Agent Foundations for Cooperative Information Systems

Michael N. Huhns
Center for Information Technology
University of South Carolina
Columbia, SC 29208 USA
huhns@sc.edu

Abstract

Cooperative Information Systems (CIS) is a research area that has emerged from the synthesis of databases and artificial intelligence. It can succinctly be defined as the study of multiagent systems and organizational and database abstractions geared toward the large, open, heterogeneous information environments of today. This short paper describes how CIS architectures have evolved a set of common types of computational agents.

1. Introduction

Due to the proliferation of networking, the desires of most everyone to be interconnected, and the needs to make data accessible any time and any place, modern information environments have become large, open, and heterogeneous. They are composed of distributed, largely autonomous, often legacy-based components. In cooperative information systems, software agents are introduced into such environments to deal with these characteristics. The agents represent the components in interactions, where they mediate differences and provide a syntactically uniform and semantically consistent middleware. Software agents mitigate the complexity in two important ways—one technical and the other psychological.

Technically, each agent provides a locus of intelligence for managing a subset of the information in the system, either on its own initiative or under the direction of a user. Each intelligent agent can be readily replicated and then distributed as needed. This agent-based approach to information management is both scalable and cost-effective.

Psychologically, people need abstractions by which they can understand, manage, and use complex systems effectively. A natural and convenient abstraction appears to be one based on anthropomorphizing the information system components, that is, treating the components as animate. In this abstraction, software components are like human agents. The abstraction is effective, because people have a lot of experience in dealing with other people, and they can apply their experience to understanding and dealing with complex software.

Animate components are an example of a more general trend in software engineering to construct software that mimics real-world objects. If the mimicry is done well, the software will appear familiar and thus easy to use. The net result is a great interest in software agents.

Their greatest difficulty in achieving uniformity and consistency is the dynamism that open environments introduce. Open environments are becoming an increasing part of the modern milieu through applications such as information search, electronic commerce, and virtual enterprises. They typically have the following key distinguishing characteristics:

- span enterprise boundaries;
- have components that are heterogeneous in a number of ways, such as the underlying database management systems used, and the semantics associated with the information stored or manipulated;
- comprise information resources that can be added or removed in a loosely structured manner;
- lack global control of the content of those resources, or how that content may be updated; and
- incorporate intricate interdependencies among their components.

To build applications that work effectively within open environments requires balancing their ease of construction and robustness with their flexibility. An agent-based cooperative information system approach appears to meet this requirement. Section 2 overviews cooperative information systems, and their quintessential applications and problems. Section 3 describes a variety of agent types that have emerged for cooperative information systems. Section 4 concludes with a discussion of the main themes of CISs, and how they relate to multiagent systems.

2. An Overview of CIS

A cooperative information system represents an increasingly popular approach that seeks to maximize ease of construction, flexibility, and robustness through combinations of techniques from distributed artificial intelligence, databases, and distributed computing. In CIS, software agents mitigate an information environment's heterogeneity by interacting through common protocols, and manage its large size by making intelligent local decisions without centralized control.

We define an *agent* as an active, persistent computational entity that can perceive, reason about, and act in its environment, and can communicate with other agents. Agents are autonomous to varying degrees to reflect the autonomy of the information resources or humans whom they represent. Figure 1 shows a CIS schematically. In this figure, we consider 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 information about the resources, including any constraints that apply to their joint behavior.

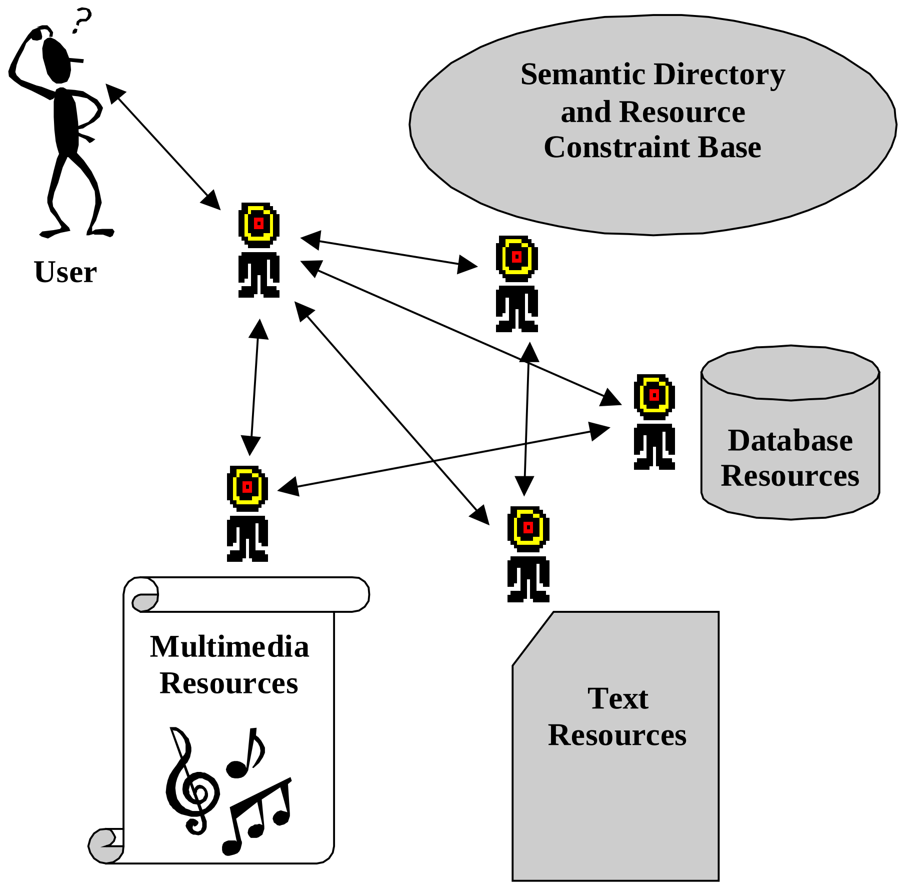

Figure 1. A schematic view of a cooperative information system, with agents representing the components

Each component of the environment, as well as the human user, is associated with an agent, who captures and enforces the requirements of its associated party. Agents interact with one another appropriately, and help achieve the robustness and flexibility in behavior that is required. The appeal of agents is that they provide a natural means for acquiring, managing, advertising, finding, fusing, and using information over uncontrollable environments. Further, agents are inherently modular, and can be constructed locally for each resource, provided they satisfy some high-level protocol of interaction.

The applications of CISs are varied. They involve the purely informational ones, such as database access, information malls, workflow management, electronic commerce, and virtual enterprises. They also include physical ones, such as sensor arrays, manufacturing, transportation, energy distribution, and telecommunications.

The above motivates the interest in cooperative information systems. Before we discuss the commonalities in their architectures, it is instructive to review two of the quintessential applications of CIS: information access and workflow management.

2.1 Information Access

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 communications infrastructure and 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! This would be a fundamental step backward from the very idea of a database architecture [Elmasri & Navathe 1994].

A promising approach is to use *mediators* [Wiederhold 1992]. A mediator is a simplified agent that acts on behalf of a set of information resources or applications. Figure 2 shows a mediator architecture. 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* [Neches et al. 1991], and it is often managed by its own specialized agent.

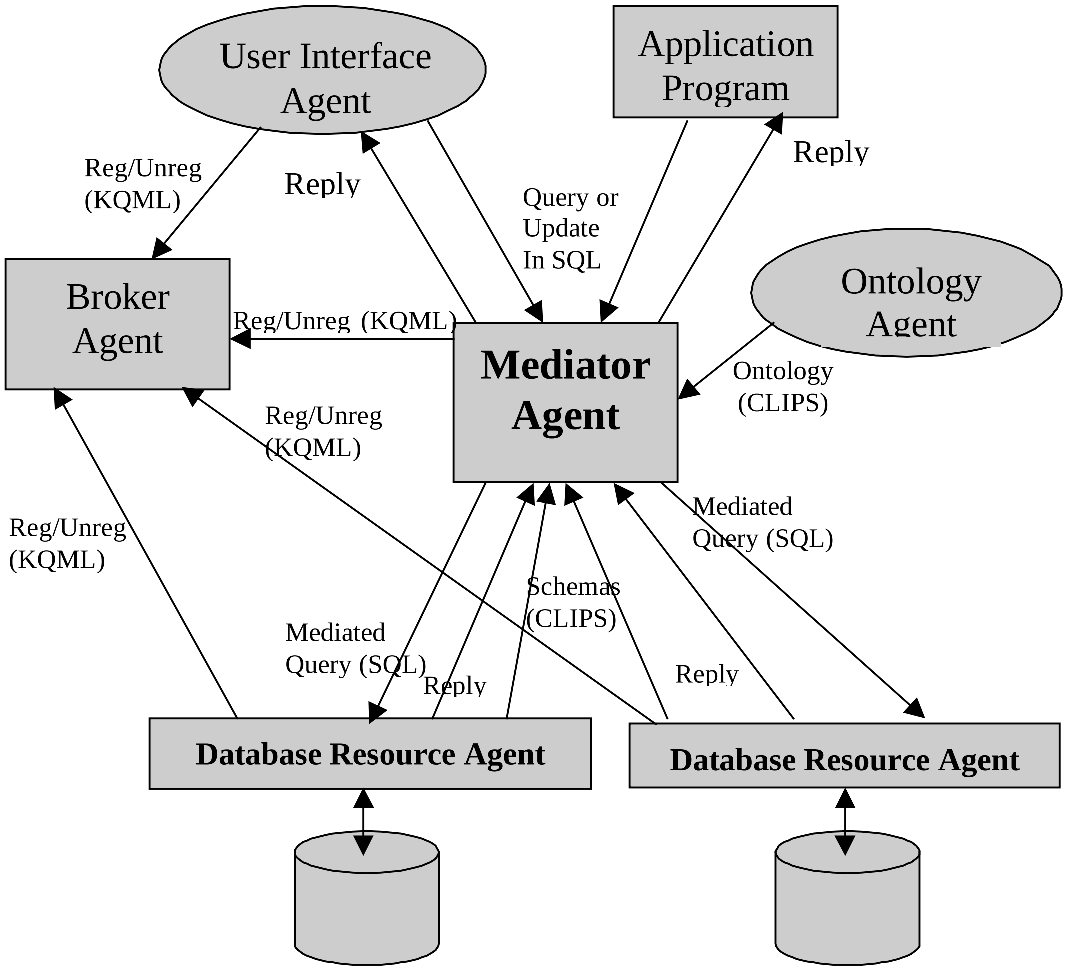

Figure 2. Agent architecture for a cooperative information system based on a mediator agent

2.2 Workflow Automation

CISs not only involve retrieving information, but also updating it. Updates are qualitatively more complex than retrievals, because they can potentially introduce inconsistencies. This is especially the case when several databases are involved, and there are subtle interdependencies among them. A *workflow* is a composite activity that accesses different resources and has human interaction to solve some business need.

Traditional databases support so-called ACID transactions, which are computations that are atomic, consistency-preserving, isolated, and durable [Gray & Reuter 1993]. In other words, a transaction happens entirely or not at all, does not violate consistency, does not expose any partial results, and if successful has permanent results. Transactions are effective in homogeneous and centralized databases, but do not apply in distributed and heterogeneous settings. This is because to ensure the ACID properties requires the component databases to expose their internal control states, and requires locking data items on a database even when they are no longer in use.

This has led to a number of extended transaction models [Bukhres & Elmagarmid 1996, Elmagarmid 1992], which capture some of the aspects of workflows. Consider a trip-planning workflow that has a number of separate activities, such as opening an account, reserving a flight, booking a hotel, renting a car, and generating a bill. These execute on databases belonging to autonomous organizations, such as different airlines or hotels. Since the airlines make reservations independently of each other and of hotel bookings, the travel agency has to provide the control to make sure air tickets are not bought unnecessarily. Typically, a human would carry out the necessary steps. Approaches such as [Buchman et al. 1992] provide a way of representing the dependencies among the steps, and executing them appropriately. However, a major challenge is in automating workflows with agents, possibly by mimicking how humans carry them out.

3. CIS Agent Types

Cooperative information systems are beginning to evolve a standard set of agent types, which renders development and deployment of CISs much earier. Some of these are
User agents, which have the following characteristics:

- 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
- 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 the following 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.
- SQL database agents manage specific information resources
- Data analysis agents apply machine learning techniques to form logical concepts from data or use statistical techniques to peform 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 (see Figure 3).

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, e.g., based on specifications from the Workflow Management Coalition, where the workflows might extend over the web.

Mediators are specialized execution agents, which

- Determine which resources might have relevant information with help from brokers
- Decompose queries to be handled by multiple agents
- Combine the partial responses
- 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
- 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.

Most agent-based information systems incorporate one or more agents of the above types.

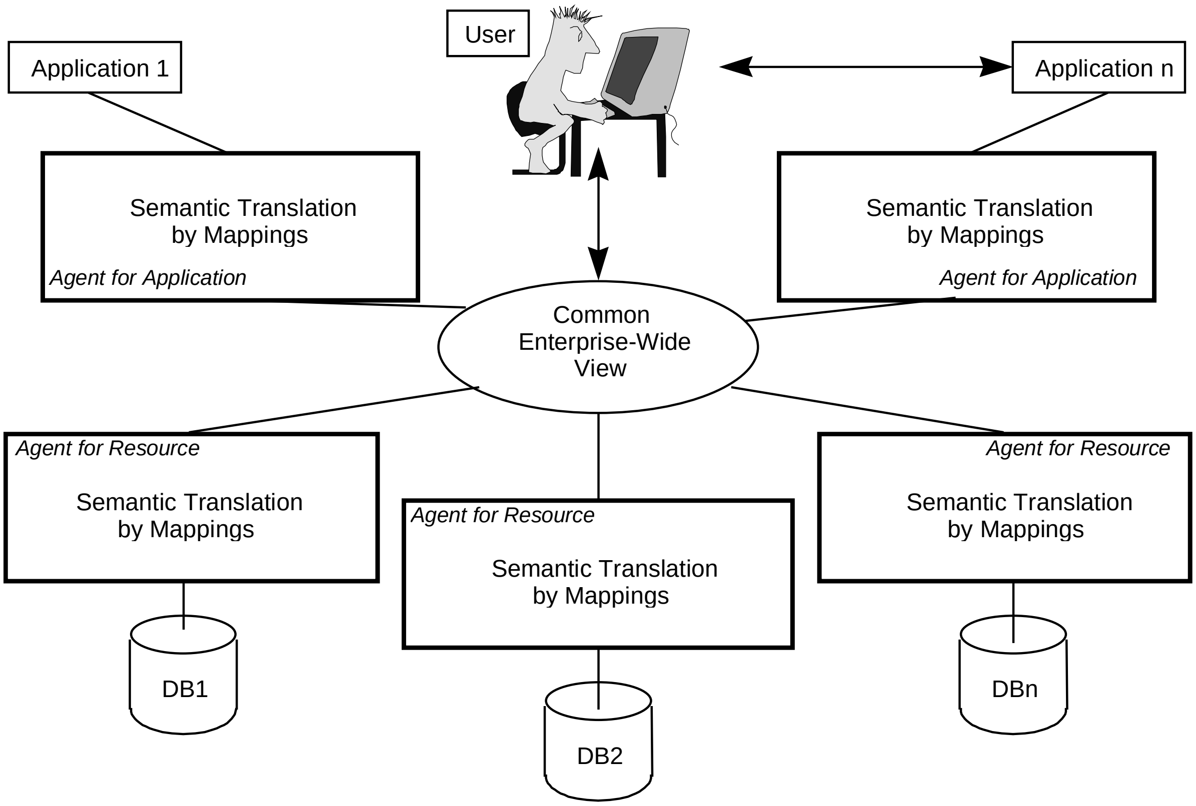

Figure 3: Logical view of a CIS execution environment, showing how agents apply mappings to achieve semantic interoperation

4. Conclusion

For years, information-system personnel managed corporate data that was centralized on mainframes. The data was kept consistent, but eventually the amount of data increased to the point that centralized storage was no longer viable. Also, users wanted a way to share data across applications and wanted more direct involvement in the management of the data. So, data began proliferating onto workstations and personal computers, where users could manage it themselves. But this resulted in redundancy, inconsistency, and no coherent global view. Hence, there are now attempts to reintegrate data. Users still need to manage their own data, which remains distributed, but they and their applications need coherent global access, and consistency must be restored.

This paper describes a cooperative information system approach to enabling interoperation among enterprise information objects, i.e., among suppliers and consumers of information. In this approach, an enterprise information object is integrated based on semantic mappings defined between two contexts: the context of a model of the object and a common enterprise-wide context. Locally sited agents implement the mappings.

In looking at the examples of CISs, we find that certain problems show up in different guises. The unifying themes of CIS are the following. One, there is a desire for the effect of logical homogeneity and centralization despite physical distribution and heterogeneity. Two, there is a need to support the logical openness of CISs. Openness translates into a number of interesting systemic challenges, relating to how a CIS may initialize and stabilize itself when some agents come together, are added, or leave.

Extensions of our work are focused on developing additional information-system applications for agents, including intelligent directory service agents, negotiating electronic data interchange (EDI) agents, database administration agents, and intelligent information retrieval agents. Our most important future work is centered on ways in which agents can acquire and maintain models of each other in order to improve their interactions.

Bibliography

[Buchman et al. 1992] Alejandro Buchman, M. Tamer Ozsu, M. Hornik, D. Georgakopoulos, and Frank A. Manola, “A Transaction Model for Active Distributed Object Systems,” in [Elmagarmid 1992], ch. 5, pp. 123-158.

[Bukhres & Elmagarmid 1996] Omran A. Bukhres and Ahmed K. Elmagarmid, editors, Objecty-Oriented Multidatabase Systems: A solution for Advanced Applications, Prentice Hall, 1996.

[Elmagarmid 1992] Ahmed Elmagarmid, ed., *Database Transaction Models for Advanced Applications*, Morgan Kaufmann Publishers Inc., San Mateo, CA, 1992.

[Elmasri & Nevathe 1994] Ramez Elmasri and Shamkant Nevathe, *Fundamentals of Database Systems*, 2nd edition, Benjamin Cummings, Redwood City, CA, 1994.

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

[Guha 1990] R. V. Guha, "Micro-theories and Contexts in Cyc Part I: Basic Issues," MCC Technical Report Number ACT-CYC-129-90, Microelectronics and Computer Technology Corporation, Austin, TX, June 1990.

[Neches et al. 1991] Robert Neches, Richard Fikes, Tim Finin, Tom Gruber, Ramesh Patil, Ted Senator, and William R. Swartout, “Enabling Technology for Knowledge Sharing,” *AI Magazine*, vol. 12, no. 3, pp. 36-56, 1991.

[Wiederhold 1992] Gio Wiederhold, "Mediators in the Architecture of Future Information Systems," *IEEE Computer*, Vol. 25, No. 3, March 1992, pp. 38-49.