user_guide.pdf

Castro, J. , Kolp, M., & Mylopoulos J. (2001) A Requirements-Driven Development Methodology. Proceedings of the Thirteenth Conference on Advanced Information Systems Engineering. Interlaken, Switzerland. June 2001. Springer-Verlag. LNCS2068. 108-123.

Chung, K.L. (1993) "Representing and Using Non-Functional Requirements: A Process-Oriented Approach," Ph.D. thesis, Univ. of Toronto, 1993.

Chung, L., Nixon, B.A., Yu, E., & Mylopoulos, J. (2000) Non-Functional Requirements in Software Engineering. Kluwer Academic Publishers.

Dardenne, A., van Lamsweerde, A., & Fickas, S., (1993) Goal-Directed Requirements Acquisition, Science of Computer Programming. 20(1-2): 3-50.

Du Bois, P. (1995) The Albert II Language. On the Design and the Use of a Formal Specification Language for Requirements Analysis. Ph.D. thesis. University of Namur, Belgium.

Feather. M.S. (1987) Language support for the specification and development of composite systems. ACM Transactions on Programming Languages and Systems , 9(2):198-234.

Feather, M.S., Fickas, S.F., Helm, B.R., (1991) Composite system design: the good news and the bad news, Proceedings of Fourth Annual KBSE Conference, Syracuse, 1991, pp. 16-25.

Finkelstein, A. & Sommerville, I. (1996). The Viewpoints FAQ: Editorial - Viewpoints in Requirements Engineering. IEE Software Engineering Journal, 11(1): 2-4.

Gasser, L. (1991) Social Conceptions of Knowledge and Action: DAI Foundations and Open Systems Semantics. Artificial Intelligence. 47(1-3): 107-138.

Ghezzi, C. & Nuseibeh, B. (1998). Guest Editorial - Managing Inconisstency in Software Development. IEEE Transactions on Software Engineering 24(11): 906-907.

Jarke, M. & Kurki-Suonio, R. (1998). Guest Editorial - Special Issue on Scenario Management. IEEE Transactions on Software Engineering, 24(12): 1033 -1035.

Jarke, M. (1998) Requirements Tracing - Introduction. Communications of the ACM, (41)12: 32-36.

Jarke, M., & Kethers, S.. (1999) Regionale Kooperationskompetenz: Probleme und Modellierungstechniken. Wirtschaftsinformatik 4/99, pp. 316-325 (in German).

Jennings, N.R., Sycara, K., & Wooldridge, M. (1998). A Roadmap of Agent Research and Development. Autonomous Agents and Multi-Agent Systems, 1, 7-38.

Koubarakis, M., and Plexousakis, D. (2000). A Formal Model for Business Process Modeling and Design. CAiSE 2000. pp. 142-156.

Lespérance, Y., Kelley, Todd G., Mylopoulos, John, & Yu, Eric S. K. (1999). Modeling Dynamic Domains with ConGolog. Proc. of Conf. on Advanced Information Systems Engineering CAiSE 1999 : 365-380.

Medina-Mora, R., T. Winograd, R. Flores, & F. Flores (1992) The Action Workflow Approach to Workflow Management Technology. Proc. CSCW 1992: 281-288.

Mylopoulos, J. (1998) Information Modeling in the Time of the Revolution. Information Systems 23(3-4): 127-155.

Mylopoulos, J., Chung, L., & Yu, E.. (1999) From Object-Oriented to Goal-Oriented Requirements Analysis. CACM 42(1): 31-37.

Mylopoulos, J. & Castro, J.. (2000) Tropos: A Framework for Requirements-Driven Software Development In J. Brinkkemper and A. Solvberg (eds.), Information Systems Engineering: State of the Art and Research Themes, Lecture Notes in Computer Science, Springer-Verlag, pp. 261-273, June 2000.

Newell, A. (1982) The Knowledge Level. Artificial Intelligence 18: 87-127.

Nuseibeh, B. A. & Easterbrook, S. M.. (2000) Requirements Engineering: A Roadmap. Proceedings, 22nd International Conference on Software Engineering (ICSE'00), Limerick, Ireland, 5-9 June, 2000. IEEE Computer Society Press.

Perini, A., Giunchiglia, F., Mylopoulos, J., Bresciani, P., & Giorgini, P. (2001) A Knowledge Level Software Engineering Methodology for Agent-Oriented Programming. Proceedings of the Fifth International Conference on Autonomous Agents, Montreal, Canada, May, 2001. ACM Press.

Petit, M. (2000) A Multi-formalism Component-Based Approach to Manufacturing Systems Modeling, Ph.D. thesis, University of Namur, Belgium.

Robinson, W.N., & Volkov, S. (1998) Supporting the Negotiation Life-Cycle. Communications of the ACM, 41(5): 95-102.

Shapiro, S. & Lespérance, Y. (2001) Modeling Multiagent Systems with the Cognitive Agents Specification Language - A Feature Interaction Resolution Application. To appear in Castelfranchi, C. and Lespérance, Y., editors, Intelligent Agents Volume VII - Proceedings of the 2000 Workshop on Agent Theories, Architectures, and Languages (ATAL-2000), LNAI, Springer-Verlag, Berlin, 2001.

Shaw, M. (2000) Sufficient Correctness and Homeostasis in Open Resource Coalitions: How Much Can You Trust Your Software System? Proceedings of the 4th International Software Architecture Workshop (ISAW-4), affiliated with the 22nd International Conference on Software Engineering (ICSE 2000), Limerick, Ireland, June, 2000.

Simon, Herbert A. (1969). The Sciences of the Artificial. MIT Press.

Suchman (1987). Plans and Situated Actions: The Problem of Human-Machine Communication. Cambridge University Press.

van Alstyne, M. (1997) The State of Network Organization: A Survey in Three Frameworks. Journal of Organizational Computing & Electronic Commerce, 7(3) 83-151.

van Lamsweerde, A. (2000) Requirements Engineering in the Year 2000: A Research Perspective. Proc. Int. Conf. on Software Engineering, June 2000, Limerick, Ireland.

Wagner, G., Lesperance, Y. & Yu, E., eds. (2000) Agent-Oriented Information Systems 2000: Proceedings of the 2nd International Workshop at CAiSE*00, Stockholm, June 2000. iCue Publishing, Berlin. ISBN 3-8311-0093-4. See also http://aois.org.

Wegner, P. (1997) Why Interaction Is More Powerful Than Algorithms, Communications of the ACM., 40(5): 80-91. May 1997.

Wooldridge, M., Jennings, N. R., & Kinny, D. (2000) The Gaia Methodology for Agent-Oriented Analysis and Design. Journal of Autonomous Agents and Multi-Agent Systems 3 (3) 285-312.

Yu, E. (1997) Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering' Proceedings of the 3rd IEEE Int. Symp. on Requirements Engineering (RE'97) Jan. 6-8, 1997, pp. 226-235.

Yu, E. (1995) Modelling Strategic Relationships for Business Process Reengineering. Ph.D. thesis. Dept. of Computer Science, University of Toronto.

Yu, E. & Mylopoulos, J. (1994) Understanding ``Why'' in Software Process Modelling, Analysis, and Design' Proceedings of 16th International Conference on Software Engineering, May 16-21, 1994, Sorrento, Italy, pp. 159-168.