Social Abstractions for Information Agents

Munindar P. Singh
Department of Computer Science
North Carolina State University
Raleigh, NC 27695-7534, USA
singh@ncsu.edu

Michael N. Huhns
Department of Electrical and Computer Engineering
University of South Carolina
Columbia, SC 29208, USA
huhns@sc.edu

1 Introduction

Most of the modern applications of computing technology operate in large-scale, open, information-rich environments. Open environments are distinguished by fact of having a number of networked and interrelated, but heterogeneous, information resources. The applications include ubiquitous information access, electronic commerce, virtual enterprises, logistics, and sensor integration, to name but a few. These applications differ from conventional database applications not only in the nature and variety of information they involve, but also in including a significant component that is beyond the information system *per se*: the creation, transformation, use, and ultimate fate of information. Early work on information-rich environments naturally enough focused on infrastructural issues, but as those issues have come to be resolved, attention is increasing on the high-level aspects more naturally addressed in an agent context. We define *information agents* as the agents that exist and act in such environments. We show that this definition is more general than typically studied, and we show how systems of multiple information agents can be specified and built from the social level.

Even with information agents, previous work has focused on some of the simpler aspects of distributed computing and databases. Many of the features that make agents a worthwhile metaphor for computer science have not been deeply investigated yet. These features include the sociability of agents, which is the theme of the present paper. We describe some social abstractions for agents that are centered around the notion of social commitments. Social commitments have garnered much attention in the literature on theoretical aspects of multiagent systems. Here we show how they can be part of a practical architecture for information-rich environments, and can be applied to a range of interesting problems. We discuss two examples on electronic commerce and virtual enterprises, which were prototyped, albeit in a rather simple manner, using the above abstractions. We also discuss a reconstruction of the MINDS system, one of the first adaptive multiagent systems applied to the problem of building a referral network for cooperative information access.

Organization

Section 2 introduces the key themes in information-rich environments and motivates a notion of coherence-preserving information agents. Section 3 describes social abstractions centered around the notion of social commitments. Section 4 discusses the problem of interoperation in cooperative information systems and shows how commitments may be used to specify the requirements for interoperation. Section 5 applies these notions to applications in referral networks, electronic commerce, and virtual enterprises. Section 6 concludes with a discussion of future directions.

2 Key Concepts

In order to discuss social abstractions for information agents, we first discuss information-rich environments and then some important aspects of information agents in which our approach differs from more conventional approaches.

2.1 Information-Rich Environments

Information-rich environments have been around for a long time. We previously defined them in broad terms as environments consisting of a large number and variety of distributed and heterogeneous information sources [6]. The associated applications are varied. They involve the purely informational ones, such as database access, information malls, workflow management, electronic commerce, and virtual enterprises. They also include the information component of physical applications, such as distributed sensing, manufacturing, transportation, energy distribution, and telecommunications. Information-rich environments have the following properties. They

- Span enterprise boundaries
- Include heterogeneous components
- Comprise information resources that can be added or removed in a loosely structured manner
- Lack global control of the accuracy of the contents of those resources
- Incorporate intricate interdependencies among their components.

Information-rich environments may often involve a significant physical or social component. However, they are amenable to specialized multiagent systems, termed *Cooperative Information Systems*, which are multiagent systems with organizational and database abstractions geared to open environments. A typical CIS, as shown in Figure 1, includes an environment consisting of a variety of information resources, coupled with some kind of a semantic directory or ontology for the domain of interest. The semantic directory contains descriptive information about the resources, including any constraints that apply to their joint behaviors. Each component of the environment, as well as its human user(s), is modeled as associated with an agent. The agents capture and enforce the requirements of their associated parties. They interact with one another appropriately, and help achieve the robustness and flexibility that is necessary for effective operation.

Information access involves finding, retrieving, and fusing information from a number of heterogeneous sources. At the level of abstraction that concerns CIS, we are not concerned with network connectivity or the formatting variations of data access languages. Rather, our concern is with the meaning of the information stored. It is possible, and indeed common, that when different databases store information on related topics, each provides a different model of it. The databases might use different terms, e.g., *employee* or *staff*, to refer to the same concept. Worse still, they might use the same term to have different meanings. For example, one database may use *employee* to mean anyone currently on the payroll, whereas another may use *employee* to mean anyone currently receiving benefits. The former will include assigned contractors; the latter will include retirees. Consequently, merging information meaningfully is nontrivial. The problem is exacerbated by advances in communication infrastructures and increases in competitive pressures, because different companies or divisions of a large company, which previously proceeded independently of one another, are now expected to have some linkage with each other.

The linkages can be thought of as semantic mappings between the application (which consumes or produces information) and the various databases. If the application somehow knows that *employee* from one database has a certain meaning, it can insert appropriate tests to eliminate the records it does not need. Clearly, this approach would be a nightmare to maintain: the slightest changes in a database would require modifying all the applications that consume its results!

2.2 Information Agents

Approaches based on agents and multiagent systems are natural for information-rich environments [8], but there are three major definitional matters that come up that must be resolved.

- The first is the quite routine one of the definition of agents. Numerous definitions of agents are known in the literature [2, 6, 11]. Indeed, the only agreement seems to be that there is a range of definitions! Some of the important properties of agents include autonomy, adaptability, and interactiveness, but exceptions reveal that an agent need not have these properties. However, we believe that agents *must* be capable of interacting with other agents at the social or communicative level [5, 6]. We distinguish social or communicative interactions from incidental interactions that agents might have as a consequence of existing and functioning in a shared

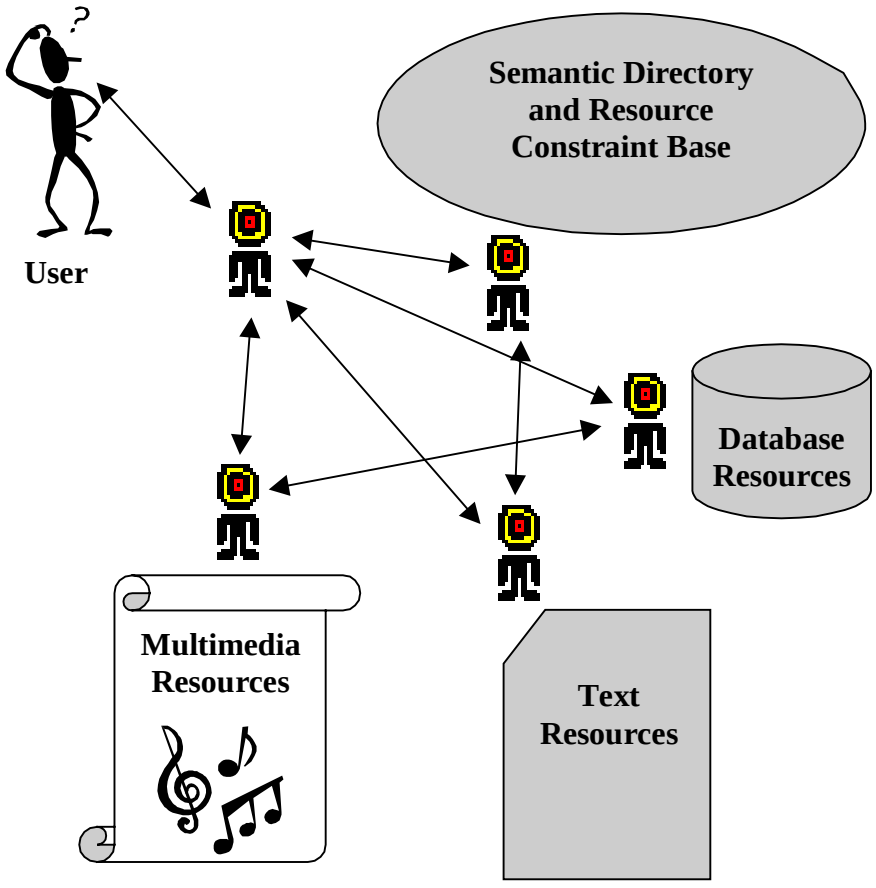


Figure 1: A schematic view of a cooperative information system

environment [7]. Independently, Wegner has also argued that interaction is a key extension beyond traditional computer science [14].[1]

- The second is the less contentious matter of the definition of information agents. Information agents are loosely defined as agents who help find information. In most work, they play the role primarily of accessing resources and retrieving information from them. However, a better view of information agents is as agents who manage information. This is not entirely inconsistent with current work. However, it is an important enough distinction and therefore emphasized here, even though the present book's subtitle mentions gathering rather than updating information. There are two main reasons why the ability of agents to make updates is significant:
  - The agents are applied in settings where the information they gather is fed into a larger process in the enterprise or virtual enterprise in which it functions. This happens, for example, in the case of applications that involve commerce among the various participants or the construction of artifacts, such as in configuring mechanical parts.
  - The agents must maintain knowledge about each other in order to coordinate successfully and to assist or compete with others as appropriate. Consequently, even if the agents only retrieve information that the user directly cares about, they may perform updates on their internal data structures. These "hidden" updates are extremely important for the effective and efficient performance of desired tasks. Their presence, however, changes the true model of the information access from mere retrieval to management.
- Applications in information-rich environments belie an important and common assumption of traditional information environments. Traditional environments are centered around databases (treated here not as arbitrary collections of data, but rather those that are built and accessed using a database management system (DBMS) [1]). Databases, through the transaction concept, are designed to ensure that the data they store remains consistent despite any number of concurrent users and despite any kinds of application failures [3]. Consistency is, of course, a desirable property and is essential in applications such as those involving banking.

  However, there is an implied assumption in the traditional work, which is that consistency is essential for all applications. It turns out that this assumption is readily falsified in information-rich environments for the following reasons:
  - The mutual consistency of autonomous information sources would usually be an unrealistic expectation. When information is to be accessed from several independent sources, the most we can hope for in general is that all of the sources and the partial results generated from them be relevant to the information request. Ideally, a smart information agent may be able to synthesize the partial results into a reasonable argument that supports or refutes some claim.
  - When the information is gathered in a multiagent system whose member agents maintain knowledge about each other, the choice of the agents with which an agent interacts is often not governed by their mutual consistency, but by the expected structure or suitability of the combination of the resulting answers.
  - When the information gatherers are part of a larger process, there may be additional organizational requirements. For example, if the agents are updating information sources, they might wish to coordinate these updates. More importantly, if the information sources are not updated by the agents but change anyway—due to other factors—the agents might still benefit from informing each other of those changes. For instance, an agent in electronic commerce may notify the agents of potential customers of the change in price or unexpected delays in availability.

  For the above reasons, we argue that a more applicable theme in open environments is *coherence*. This is not to suggest that consistency is undesirable. For example, in electronic commerce, orders should still be billed consistently. However, in cases of failures in the underlying information or exceptions arising due to unexpected events in the physical environment, the loss of consistency can be rendered harmless if coherence is preserved. Thus, consistency, while still useful, has a much reduced role in the kinds of applications that are the most important in open environments.

[1] As a practical instantiation of these requirements, an agent, when queried appropriately, should be able to state its name, age, and originator (or the equivalent).

3 Social Abstractions

We hinted above that the notion of process in information-rich environments is richer and more flexible than in conventional settings. Indeed, traditional abstractions such as database transactions are notoriously unsuited for representing and managing flexible processes.

Accordingly, we have been pursuing a research program termed *Interaction-Oriented Programming (IOP)* to develop and study primitives for the specification of systems of agents and constraints on their behavior. These primitives include societies, the roles agents may play in them, what capabilities and commitments they require and what authorities they grant. Agents can autonomously instantiate abstract societies by adopting roles in them. The creation, operation, and dissolution of societies are achieved by agents acting autonomously, but satisfying their commitments. A commitment can be canceled, provided the agent then satisfies the metacommitments applying to its cancelation.

The representations for IOP must support several functionalities, which typically exist informally, and are either effected by humans in some unprincipled way, are hard-coded in applications, or are buried in operating procedures and manuals. Information typically exists in data stores, or in the environment, or with interacting entities. Existing approaches do not model the interactive aspects of the above. The IOP contribution is that it

- enhances and formalizes ideas from different disciplines
- separates them out in an explicit conceptual metamodel to use as a basis for programming and for programming methodologies
- makes them programmable.

The notion of commitments may be familiar from databases. However, in databases, commitments correspond to a value being declared and are identified with the successful termination of a transaction. When a transaction terminates successfully, it commits, but it is not around any longer to modify its commitments. Thus, the commitments are rigid and irrevocable. If the data value committed by one transaction must be modified, a separate, logically independent transaction must be executed to commit the modified value. Traditional commitments presuppose that different computations are fully isolated and that locks can be held long enough that the atomicity of distributed computations can be assured.

Although suitable for traditional data processing, the above reasons cause traditional commitments to be highly undesirable for modern applications, such as electronic commerce and virtual enterprises, where autonomous entities must carry out prolonged interactions with one another [12].

Commitments reflect an inherent tension between predictability and flexibility. By having commitments, agents become easier to deal with. Also, the desired commitments serve as a sort of requirement on the construction of the agents who meet those commitments. However, commitments reduce the options available to an agent.

3.1 Commitments

We propose an alternative characterization of commitments that is better suited to agents and multiagent systems. In our formulation, the commitments are directed to specific parties in a specific context. Thus, an agent might not offer the same commitments to every other agent. The context is the multiagent system within which the given agents interact. An agent or multiagent system with jurisdiction over some resources and agents is called a *sphere of commitment (SoCom)*.

The *debtor* refers to the agent who makes a commitment, and the *creditor* to the agent who receives the commitment. Commitments are formed in a *context*, which is given by the enclosing SoCom (or, ultimately, by society at large). Based on the above intuitions, we motivate the following logical form for commitments.

Definition 1 A commitment $\mathsf{C}(x, y, p, G)$ relates a debtor x, a creditor y, a context G, and a discharge condition p.

3.2 Operations on Commitments

We define the following operations on commitments.

O1. *Create* instantiates a commitment. It is typically performed as a consequence of an agent adopting a role or by exercising a social policy (explained below).

O2. *Discharge* satisfies the commitment. It is performed by the debtor concurrently with the actions that lead to the given condition being satisfied.

O3. *Cancel* revokes the commitment. It can be performed by the debtor.

O4. *Release* essentially eliminates the commitment. This is distinguished from both *discharge* and *cancel*, because *release* does not mean success or failure of the given commitment, although it lets the debtor off the hook. The *release* action may be performed by the context or the creditor of the given commitment.

O5. *Delegate* shifts the role of debtor to another agent within the same context, and can be performed by the new debtor or the context.

O6. *Assign* transfers a commitment to another creditor within the same context, and can be performed by the present creditor or the context.

Through an abuse of notation, we write the above operations also as propositions, indicating their successful execution. We define some additional operations and propositions corresponding to important speech acts. These include *notify* and *authorize*. $\mathit{notify}(x, y, q)$ mean that x notifies y of q, and $\mathit{authorize}(x, y, p)$ means that x authorizes y to allow condition p.

3.3 Policies

Social policies are conditional expressions involving commitments and operations on commitments. Policies have a computational significance, which is that they can help control the execution of operations on commitments, even without explicit reference to the context. It is their locality that makes policies useful in practice. Agents can commit to social policies just as to other expressions; in this case, the agents' commitments are higher order, and are termed *metacommitments*. An example metacommitment is $\mathit{cancel}(x, \mathsf{C}(x, y, p, G)) \Rightarrow \mathit{create}(x, \mathsf{C}(x, y, q, G))$, which means that x can cancel his commitment for p if instead he adopts a commitment for q (for suitable p and q).

3.4 Social Learning

Agents that function as part of a community can learn not only from their own actions, but also from other members of the community. For example, imagine a logistical deployment. When the deployment is managed in a conventional manner, there is a central authority that generates a plan for how assets are deployed. However, such a centralized plan is usually unable to deal with exceptions or provide contingencies for all unanticipated situations.

When the management of a deployment is decentralized by using multiple agents, a "commuter model" arises. In the commuter model, an agent is associated with each item that is being deployed, and the agent is responsible for reaching its assigned destination without centralized control, much like how people commute to work each day. Each commuter operates within a defined, but dynamic infrastructure (roads, rails, traffic lights, etc.) designed to a master plan (urban plan, rapid transit, highway plan). However, detailed commuting decisions (whether to go by bus, rail, or freeway; what time to leave for work; which car goes next at an intersection; etc.) are made locally by the commuter using simple rules. Contingencies or changes in operation are also local commuter decisions, but they are supported by an infrastructure of services (traffic lights, traffic police, traffic reports, tow trucks, etc.), and monitoring and planning facilities (traffic counters, urban planners, etc.).

The agents follow the commuter model by shifting the detailed decision process to the lowest practical level: individual items or vehicles. Each agent makes dynamic local decisions to:

- Decide the means of conveyance
- Contend for storage, transportation and special handling resources
- Cooperate with other agents to satisfy mutual goals
- Request services to handle special problems and contingencies
- Respond to real-time queries or changes in objectives and plans
- Transfer acquired expertise to other agents.

Commuting becomes successful when people accumulate expertise through repetition. By traveling to and from the same location each day, they learn how to deal with the exceptions that occur and they discover opportunities to improve their efficiency. Logistics materiel, however, would not typically repeat a deployment path. How then could a logistics system benefit from the experiences of the materiel? The agents could learn in a different way from each other. The expertise gained by one agent as it traversed a deployment path could be shared with other agents that later have to traverse the same path. For example, a box that is mishandled at an intermediate depot, causing it to be late to its destination, might advise other items to avoid that depot or to budget additional travel time.

We envisage two kinds of learning to take place. One is class-level learning, e.g., the learning that adapts the general algorithms or strategies pursued by intelligent agents for different kinds of materiel. The other is instance-level learning that adapts the tactics used by a particular agent responsible for a particular physical item. The former could take place in a small number of sites and be used to update the programs with which new intelligent agents are instantiated. The latter would take place within each agent, but could be used to feed into the strategic learning by having the agents register when they arrive at various depots. The above approach would make the system adaptable both strategically and tactically. Successful learning requires the agents to adopt a social commitment for improving the performance of other agents.

4 Commitments for Interoperation

A number of architectures for cooperative information systems—multiagent systems in information-rich environments—have been proposed. These architectures not only allow a cooperative information system to be constructed and managed, but also can be seen as paradigms for achieving interoperation. Despite small variations, these architectures are converging to a more or less "standard" pattern. This pattern is based on the idea of mediators, as proposed by Wiederhold [16].

A mediator is a simplified agent that acts on behalf of a set of information resources or applications. The basic idea is that the mediator is responsible for mapping the resources or applications to the rest of the world. Mediators thus shield the different components of the system from each other. To construct mediators effectively requires some common representation of the meanings of the resources and applications they connect. Such a representation is called an ontology [10], and it is often managed by its own specialized agent.

It is a sign of their maturity that cooperative information systems are beginning to evolve a standard set of agent types. These are

- User agents, which contain mechanisms to select an ontology; support a variety of interchangeable user interfaces, such as query forms, graphical query tools, menu-driven query builders, and query languages; support a variety of interchangeable result browsers and visualization tools; maintain models of other agents; and, provide access to other information resources, such as data analysis tools, workflows, and concept learning tools.

- Broker agents implement a "yellow pages" and "white pages" directory service for locating appropriate agents with appropriate capabilities. Brokers manage a namespace service, and may have the ability to store and forward messages, and locate message recipients. Broker agents also function as communication aides, by managing communications among the various agents, databases, and application programs in the environment.

- Resource agents come in a variety of common types, depending on which resource they are representing, and provide a variety of capabilities. Wrappers implement common communication protocols and translate into and from local access languages. For example, a local data-manipulation language might be SQL for relational databases or OSQL for object-oriented databases. Database agents manage specific information resources. Data analysis agents apply machine learning techniques to form logical concepts from data or use statistical techniques to perform data mining. Resource agents apply the mappings that relate each information resource to a common context to perform a translation of message semantics. At most n sets of mappings and n resource agents are needed for interoperation among n resources and applications, as opposed to $n(n-1)$ mappings that would be needed for direct pairwise interactions among n resources without agents.

- Execution agents, which might be implemented as rule-based knowledge systems, e.g., in CLIPS, are employed to supervise query execution; operate as script-based agents to support scenario-based analyses; execute workflows, which might extend over the web and might be expressed in a format such as the one specified by the Workflow Management Coalition [15].

- Mediators are specialized execution agents. They determine which resources might have relevant information using help from brokers; decompose queries to be handled by multiple agents; combine the partial responses obtained from multiple resources; and translate between ontologies.

- Ontology agents are essential for interoperation. They provide a common context as a semantic grounding, which agents can then use to relate their individual terminologies; provide (remote) access to multiple ontologies; and manage the distributed evolution and growth of ontologies. A common context in the form of an ontology or model of the domain can provide such semantic grounding.

The resultant architecture of standard agent types renders development and deployment of CISs much easier, and essentially raises the abstraction level at which CISs can be described. Most agent-based information systems incorporate one or more agents of the above types.

4.1 Interoperation

This standard architecture described above and any of its variants help resolve some important issues in the design of multiagent systems. These issues are

- Deciding the main functionalities that could be provided as part of the routine infrastructure on which domain-specific and application-specific solutions may be readily built; the functionalities would ideally be available in reusable toolkits, but do not necessarily have to be.

- Determining how each agent may interface with other agents, such as how an agent may register with or unregister from a broker.

- Determining the normal control flow in an application.

By itself, the standard architecture and the agent types that constitute it are quite nice for the issues they are designed to address. Mediators generalize naturally from the traditional client-server architecture used in database-centric information systems. Brokers and directories can provide useful services in helping find agents with specified names or supported capabilities.

There is also a semantic aspect to effective interoperation. The above architecture addresses that aspect through the notion of ontologies and ontology agents. An ontology can provide a shared basis for the exchange of information among agents, and can thus help reduce or eliminate the effects of terminological discrepancies.

However, there is yet another aspect of interoperation, which may be termed *pragmatic*. This aspect involves how the agents interact with each other, and goes beyond the exchange and comprehension of the terms used in the individual messages. This aspect deals with the high-level protocols governing how the agents should interact. It has a normative force to it, and requires that the agents behave felicitously. In other words, this is what deals with the coherence requirements of the cooperative information system being designed.

Having a separate specification of the coherence requirements can be easily accomplished by specifying an abstract multiagent system in terms of the roles required for it. For each role, one can specify the coherent behaviors in terms of the commitments entered into and satisfied by any agent playing the role under different circumstances. This serves two important purposes.

- The clear specification of coherence requirements is an important step in the design of a system, and corresponds roughly to the capturing of the important use-cases [**?**] in traditional software engineering [**?**].

- The coherence requirements can be used as a touchstone for evaluating agents contributed by different vendors. Any agent that is touted as being able to play a role in a certain abstract system must be able to satisfy the coherence requirements for the system as they reflect on the given role. This is the key to true interoperation in open settings, where we would expect the agents to be contributed by independent vendors and to belong to different people or commercial interests. A multiagent system can thus be incrementally modified while executing as long as the agents being added met the coherence requirements.

4.2 Commitments in Execution

We envisage the following way to apply commitments. Initially, abstract SoComs are defined in terms of their *roles*. Each role is associated with the capabilities it requires, the commitments it engenders, and the authorities it creates. The capabilities are the tasks the agent can do, the commitments are what the agent must do, and the authorities are what the agent may do. The commitments, in particular, may be metacommitments. Indeed, they usually are metacommitments, e.g., that the agent will adopt a base commitment upon receiving a request.

At some point, possibly during execution, an agent may decide to enter into a SoCom as a particular role or roles. To do so, the agent would have to cause the SoCom to be instantiated from the abstract specification. To adopt a role, the agent must have the necessary capabilities, and accept the associated commitments. In doing so, he also obtains the authorities to properly play the role. The agent must then behave according to the commitments. Agents can join a SoCom when configured by humans or during execution: this requires publishing the definition of the abstract SoCom.

5 Applications

We begin with a reconstruction of the MINDS system as a referral network with social abstractions. Then, we consider an example in two parts. The first deals with electronic commerce; the second adds aspects of virtual enterprises [9]. The commitments are designed based on their corresponding roles in human society.

5.1 Referral Networks

The ready online availability of information often leads to the problem of judging its quality and relevance. For this reason, it is often preferable to ask someone who knows than to attempt to collect and assimilate the information all by oneself. Informally, people try to remember which of their acquaintances is familiar with and can give reliable information about a certain class of topics. There is naturally enough interest in building agent-based referral networks as well.

One of the earliest projects that constructed referral networks of agents was the MINDS project [4]. We briefly review this project next, and then reconstruct its referral components from the perspective of the social abstractions introduced here.

MINDS (Multiple Intelligent Node Documents Servers) is a distributed collection of agents for efficiently managing and retrieving documents in a networked environment (see Figure 2). Each agent can be considered a personalized expert in the domain of "documents." The agents share both knowledge and tasks to cooperate in retrieving documents for users. The agents are able to customize the document retrievals for each user by learning document distribtion patterns, as well as user interests and preferences, during operation. The knowledge is learned with the help of heuristics for assigning credit and recommending adjustments. Because of this ability to learn, they are also self-initializing.

Document management and retrieval is a social problem, because it requires the interaction and cooperation of a community of users and agents. However, in order to behave robustly in the presence of a variety of agents, MINDS maintains heuristics for mitigating the effects of uncooperative agents. Two examples of the heuristics are the following:
Heuristic 1: IF a document is created by user1, THEN metaknowledge of user1 about user1 regarding each keyword of the document is increased to 1.0 (maximum relevance) i.e., users are presumed to know a lot about the documents they create.
Heuristic 2: IF a retrieve predicated on keyword1 is issued by user1,
AND at least one user2 surrogate contains keyword1,
THEN (a) user1 metaknowledge about user2 regarding keyword1 is increased (weight 0.1),
(b) user2 metaknowledge about user1 regarding keyword1 is increased (weight 0.1) i.e., if a query is answered successfully, then both the questioner and the responder learn about their mutual interest in the subject area of the query.

Each agent in MINDS manages a set of documents and a document directory. The directory contains (1) document surrogates (attributes of a document, such as its name, authors, version, location, creation date, type, access privileges, and keywords), (2) a domain ontology that organizes the document keywords and is shared by all agents and users in the community, and (3) models of both the current system state (from each agents's local perspective) and the local user's document preferences. The directory is used to guide each agent's search for requested documents, which proceeds in parallel according to a best-first strategy, and improves in efficiency as the system is used. The efficiency improvement can be modeled as a form of feedback learning, as shown in Figure 3.

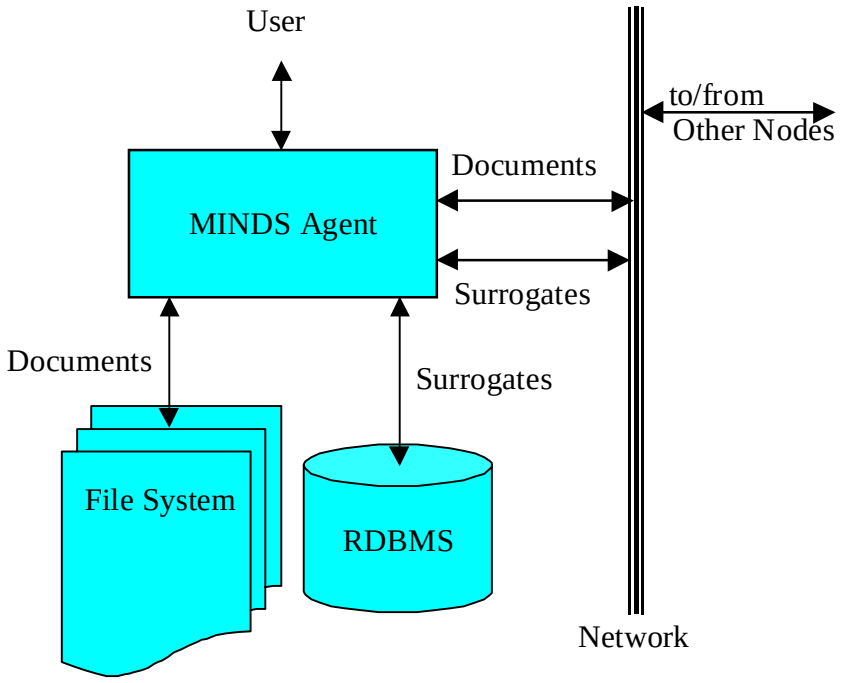
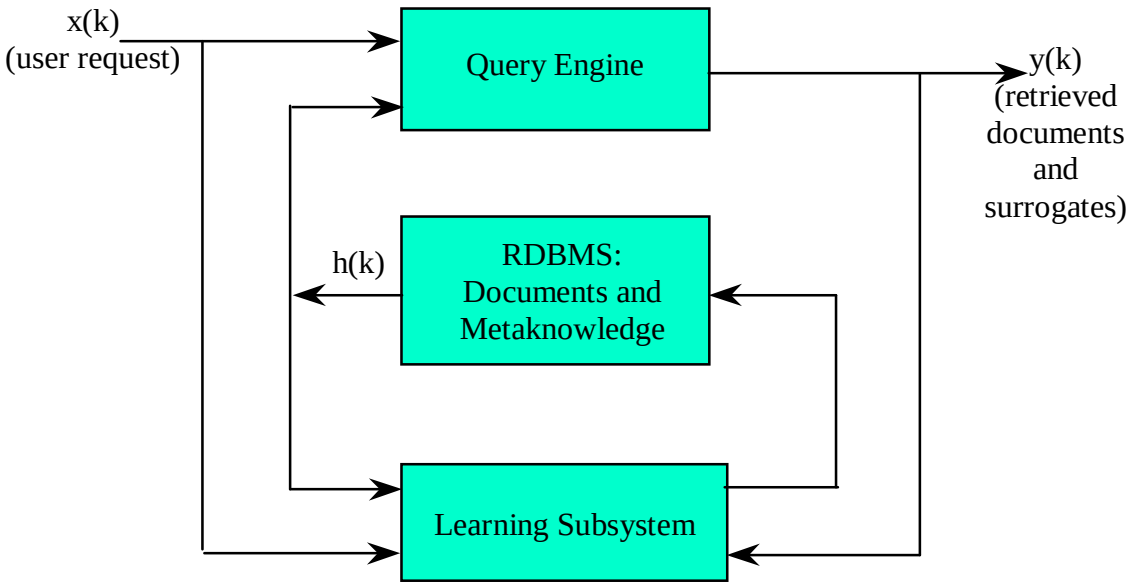


Figure 2: The architecture of MINDS



Output function: $y(k) = q(x(k), h(k))$
State function: $h(k+1) = a(x(k), h(k))$

Figure 3: The learning dynamics within each MINDS agent

The agents make a social commitment to be cooperative, in that they forward queries that they cannot themselves answer. Because questioners and responders each learn from both positive and negative examples, i.e., successful and unsuccessful queries, a newly introduced agent can soon become an active participant in locating and retrieving documents. In this way, MINDS is self-initializing.

5.2 Electronic Commerce

We first define an abstract SoCom consisting of two roles: *buyer* and *seller*, which require capabilities and commitments about, e.g., the requests they will honor and the validity of price quotes. To adopt these roles, agents must have the capabilities and acquire the commitments. Example 1 involves two individual agents who adopt the roles of *Buyer* and *Seller* to carry out a simple deal.

Example 1 Consider a situation involving two agents, *Customer* and *Vendor*, with authority over their respective databases. The SoCom manager has an abstract SoCom for buy-sell deals with the roles of *Buyer* and *Seller*. *Buyer*'s capabilities include asking for a price quote and placing an order. *Seller*'s capabilities include responding to price quotes and accepting orders based on checking the inventory locally. *Buyer*'s commitments include paying the quoted price for anything *Buyer* orders. *Seller*'s commitments include (a) giving price quotes in response to requests and (b) fulfilling orders that *Seller* has accepted.

Customer asks the manager to instantiate a deal between (*Customer*) as *Buyer* and *Vendor* as *Seller*. The manager asks *Vendor* if it would like to join as *Seller*. When *Vendor* agrees, and since both agents have the requisite capabilities, capacities, and resources, the deal is set up.

Customer now wishes to check the price of a valve with a diameter of 21mm. Upon receipt of the query from *Customer*, *Vendor*—based on its role as *Seller*—offers an appropriate answer. ∎

5.3 Virtual Enterprises

Now we consider more general situations where one or more agents may form a cooperative SoCom or team. For simplicity, we assume that teams have a distinguished agent who handles their external interactions. We refer to this agent as the VE.

Example 2 In the first situation, there are two agents with authority over the Valvano and Hoosier databases. These agents have similar capabilities to the *Seller* of Example 1. They form a VE, called Valvano-cum-Hoosier VE, which can adopt the role of *Seller*. *Buyer* behaves as before and expects *Seller* to behave according to the buy-sell deal. However, *Seller* is implemented differently, with commitments among its members, which we do not elaborate here. The possible commitments of the Valvano-cum-Hoosier VE include the following.

- The VE will give price quotes to anyone who requests them.
- The VE will refund the purchase price if an order with matching valves and hoses cannot be fulfilled. There are still no refunds if an order for matching valves and hoses can be fulfilled.
- If the VE cannot fulfill an order, it will try to find an alternative order that will satisfy *Customer*'s requirements.

The original vendors, *val* or *hos*, would not take refunds individually. Thus, a customer might be saddled with valves for which matching hoses could not be found. However, when dealing with the VE, a customer can get a refund in those situations. ∎

In the above example, the actions are performed by the constituents of the SoCom. Sometimes, however, it is useful to perform actions at a higher level SoCom. Such actions might be necessary when the actions of the member agents need to be atomically performed or undone.

Example 3 Continuing with Example 2, suppose an order for matching valves and hoses is successfully placed. It turns out later that the valve manufacturer discontinued the model that was ordered, but recommends a substitute. The substitute valve fits different diameter hoses than the original choice. The VE knows that the original order could be satisfied using the new valve and a different set of hoses. The VE can handle this replacement itself and, based on its prior commitment, not charge the customer any extra. The customer does not need to know of the internal exchanges among the members of the VE SoCom. ∎

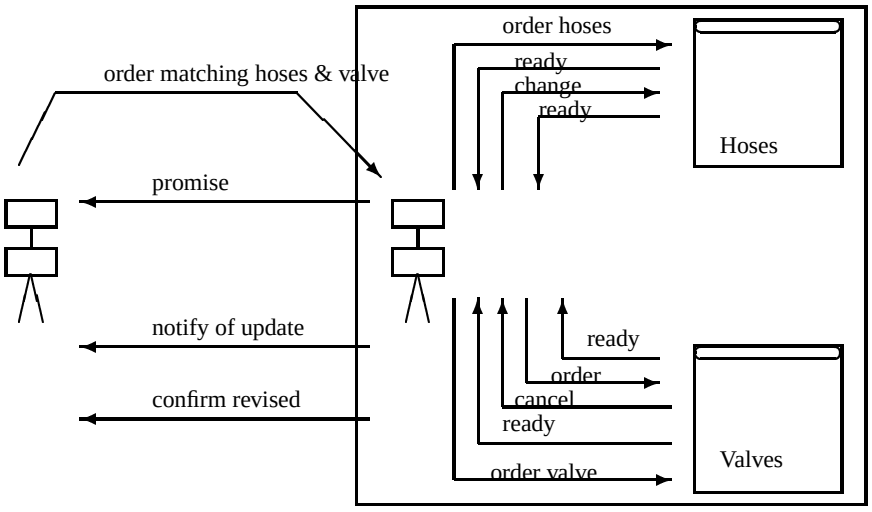


Figure 4: Commitment-Based Recovery

In this situation, where the Valvano-cum-Hoosier VE detects a problem in the supply of valves for which an order has been placed, the VE automatically meets its commitments by revising the order and notifying the customer. The discontinuation of a valve after an order for it was accepted exemplifies a kind of failure that arises after an original interaction had ended. Traditional approaches would be inapplicable in such a situation.

6 Conclusions and Future Work

Although multiagent systems have been known for a number of years and practical applications of them are spreading, building good multiagent systems remains somewhat of a black art. These problems also hold in information-rich environments. Prevailing techniques do not support the fundamental properties that make MAS attractive, either because they are suited to conventional, consistency-based settings or because they ignore special properties of the underlying systems, resulting in completely *ad hoc* modes of constructions.

Engineering in general must be based on good science, albeit with the availability of tools and methodologies that facilitate the application of the scientific ideas. The engineering of cooperative information systems is no different. Work on social abstractions has been proceeding for a while, although it was mostly centered around a rather small community of researchers. With the expansion of information-rich environments, a larger body of researchers has accreted to the area. For this reason, progress has accelerated. Although we do not claim that the approach described here is ready for commercial use, its uses for research prototypes have been promising.

We described interaction-oriented programming, and outlined some conceptual modeling issues in it. IOP offers some benefits over previous approaches for building multiagent systems. In the spirit of conceptual modeling, IOP focuses on higher-level concepts than the underlying implementations. These concepts provide a superior starting point to the traditional approaches. Fundamentally, conceptual modeling is as good as the methodologies that one may use to build conceptual models. Accordingly, we have been considering methodologies that may be applicable to IOP. In the above, we gave a sampler of some of our preliminary results. These methodologies are presently being applied by hand, although there is some work afoot to build tools to assist in their application.

The idea of using the social abstractions to specify the coherence requirements in a system is a valuable one. On the one hand, it can be used to ensure that the agents can be constructed more easily to yield the complex varieties of behavior that are often desired. On the other hand, it can be used as a basis for specifying standard or quasi-standard cooperative information systems for which independent developers and vendors could supply the member agents.

There are some important directions for future research. Of special interest is the development of richer metamodels to capture the realistic ways in which a set of computations may be considered coherent. This requires empirically identifying the “best practices” in key application domains, encapsulating them as reusable patterns, and incorporating them in metamodels. Typical application areas would be contracting among autonomous entities, accounting, and other enterprise functions.

There is a strong need for an intuitive formal semantics. We have made some progress along this direction [13]. A related theme is compositionality, which we applied in combining individual models for electronic commerce and virtual enterprises. However, a full formal treatment of specification and verification that would allow dynamic composition of models remains to be made; for more traditional temporal logics approaches, such results are now becoming available [**?**]. In related research, we have begun developing algorithms for temporal, causal reasoning, which can be used to analyze and design commitment protocols for high-level interaction among autonomous agents. We hava also begun to develop techniques for social learning among agent communities.

Acknowledgments

A previous version of this paper was presented at the International Workshop on Cooperative Information Agents (CIA), held in 1998. Michael Huhns was supported by the Defense Advanced Research Projects Agency. Munindar Singh was supported by the NCSU College of Engineering, the National Science Foundation under grants IIS-9529179 and IIS-9624425 (Career Award), and IBM corporation.

References

[1] Ramez Elmasri and Shamkant Navathe. *Fundamental of Database Systems*. Benjamin Cummings, Redwood City, CA, second edition, 1994.

[2] Stan Franklin and Art Graesser. Is it an agent or just a program?: A taxonomy for autonomous agents. In *Intelligent Agents III: Agent Theories, Architectures, and Languages*, pages 21–35. Springer-Verlag, 1997.

[3] Jim Gray and Andreas Reuter. *Transaction Processing: Concepts and Techniques*. Morgan Kaufmann, San Mateo, 1993.

[4] Michael N. Huhns, Uttam Mukhopadhyay, Larry M. Stephens, and Ronald D. Bonnell. DAI for document retrieval: The MINDS project. In Michael N. Huhns, editor, *Distributed Artificial Intelligence*, pages 249–283. Pitman/Morgan Kaufmann, London, 1987.

[5] Michael N. Huhns and Munindar P. Singh. The agent test. *IEEE Internet Computing*, 1(5):78–79, October 1997. Instance of the column *Agents on the Web*.

[6] Michael N. Huhns and Munindar P. Singh. Agents and multiagent systems: Themes, approaches, and challenges. In *[8]*, chapter 1, pages 1–23. 1998.

[7] Michael N. Huhns and Munindar P. Singh. A multiagent treatment of agenthood. *Applied Artificial Intelligence*, 1998. To appear.

[8] Michael N. Huhns and Munindar P. Singh, editors. *Readings in Agents*. Morgan Kaufmann, San Francisco, 1998.

[9] Anuj K. Jain and Munindar P. Singh. Using spheres of commitment to support virtual enterprises. In *Proceedings of the 4th ISPE International Conference on Concurrent Engineering: Research and Applications (CE)*, pages 469–476. International Society for Productivity Enhancements (ISPE), August 1997.

[10] Ramesh S. Patil, Richard E. Fikes, Peter F. Patel-Schneider, Don McKay, Tim Finin, Thomas Gruber, and Robert Neches. The DARPA knowledge sharing effort: Progress report. In *[8]*, pages 243–254. 1998. (Reprinted from *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning, 1992*).

[11] Charles J. Petrie, Jr. Agent-based engineering, the web, and intelligence. *IEEE Expert*, 11(6):24–29, December 1996.

[12] Munindar P. Singh. Commitments among autonomous agents in information-rich environments. In *Proceedings of the 8th European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW)*, pages 141–155. Springer-Verlag, May 1997.

[13] Munindar P. Singh. An ontology for commitments in multiagent systems: Toward a unification of normative concepts. *Artificial Intelligence and Law*, 1998. In press.

[14] Peter Wegner. Why interaction is more powerful than algorithms. *Communications of the ACM*, 40(5):80–91, May 1997.

[15] The WfMC. Workflow management coalition (WfMC) reference model. http://www.aiai.ed.ac.uk/WfMC/, 1995.

[16] Gio Wiederhold. Mediators in the architecture of future information systems. In *[8]*, pages 185–196. 1998. (Reprinted from *IEEE Computer, 1992*).