Cooperative Agents for Information Gathering

by

Elhadi M. Shakshuki

A thesis
presented to the University of Waterloo
in fulfilment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Systems Design Engineering

Waterloo, Ontario, Canada, 2000

The University of Waterloo requires the signatures of all persons using or photocopying this thesis. Please sign below, and give address and date.

Abstract

The rapid growth of the network-centered (Internet and Intranet) computing environments requires a new design paradigm for information gathering systems. Typically, in these environments, information resources are dynamic, heterogeneous and distributed. Moreover, these computing environments are open, in the sense that information resources may join or disjoin at anytime.

Agent-based technology provides a promising design paradigm and has a growing appeal for designing cooperative distributed systems. This thesis develops a multi-agent architecture for cooperative information gathering systems (CIGS). These autonomous, goal-driven agents cooperatively assist different users to locate and retrieve information from distributed resources in an open environment. The system architecture is comprised of three tiers: at the front end, the User Agents interact with the users to fulfill their interests and preferences; at the back end, the Resource Agents access and capture the content and changes of the information resources; and at the middle tier, the Broker Agents facilitate cooperation among the agents.

Cooperation is an essential concept for information gathering in a multi-agent setting. In this thesis cooperation has been analyzed and instrumented to govern the agents' interaction in an information gathering domain. Various interaction strategies have been proposed at different levels of the agent's model. Through these strategies the agent might to choose to exhibit selfish, benevolent, or cooperative behavior based on the interaction style. The feasibility of the proposed system has been demonstrated by implementing a prototype. This implementation demonstrates how CIGS could be used to transparently locate and retrieve information from dynamic resources that might be distributed and heterogeneous.


قُل لَّوْ كَانَ ٱلْبَحْرُ مِدَادًا لِّكَلِمَٰتِ رَبِّى
لَنَفِدَ ٱلْبَحْرُ قَبْلَ أَن تَنفَدَ كَلِمَٰتُ رَبِّى
وَلَوْ جِئْنَا بِمِثْلِهِۦ مَدَدًا ﴿١٠٩﴾ قُلْ إِنَّمَآ
أَنَا۠ بَشَرٌ مِّثْلُكُمْ يُوحَىٰٓ إِلَىَّ أَنَّمَآ
إِلَٰهُكُمْ إِلَٰهٌ وَٰحِدٌ ۖ فَمَن كَانَ يَرْجُوا۟
لِقَآءَ رَبِّهِۦ فَلْيَعْمَلْ عَمَلًا صَٰلِحًا وَلَا
يُشْرِكْ بِعِبَادَةِ رَبِّهِۦٓ أَحَدًۢا ﴿١١٠﴾

[18:109] Say: "If the ocean were ink (wherewith to write out) the words of my Lord. Sooner would the ocean be exhausted than would the words of my Lord, even if we added another ocean like it, for its aid." [18:110] Say: "I am but a man like yourselves, (but) the inspiration has come to me, that your God is one God: whoever expects to meet his Lord, let him work righteousness, and, in the worship of his Lord, admit no one as partner."


قَالُوا۟ سُبْحَٰنَكَ لَا عِلْمَ لَنَآ إِلَّا مَا عَلَّمْتَنَآ ۖ
إِنَّكَ أَنتَ ٱلْعَلِيمُ ٱلْحَكِيمُ ﴿٣٢﴾

[2:32] They said: "Glory to Thee: of knowledge we have none, save what Thou hast taught us: in truth it is Thou Who art perfect in knowledge and wisdom."

Dedicated to the Memory of My Father

Mohamed E. Shakshuki

Acknowledgements

My all thanks and praise go to Allah for giving me the knowledge and the strength to accomplish this work.

I would like to thank my supervisor, Professor Mohamed Kamel, for his guidance, encouragement and support. I would also like to thank members of my doctoral committee, Professor Ahmed Karmouch, Department of Electrical and Computer Engineering, University of Ottawa, Professor Frank Safayeni, Department of Management Science, University of Waterloo, Professors Keith Hipel and Andrew Wong, Department of Systems Design Engineering, University of Waterloo, for their careful reading of the dissertation and valuable comments on my work.

Thanks to my best friend Hamada Ghenniwa. Apart of being a best friend, he was always there with his invaluable support and suggestions of this dissertation from the very first day that led to its final presentation.

I am indebted most of all to my family, especially my mother, brothers and sisters back home in Libya for all their kindness and love. My wife encouraged me, believed in me, and continuously gave me support. Most precious of all my children who always gave me something to smile about.

This research was supported by an international scholarship from the educational ministry of Libya. This support is gratefully acknowledged.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Globalization is the key to success in many organizations. Because every organization depends on its knowledge for effectively and efficiently executing its business mission, information is increasingly dispersed over many information resources. Although information that needs to be acquired is likely to be available in such an environment, parts of this information may also be difficult to locate. This location transparency is an important issue that traditional information retrieval systems are no longer able to handle. Traditionally, users supply the query and the location of the information resource; this approach may be applicable when the number of the information resources are few, but not feasible for open environments, such as Internet-based applications.

Although the Internet provides the infrastructure platform to access the information resources, several critical issues need to be addressed to provide value for it. Firstly, the explosion of the available information over the Internet, makes the user unable to locate the relevant information. Secondly, most of these information resources are dynamic. That is, their owners constantly update the resources in a

way that might alter their content. Thirdly, the information systems environment is open, permitting information resources to join or disjoin at any time.

With these issues in mind, the problem of how to gather information in an open environment becomes very challenging. Currently, Internet search engines are one of the commonly used tools for information gathering. Although these engines are valuable, they provide limited services and have several disadvantages. The search for information is limited and typically biased towards indexing more 'popular' information, and each search engine covers a small portion of information resources on the Internet [63]. In general, the task of the information gathering is the user's responsibility.

An information gathering system in an open environment is a system that is capable of locating, retrieving and integrating information. The main design principles for such a system should:

- provide the users with an integrated view of information.
- provide information relevant to the user's topic of interest and preferences,
- pro-actively search for information and avoid repetitive user's intervention,
- monitor any possible changes to the information resources and make the appropriate updates,
- provide information within a user-defined time limit,
- cope well with new information sources that may enter and old information sources that may exit the environment.

1.1 Cooperative Information Systems

A cooperative information system comprises of several entities that have limited knowledge, are able to perform some functions independently and exercise some degree of authority in sharing their capabilities. These entities, working together, may achieve individual and/or global goals in domains such as medicine and banking. Typically, cooperative information systems are either logically or physically distributed, their content is dynamic, and might be part of an open environment.

A cooperative information system can be modeled as a multi-agent system in which each agent is autonomous, intelligent, rational and able to coordinate and cooperate with other agents. In this view, the information gathering system is a collection of information systems, where each system is autonomous, i.e., it might be under separate and independent control and it's existence does not need to be justified by the existence of other systems; may exhibit unique hardware and/or software structures; and may operate in an open environment. There are several design issues that need to be considered in designing a cooperative information system, such as autonomy, heterogeneity and transparency.

- Autonomy: The system's entities are under separate and independent control.
- Heterogeneity: The entities of the system may exhibit technical differences between each other in both hardware and in software.
- Transparency: The system management is user independent.

Researchers in distributed database systems attempted to provide solutions for these design issues; however, these solutions are associated with several limitations such as hardwiring the system's autonomy. Researchers in distributed artificial

intelligence have considered another avenue that is based on cooperative multi-agent systems. Although cooperative multi-agent systems approach provides promising solutions for the design issues, it is attached to several assumptions, such as the existence of a master-slave relationship between the agents of the system.

1.2 Scope of the Thesis

The thesis of this research posits the development of a multi-agent modeling approach for cooperative information gathering system in open environment. The goal of this research is to develop a cooperative information gathering system (CIGS) in open domain to fulfill the users' interest.

In a cooperative information gathering system, it is assumed that the entities of the system have no global control and viewpoint. To deal with the dynamic and uncertain characteristics of the environment, it is necessary to these entities that the concepts of intelligence and rationality be incorporated. We refer to these entities as 'agents'. In general, an agent can reason about its knowledge, act on that knowledge, cooperate, and coordinate intelligently and rationally with other agents during the course of achieving global and/or individual goals.

For large-scale networks of distributed information systems with agents having limited knowledge and capabilities, what necessitates the need for cooperative retrieval and dynamic construction of responses to queries. The work presented in this thesis is based on viewing cooperation as a characteristic of an entity that has the will to help others on achieving their goals whenever it is both possible and beneficial. To allow agents to deal with other agents that might exhibit different types of behaviors, different strategies are proposed. These strategies enable agent's behavior to range from selfish to benevolent.

To deal with the main design issues for a cooperative information gathering system using agent-orientation, this thesis adopted the Coordinated Intelligent Rational Agent (CIR-Agent) model [44]. This model provides an integrated solution for the main design issues, such as autonomy, heterogeneity and transparency that support the design principles for cooperative information gathering. In the CIR-Agent model:

- Autonomy is incorporated by modeling an agent as an independent proactive entity that has the ability to make decisions concerning its own actions without any external interference. Furthermore, the existence of the agent is not justified by the existence of other agents.
- Heterogeneity is dealt with by modeling an agent as a goal-driven entity. This allows agents to interact with each other at the 'goal level' and hides the internal structure of the agents.
- Transparency is achieved by modeling an agent as a cooperative, coordinated agent that is capable of seeking and constructing connections with other agents.

To deal with autonomy, heterogeneity and transparency at different abstraction levels of the information gathering problem in open environment, this thesis proposes a three-tier architecture for cooperative information gathering systems (CIGS). At the front end of the system, the User Agents interact with the users to fulfill their interests and preferences. At the back end of the system, the Resource Agents access and capture the content and changes of the information resources. At the middle tier of the system, the Broker Agents facilitate cooperation among the agents. Each agent is equipped with coordination mechanisms. These mechanisms

include three interaction devices[1]: namely, assignment, redundancy avoidance and knowledge update.

To provide a common communication language between the agents, Knowledge Query Manipulation Language (KQML) is utilized. Thus, the semantics of the speech acts that are required by the agents during interaction are identified. To specify the conversion patterns between the agents, finite state machines are used. Finally, the proposed system that incorporates the design principles and deals with the main design issues is implemented.

1.3 Organization of the Thesis

Chapter 2 presents a literature review of the work in distributed databases and distributed artificial intelligence that is relevant to solving problems related to the main design issues (autonomy, heterogeneity, transparency). A survey of several attempts of system's architecture for information gathering is discussed. Chapter 3 presents a view of information and how it can be used to model information resources. This chapter also provides a review of some views and definitions that can serve as the basis for defining and analyzing the concept of cooperation. Chapter 4 provides a detailed description of the system architecture and the functionality, architecture and design of the individual agents. Chapter 5 describes the implementation of the proposed system, and demonstrates the feasibility of the system on different scenarios. Finally, chapter 6 highlights the key issues covered in this dissertation with a summary of the main contributions of this research and future work.

[1]Devices are means by which an agent interacts with other agents to resolve problems that are related to the type of interdependencies [45].

Chapter 2

Related Works

The task of information gathering requires locating, retrieving and integrating information that might be distributed. The main objective of this chapter is to review some of the work done in distributed databases and distributed artificial intelligence as related to the main design issues of distributed information systems. Then, this chapter provides a brief description of some of the research attempts related to agent-based architecture for information gathering.

2.1 Main Design Issues of Distributed Information Systems

Researchers in the fields of distributed databases (DDBs) and distributed artificial intelligence (DAI) have been concerned with several design issues with specific focus on autonomy, heterogeneity and transparency. The following subsections provide a brief review of some solutions related to these issues.

2.1.1 Autonomy

Autonomy is a desirable feature in distributed systems. Autonomous entities (systems) are able to perform jobs based on their importance; adapt to the needs of the users; and tolerate failures.

In distributed database systems (DDBSs), several types of autonomy have been identified [33, 93]:

- association autonomy —the ability of a database system (DBS) to decide whether and how much to share its functionality and resources with others;
- communication autonomy —the ability of a DBS to decide whether to communicate, when and how it responds to a request, with other DBSs;
- execution autonomy —the ability of a DBS to decide whether and how to handle externally generated operations.

These types of autonomy can be viewed as control for three different functions of DBSs: namely, sharing functionality and resources, communication, and executing tasks. A major problem with this view, the level of autonomy is predetermined.

In DAI, the entities are assumed to be equipped with reasoning capabilities about their environment. The Carnot project [103] has focused on relatively static environments of centralized and distributed enterprise databases where information is centrally managed. The objective of this project was to virtually unify physically distributed enterprise-wide, heterogeneous information resources. In this approach the integration of information is enabled by the use of the Cyc common-sense knowledge base [64], where the models of the enterprise are related to each other through a global context by means of articulation axioms. Similar to Carnot,

in the Infomaster project [42] the aim is to provide integrated access to multiple heterogeneous information sources; in this approach the integration of information is centrally managed by the facilitator, which harmonizes the heterogeneity among the information sources using a common schema.

Alternatively, the InfoSleuth project [9, 54] is based on a multi-agent system for heterogeneous information resources in open and dynamic environments. In this approach, all the agents of the system have to announce and update their presence, location, and capabilities to one specific type of agent, the broker agent. In this system, the agent's autonomy depends on the existence and the functionality of the other specialized agents that should be involved to accomplish the desired goal. In addition, the broker agent has the global view of all other agents in terms of their capabilities and existence. In InfoSleuth the agents that collaborate together to perform a task on behalf of a user are classified into the following types: a user agent, which uses knowledge of the system's common domain model (ontology) to assist the user in formulating queries and displaying results; an ontology agent, which provides an overall knowledge of ontology and answers queries about ontology; a broker agent, which receives and stores all the advertisements and capabilities of all agents; a resource agent, which provides mapping from the common ontology to the database schema and language native to its resource; an analysis agent, which corresponds to resource agents that are specialized for data analysis/mining methods; a task execution agent, which uses information supplied by the broker agent to identify the resources that have the requested information, routes requests to the appropriate resource agents and reassembles the results; and a monitor agent, which track the agents interactions and the task execution steps.

Another approach based on the multi-agent system has been proposed [15, 100] to provide a distributed system architecture for a large scale digital library envi-

ronment at the University of Michigan. The interaction between agents of this system is achieved through negotiation processes based on the WALRAS system, a market-oriented programming environment [73, 99]. Although, the agents of this system are designed to make their decisions based on their own perspective for providing services, the agents' autonomy depend on the presence and functionality of the registry agent to achieve their goals. In this approach, the agents are classified into the following three different types: user interface agents, mediator agents and collections agents. User interface agents encapsulate user queries into the University of Michigan Digital Library (UMDL) protocols. The user agent also publishes the user's profile to the appropriate agents, which are used by mediator agents to guide the search. Mediator agents which are further classified into: registry agents that capture the addresses and the contents of each resource; query planning agents receive queries and route them to the appropriate resources, and then collect the results; and facilitator agents to mediate negotiation among agents. Collection interface agents that perform the translation between the agents and the resources, and publish the contents of a collection that consists of a set of documents.

In MACRON (Multi-agent Architecture for Cooperative Retrieval ONline) [79] the agents are organized into a functional and a query-answering units. Each unit has a functional manager and a set of other agents. The manager agents of these units negotiate during interaction for assigning jobs; however, the agents of the functional units interact with their managers using a master-slave protocol.

2.1.2 Heterogeneity

Heterogeneity usually arises in distributed systems design because different technology might be used, different designers might be involved and/or the meaning of

the concepts might be changed over time; this results in technical and conceptual differences (heterogeneity) between the entities of the system.

For instance, in DDBSs heterogeneity can be categorized into those related to the differences in DBMSs (technical), those related to the differences in data models (conceptual), or a combination of them [67, 93]. In federated database systems (FDBS) heterogeneity at the conceptual level has been dealt with as follows. The local schemas are translated into equivalent homogeneous *component schema* using a canonical or common data model. The definitions of the available parts of these component schemas are defined as *export schemas* and finally their integration into a *federated schema* of the federation. The views that are customized for each user are defined as external schemas.

In Carnot [17, 53] integrated access to multiple heterogeneous databases is accomplished by mapping each local database schema into a global schema and vice versa. These two-way mappings are represented as a set of logical equivalencies, called articulation axioms [52]. Thus, information in the local databases can be accessed either through the view provided by global schema or through the view provided by a local database schema. This approach however is based on the assumption that the Cyc knowledge will provide a comprehensive common representation.

The Infomaster project [29, 42] provides a solution to the problem of integrating a variety of heterogeneous distributed information sources by developing different wrappers on top of each resource. The facilitator utilizes a set of rules and constraints to describe information sources and translation among these sources.

In InfoSleuth [35, 101, 102] an ontology based approach is used. In this approach, the agreement among the various agents on the terms for specifying agent

context and the context of the information handled by the agents is the function of the ontology agent. Dealing with the heterogeneity issue in this approach is based on the global view provided by the ontology agent.

At UMDL [6] heterogeneity has been dealt with using a conspectus language (CL) and protocol for agent interaction and negotiation. CL is used as a common language to describe the agents' contents, capabilities and the structure of the documents. The mediator agents using information gathered from the conspectus determine the subset of appropriate collection interface agents, which are capable of handling the query. The proposed approach in MACRON [25] has dealt with heterogeneity by designing goal-driven agents.

2.1.3 Transparency

In designing distributed systems, it is highly desirable to free users from having to identify where the information of interest is located and how to access and retrieve it.

For example, in tightly coupled federated database systems (FDBSs) location, replication and distribution transparency are provided [85, 93]. This is accomplished by developing a federated schema that integrates multiple export schemas. The transparencies are provided by mapping between the federated and the export schemas, such that the user can pose a query against either a single schema or multiple federation schemas without knowing where the requested data is located. A central federation administrator defines and manages the federated schema and the external schemas related to it.

In contrast, in loosely coupled FDBSs no global or partial static schema integration takes place and thus no transparency is provided. Instead of a central

federation administrator, the user must find the appropriate export schemas that can provide the required data and define the respective mapping operations. In the above systems, the integration of the DBSs managed either by the users of the federation or by the administrator of the FDBS together with the administrators of the DBSs. The amount of integration depends on the needs of federation users and desires of the administrators of the DBSs to participate in the federation and share their databases. Therefore, the system management is dependent on the users of the federation.

A Distributed Semantic Query Manager (DSQM) [104] has been proposed to execute queries and/or updates against integrated information resources in the context of enterprise integration. Using the articulation axioms in the architecture of DSQM was one of the main ideas to dynamically expand a query into subqueries that access all semantically equivalent and relevant information resources to be transparent to the user. In this approach, however, the Enterprise Modeling and Model Integration Software Tool is used to generate the articulation axioms when a database schema is initially integrated.

Transparency in Infomaster [5] is achieved through the facilitator that determines which sources contain the information necessary to answer the query, and provides the illusion of a centralized, homogeneous information system to the user.

Alternatively, transparency in the InfoSleuth system [81] has been achieved through the broker agents. The broker agents are responsible for identifying the capable agents that are required for handling the query received from the user agent.

In UMDL [16] transparency is achieved by allowing the agents to interact to answer a query, where the query planning agents receive queries and route them to

resources and then collect the results. Similarly, in MACRON [25] transparency is achieved through agents interaction, at which the query manager agent seeks the relevant information resources through the functional manager agents.

2.2 Information Gathering Systems

The vast growth of information space in the Internet and large scale Intranet computing environments has been a major contribution in motivating many researchers to challenge the problem of information gathering. In this section, some of these research attempts are briefly discussed.

In the InfoSleuth project [76, 77] the focus was on providing a framework for designing large-scale software applications that retrieve, fuse and analyze information in dynamic heterogeneous information systems. In this project, the strategy was based on the deployment of agent-based architecture, in which the system architecture is divided into four hierarchical layers: Agent Application, Generic Agent, Conversation, and Message layer. This classification is based on the type of the messages that each layer supports. Each agent is classified in terms of its functionality with a specialized *ad hoc* architecture that is driven by the required functionality. The notion of agenthood in this approach has not been utilized as a modeling tool for cooperative distributed systems; rather it is viewed as a developing methodology.

In the UMDL project [15] the focus was on providing an agent-based infrastructure for digital library services. In this approach, the interaction between the agents has been viewed within an economic framework as consumer-producer relationships [73]. The system architecture of UMDL is divided into three main classes including

user, mediator and collection agents. Similar to the InfoSleuth, each agent can be considered as a model with *ad hoc* architecture driven by the required functionality.

The work in [59] has developed a multi-agent system for information gathering in distributed and heterogeneous environments. This approach viewed the information gathering system as a network of information agents. The system comprises a group of agents of one type identified by information agents, where each agent wraps an information resource and has models of other agents. The goal of each agent is to provide information and expertise on a specific topic by drawing on relevant information from other information agents. The agent architecture is based on Services and Information Management for decision Systems (SIMS) [1], Loom Interface Manager (LIM) [80] and a planner called *Sage* [57].

In [66] the problem of information gathering has been viewed as distributed problem solving for which a multi-agent system is considered an appropriate solution. The system architecture comprises two types of agent units: query- and function-units. Top-level queries drive the creation of partially elaborated information gathering plans, resulting in the employment of multiple semi-autonomous, cooperative agents for the purpose of achieving goals and sub-goals within those plans. The agent architecture is based on the Generalized Partial Global Planning (GPGP) [26]. The main assumption of GPGP is that the problem can be modeled within the distributed problem-solving context. Although this assumption provides useful heuristics for closed environments, it is not realistic for open environments where generating PGP is not attainable. The RETSINA [96] and DECAF [46] projects extended these attempts at the architecture level by introducing multi-agent infrastructure. The objective in these projects is to support reusable agent types (Interface, Task Agents and Information Agents) that can be adapted to address a variety of different domain-specific problems. However, each agent inherited

the same architecture of its predecessors [25].

2.3 Summary

Researchers from DDBSs and DAI have been concerned with several design issues including autonomy, heterogeneity and transparency. From the preceding literature review it can be concluded that the solutions provided have some drawbacks. In dealing with the autonomy issue for example, the systems are considered to be deterministic in distributed database systems, whereas in most approaches of distributed artificial intelligence either there is a master slave relationship or the system depends on the existence and the functionality of one specific agent. In dealing with heterogeneity, there are some good efforts towards providing languages and protocols to facilitate interaction between systems. However, due to the dependency on having one system with a global view, the heterogeneity issue is not completely resolved. Transparency in DDBSs are user-dependent, in which users participate in the system control, whereas a multi-agent systems approach provides an appropriate solution by allowing the agents to interact and work together for achieving users' goals.

Many researchers have attempted to develop systems for information gathering based on multi-agent systems. These attempts have shown some limitations due to their initial view of the problem, such as viewing information gathering as distributed problem solving.

The main objective of this proposed research is to address these significant shortcomings by viewing information gathering in a different way. This research will provide a better understanding of the domain and utilize the appropriate technology to provide solutions for the design issues.

Chapter 3

Cooperative Information Gathering Systems

A natural starting point for designing information gathering systems is to understand the fundamental aspects of the domain: namely, data and information. This chapter discusses our understanding of information within the scope of an information gathering problem. The view of information gathering in distributed environments as cooperative information gathering is then presented. Finally, a brief review of some of the work attempting to understand the concept of cooperation is presented; followed by an analysis identifying the agent's cooperative behavior.

3.1 A View of Information

The terms 'data' and 'information' are usually used interchangeably. However, in this thesis data refers to a collection of electronic signals (bits) that can be used as a basis for computation and reasoning. Information refers to a collection of data

that has a 'meaningful' pattern. This distinction facilitates how information can be viewed in a computer-based environment.

Conventionally, information has been viewed as a physical entity. This convention may hold for systems that require intervention of humans who are able to interpret and reason about this entity; however, the aim is to design systems that are capable of reasoning about and manipulating information intelligently. To manipulate the information (i.e., to locate and/or retrieve information), it is necessary to model the information in such a way that we are able to determine the appropriate and necessary operators.

For this purpose, information is viewed as the characteristics of physical entities called information resources. One of these characteristics is *topic* and denoted by T of the information resource IR. A topic T is defined as a pattern that consists of a set of concepts, or:

$$T =< C_1, \ldots, C_n > .$$

Where C_i is a concept i, and n is the number of concepts representing T. This can be demonstrated for example on a topic such as 'weather'; 'weather' may then be represented as a set of concepts associated with a specific pattern, such as temperature and pressure. Each concept may be treated as a topic. A topic can be represented syntactically by a set of words, each of which is associated with ontological meaning. The second characteristic is quality, Q, to quantify the degree of relevance of the information resource with respect to a topic.

3.2 Document Representation

The content of an information resource can be unstructured (e.g. flat files), semi-structured (e.g. html files) or structured (e.g. databases). In this work the focus was on semi-structured information.

In a document not all the concepts are equally important to its content. Thus, importance factors (or qualities) are assigned to the concepts in proportion to their presumed importance for the document topic [83]. Then a document (d) can be represented using the vector space model [84] as:

$$d =< (C_1, Q_1), \ldots, (C_i, Q_i) >,$$

where Q_i represents the quality of concept C_i in document d.

In the vector space model, the quality of a concept is the product of its frequency (Cf), in a document and its inverse frequency (idf). The inverse frequency is used to raise the importance of the concepts that appear in fewer documents, while diminishing that of generic concepts across different topics. Therefore, the concept qualities can be calculated as:

$$Q_{d,j} = Cf_{d,j} \times idf_j,$$

where $Cf_{d,j}$ is the frequency of concept C_j in document d, and idf_j is the inverse frequency of the concept C_j in the collection of documents. One commonly used measure of the inverse frequency is

$$idf_j = log(\frac{N_D}{n_j}),$$

where n_j is the number of documents containing concept C_j, and N_D is the total number of documents in collection D; where, D is an identifier for a specific

collection. A collection refers to the context within which the inverse document frequency is evaluated. Finally, the representation of a document that contains all the concepts and their associated qualities can be extracted. Hence, information resources can be represented as a set of concepts (concepts-qualities vector) associated with their qualities as:

$$IR =< (C_1, Q_{C_1}), \ldots, (C_n, Q_{C_n}) >,$$

where Q_{C_n} represents the quality of concept C_n in the information resource IR.

3.3 Cooperative Distributed Systems

Globalization becomes a key to the success of any organization. Every organization depends on knowledge for effectively and efficiently executing its business mission. Often the necessary information to fulfill that organization's requirements is dispersed over many local/remote resources. Traditional information retrieval systems require the user to supply the query and the location of the information resource. Furthermore, in such a complex and dynamic environment it is clear that cooperation between the entities of the system for information gathering becomes a necessity. Researchers in various disciplines such as distributed databases, distributed artificial intelligence and social psychology have described the cooperation from different perspectives. The objective of this section is to review some of the research related to cooperation and how it facilitates a generic definition.

Distributed Computing Systems: Research in this field has viewed cooperation as the characteristic of a group of processors working together to balance the load between them [48]. For example, in the sender-initiated approach the

overloaded processor attempts to find an under loaded processor, whereas in the receiver-initiated approach the under loaded processor attempts to find an overloaded processor [32]. Although no definition for cooperation is given, different strategies using different algorithms were proposed. These strategies include: reducing others' loads when less loaded, and reducing local load when overloaded.

Researchers in distributed information systems have described cooperation as the characteristic of several processes' behavior within the context of the system structure [88]. There are several paradigms that are used to structure an information system; these paradigms can be classified based on which process makes the decision to cooperate. For example, in a master-slave based model a master process initiates and controls any dialogue with other (slave) processes, whereas slave processes respond to commands from a single (master) process and exchange messages when invited by the master process.

Distributed Databases: In distributed database management cooperation is defined in terms of integrating multiple schemas into a global scheme [97]. Two views have been provided for describing cooperation between databases [52]; in both views however, cooperation is described as the characteristic of the collective behavior of all databases to integrate their local schema into either a global or federated schema.

Decision Theory: In this field, cooperation and competition were considered as the two main behaviors that entities might exhibit. Cooperation has been viewed as the characteristic of the entities that try to avoid conflict, while each attempts to maximize its payoff [19, 39].

Social Behavior: There have been various approaches to the study of cooperation

in different fields such as psychology and biology. These approaches are centered on different lines of investigation, theories and research findings in the context of social behavior. We review some of these approaches in the following pages.

Experimental Based Approaches: Some approaches, such as Deutsch experiments [27, 28], have been based on external reward as the basis for viewing cooperation and competition. It has been noted that cooperation would always coexist with competition. Conversely, in the cooperative condition, if the group was successful (for example, in competition with another group), a reward would be shared equally among the members. In the competitive condition, the most successful individual in the group would be rewarded. Also, it has been observed that members of the cooperative groups are more friendly, supportive, trusting and open to one another than members of the competitive groups. In this approach, people cooperate to achieve shared group goals because they are linked to the individuals' goals. These members help each other because it is rewarding. Although no precise definition for cooperation is given, research described some of the parameters for defining cooperation, such as help and reward.

Argyle [2] argued that cooperation is important in social behavior and should not be based only on external rewards. Based on this view, cooperation is defined as acting together in a coordinated way at work, at leisure, in social relationships, in the pursuit of shared goals, in the enjoyment of the joint activity, or simply in furthering the relationship. Thus, individuals have to evaluate the situation in order to cooperate towards a common goal.

Another definition of cooperation has been proposed by Huntingford [51]. Cooperation is defined as acting jointly with other individuals or working with individuals towards a common goal. Huntingford uses animals as an example; female

lions track down prey in groups, searching and stalking together and often setting up elaborate ambushes. Following a successful hunt the prey is shared by all members of the pride [11]. Cooperation has also been defined by Marwell and Schmidt [70] as joint behavior that is directed toward a goal in which the individuals have a common interest. Each of these approaches focussed on the common interest of the individuals.

Cooperation and helping have been seen as closely related concepts [2]. The cooperation choice is based on maximizing joint benefits, whereas the helping choice maximizes the benefits of one individual. Based on this view, cooperation involves helping, but in both directions. Grzelak and Derlega [47] have also distinguished between cooperation and helping. In their study, help is characterized in terms of the unilateral dependence of individuals on others, whereas cooperation is characterized in terms of mutual dependence among individuals. Nevertheless, the instances of cooperation are viewed to be similar to those of helping [87] in that both involve the interdependence between the individuals of social intention, including the costs and benefits. In this view, both parties benefit and both experience costs, but their benefits and costs differ. The costs and benefits may take different forms such as psychological, moral, social, and/or material. Furthermore, cooperation is a relative concept and characterizing an act as cooperative between individuals may involve a value judgement. Whether an act is judged cooperative may depend on who is judging. The same act may appear cooperative to one individual and non-cooperative to another. For example, intervening to prevent a crime may be cooperative to the victim but not the offender. In these views, cost, benefit and intention aspects have been considered as the main aspects of cooperation and helping.

Another study on cooperation [71] is based on the intentions driven for posi-

tive social behavior. In this study, researchers have observed that there are a wide range of intentions of individuals underlying the positive social behavior, it is also difficult to establish an individual's intentions [34]. Dawes [20] also argues that in many situations, individuals who try to pursue private benefits may have a positive effect on others. In these studies, intention was the only aspect considered when studying cooperation.

In summary, most of the researchers in the experimental based approach have considered cost, benefits and/or intentions as the main aspects for their views and definitions of cooperation. However, the focus was devoted to individual self interest and common goals.

Theoretical Based Approaches: Some of the major theories that have been proposed to explain different kinds of social behavior are classified as: cost-benefit theories; reciprocity, equity and the just world theories; empathy, arousal and tension theories; moral standards theories; and sequential theories. These theories have been used to study cooperation in experiments and discussed in the following pages.

Cost-Benefit theories focus on identifying situations in which individuals cooperate based on external rewards. These theories suggest that individuals will cooperate if the benefits of cooperating outweigh the costs. Different dimensions have been suggested in measuring costs and benefits: effort or harm as the cost of cooperating, self blame as the cost of not cooperating, and praise from self as the benefit of cooperating [82]. Testing these theories however, requires the operationalization and measuring of the costs and benefits, or rewards and punishments [4] based on the individual's past experience.

Reciprocity, Equity and the Just World theories focus on identifying situations

in which people cooperate based on social relationships. The reciprocity theory suggests that an individual who has been helped is usually obliged to repay this help in some way [12]. Equity theory suggests that individuals will act so as to maintain equity in relationships; that is, a proper balance between a person's benefits and the efforts s/he exerts [49]. The 'Just World' theory [65] assumes that people want to believe in a world characterized by justice, where people deserve what they get and get what they deserve.

Empathy, Arousal and Tension theories describe situations in which people may cooperate based on an individual's internal state. These theories all assume that people cooperate in order to reduce unpleasant internal states. The empathy theory suggests that a person who observes someone in need vicariously experiences the other's distress, which is unpleasant [3]. The arousal theory suggests that a person who observes someone in need has an increased arousal level [82]. The tension theory [50] suggests that the existence of the unfulfilled goal gives rise to tension, which is unpleasant. Thus, reducing or eliminating these unpleasant internal states can be achieved by cooperation. However, the problem with these theories as with some of the other theories, is how to measure the internal state.

Moral standards theories attempt to explain the tendency of human morality. These theories suggest that children pass through a series of stages of moral development [8]. In the first stage, the children cooperate in order to obtain rewards or avoid punishments. In the second they cooperate in order to comply with the commands of those in authority. In the third, they cooperate as a response to other people's needs. In the fourth they cooperate to gain social approval. In the fifth, the emphasis is on reciprocity they cooperate with others hoping to gain help in the future. In the sixth they cooperate with no expectation of extrinsic reward, and altruism

is said to be a developmental achievement.

Sequential Theories address the decision processes involved in cooperation. These theories suggest that cooperation be broken into a sequence of decision stages. Latane and Darley [62] proposed a five stage process for bystander intervention. The bystander has to notice that something is happening, has to interpret it as an emergency, has to decide that it is his personal responsibility to act, has to decide what form of help to give, and has to actually help.

In summary, research in the theoretical based approach has been focused on identifying aspects such as cost, benefit and/or past experience that are required for motivating individuals to cooperate. This research however, requires the operationalization and measuring of these aspects and lacks a precise definition for cooperation.

Cooperation in DAI: In distributed artificial intelligence cooperation has been considered as a central concept for designing multi-agent systems. In this Section, some of the efforts are briefly discussed.

In one view Durfee, and his colleagues [30] treated cooperation as the characteristic of the collective behavior of several entities during the course of achieving a common decomposable goal. Each entity separately attempts to achieve its portion of the goal. Then, the entities are committed to help each other to achieve the goal. Other researchers [9, 60] have taken the same approach. Cooperation has also been viewed as the entities' behavior directed toward a common goal [68]. In this view, the entities are characterized by being rational and benevolent for achieving a common goal(s).

In another work [23], cooperation has been viewed as a fundamental property of

an agent, through which it enables the agent to work with other agents to perform a task. This property is designed implicitly and includes competition when an agent needs to compete in achieving a task. In this work, cooperation is viewed as a special case of coordination and has been described as having a structure and classified as 'trivial' and 'non-trivial' [22]. In the former (trivial), a single send and single received is involved, whereas in the latter (non-trivial) multiple exchanges are involved.

Another work [15] described cooperation as the characteristic of the collective behavior of several self-interested entities; these entities work on achieving a common goal within an economic framework [73]. The objective of each entity is to choose an activity with a maximum profit to achieve its goals, where the process of achieving this goal is supported by a negotiation strategy. Similarly, the work of Sandip and his colleagues [69, 86] describes cooperation as the characteristic of the collective behavior of self interested agents based on the principle of reciprocity, which means that agents help others who have helped them in the past or can help them in the future. According to this approach, a probabilistic reciprocity is introduced to promote cooperative behavior between self-interested agents with a fair distribution of the workload. This approach assumed that all agents know the cost of the goal by each individual and able to achieve it by their own. The agents decide whether or not to help other agents by using the values of benefits and costs. Hence, agents are promoted to help other agents only if the cost of achieving the goal is lower than the cost incurred by the agent that has to achieve that goal.

Alternatively, the work proposed in [72] has described cooperation as the characteristic of the collective behavior of several entities in a society, which are taking a benevolent action without looking for an immediate reward. These entities act benevolently while achieving their own goals given that those actions might help

other agents in the society. In another view [43] cooperation has been defined as the characteristic of the agents having the will to help each other in trading knowledge as well as in achieving their goals whenever it is beneficial and possible.

Another work [55, 56] has described cooperation as the characteristic of agents that are socially responsible. A socially responsible agent selects an action that has a joint benefit greater than the joint loss; a joint benefit is the combination of the acting agent's sole benefit, that the agent's shared benefit and other agents' shared benefit. In this view, the society's benefit becomes more of concern to the individual benefit.

In [43] cooperation is viewed as a characteristic of the entities' behavior that tends to reach mutual benefits during the course of achieving their goals. In such an environment each entity has the will to help others and is capable of evaluating the costs and benefits. Hence, a definition of cooperation similar to that of Ghenniwa's [43] is adopted, in which cooperation can be described in terms of an entity's will to help others to achieve their goals whenever it is both possible and beneficial. This definition captures the main aspects of cooperation such as help and benefit. Furthermore, this definition can be used as a base for analyzing different types of behaviors that range from self-interested to benevolent; a self-interested agent selects actions that maximize its benefits, whereas a benevolent agent selects actions that maximize others' benefits. However, the behavioristic view of this characteristic has to be designed explicitly for the entity to choose the appropriate degree of cooperation.

In summary, some of the researchers in DAI have considered cooperation as the characteristic of the collective behavior of several entities during the course of achieving a common goal, while other researchers lack precise definitions and

mixing between cooperation, collaboration and coordination.

3.4 Agent's Cooperative Behavior

The previous section has described several attempts to define cooperation. The objective of this section is to provide an analysis for cooperation based on the following definition: cooperation is a characteristic of an entity that has the will to help other entities to achieve their goals whenever it is both possible and beneficial.

In a multi-agent environment an agent may interact with other agents, each of which may exhibit a different degree of cooperative behavior. Thus, it is recognized that an agent should be equipped with different interaction strategies. To speak to these strategies formally, a quantitative representation needs to be considered. The main parameters that can be used to identify the degree of cooperation are the recipient's profit out of cooperation engagement, which can be represented as a utility function H; and the provider's profit out of cooperation engagement, which also can be represented as a utility function B. Based on these parameters the behavior of an agent can be described as benevolent, selfish, or cooperative as follows: benevolent behavior in which an agent maximizes the recipients' utility H; selfish behavior in which an agent maximizes its own utility B; cooperative behavior in which an agent maximizes the agents' utility, $\max_{all-possible-action}(B + H)$.

All of the aforementioned behaviors can be measured by an observer, reflected by overall actions or predicted from the strategies used. Although the agent's behavior can range from selfish to benevolent, the focus of this work is on cooperative behavior for information gathering. A detailed treatment of this behavior is described within the context of the agent model in section 4.4.

3.4.1 Measure of Cost-Units

The coherency of measuring cost is essential in describing the agents' behavior within the context of decision making in a multi-agent environment. This coherency can be achieved by using a standard definition of the cost unit that should be used by all agents for measuring cost. This is not an appropriate approach for open environment, because different agents might prefer to use different metric systems. For example, one agent might calculate the cost in terms of time, whereas monetary based system might be appropriate for another agent. One possible solution is to allow each agent to adopt its own metric system of cost with the ability to exchange that to another system. In the former method, each agent can have a function that maps between the local and non-local costs for every agent willing to cooperate. In the latter, a common repository or a specialized agent (e.g., a bank agent) knows the exchange rates between the agents.

3.5 Summary

In this chapter, our view of information and how it can be modeled within the context of electronic domain has been provided. Due to the distributed nature of information resources, the dynamic nature of information gathering, and the possibility that new heterogeneous information resources are added or removed, a cooperative distributed system approach is considered as a design paradigm for information gathering. From the preceding literature review that is related to cooperation, we concluded that cooperation is amorphous concept. However, some researchers view cooperation within the scope of a specific aspect, such as self interest or common goals. Although the agent's behavior can range from selfish

to benevolent, the focus of this work is on cooperative behavior for information gathering.

Chapter 4

Cooperative Information Gathering: Agent Model

This chapter describes the proposed information gathering system in detail. First, the information gathering environment is described. Next, an architecture for the information gathering system is discussed. This discussion is then followed by a description of the functionality, architecture and design of the individual agents.

4.1 Cooperative Information Gathering Environment

In an agent-based model, the cooperative information gathering environment can be described in terms of two main types of entities, as shown in Figure 4.1:

(1) a set of agents as active entities that are able to perform operations on the information and denoted by $\mathbf{AG}=\{Ag_1, \ldots, Ag_i\}$, where, Ag_j is the name of

agent $j \in \{1, \ldots, i\}$;

(2) a set of objects as passive entities that carry the information and are denoted by **OB**=$\{Ob_1, \ldots, Ob_n\}$, where, Ob_j is the name of object $j \in \{1, \ldots, n\}$. In this model for cooperative information gathering (CIG), objects are further classified into: (a) a set of information resources and (b) agent buffers.

(a) The set of information resources is denoted by $IR = \{IR_1, \ldots, IR_l\}$, each of which is characterized by topic and quality representing some sort of information. This includes public information resources that can be accessed by one or more agents and private information resources that can be processed by one agent only.

(b) Agent buffers, denoted by $AB = \{AB_1, \ldots, AB_i\}$, are an information resource that can be processed by one agent only for transferring information to and from its private information resources.

A global time line is used to model agent's actions during execution. A discrete and continuous notion of times is used to represent the execution of actions. A discrete time notion represents the start and the end points on the time line at which the execution will take place. A continuous time notion represents the time interval that lapses between the two ends.

4.1.1 The World

The world is a structural representation of the elements of the environment and the relationships among them. For example, an information resource would have a unary relationship to represent the topic it carries, such as 'weather'. The set of

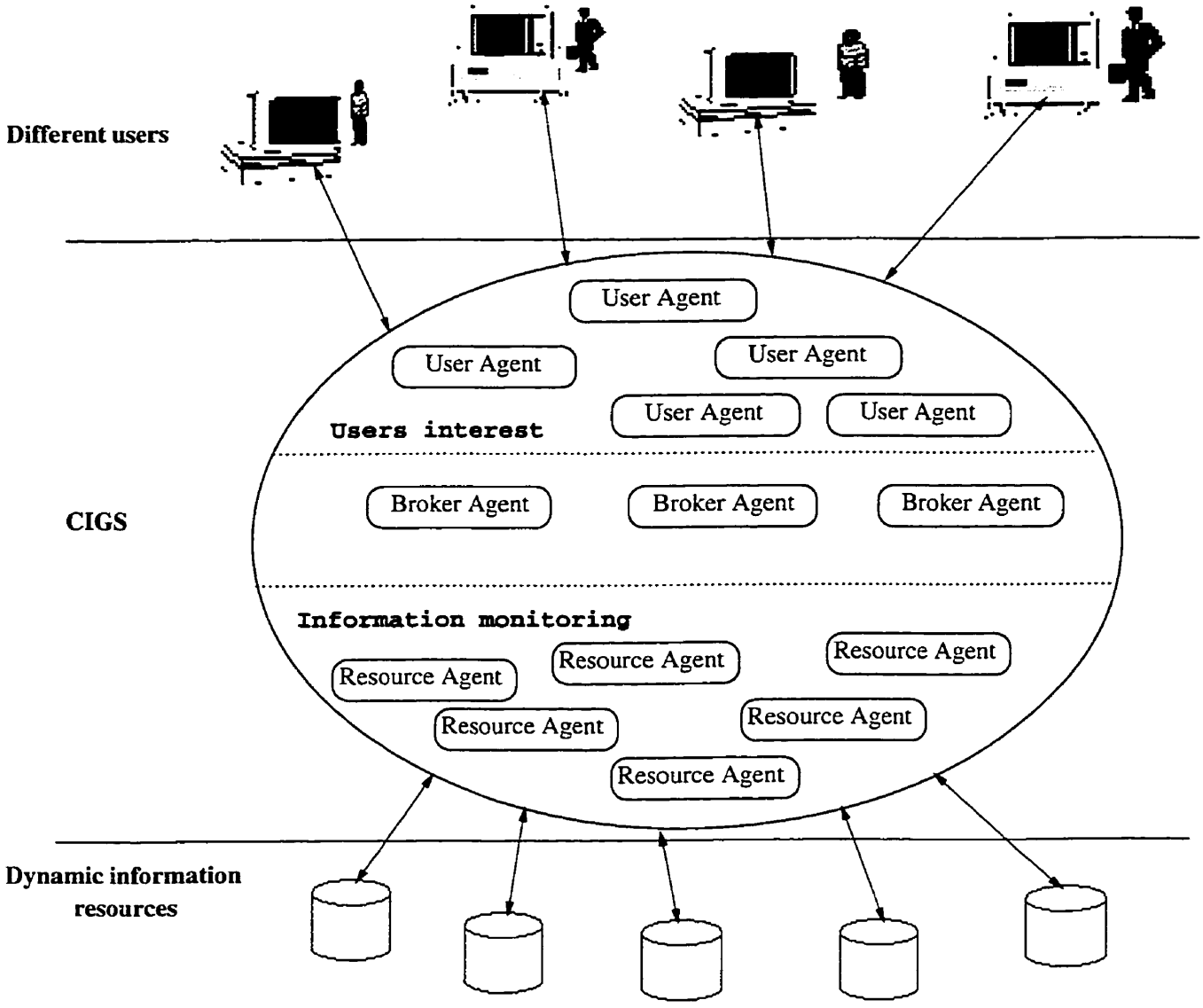

Figure 4.1: Cooperative Information Gathering Environment.

relationships that might exist between the elements of the information gathering environment can be described as follows:

- $Topic(IR_i)$, is a unary relationship on object IR_i to represent the topic of information that IR_i carries. The value of $Topic(IR_i)$ can be a set of topics.
- $Quality(IR_i, T)$, where T refers to a topic, is a binary relationship between an object IR_i and some topic T to represent the quality of information that IR_i carries with respect to T. The value of $Quality(IR_i, T)$ can be a number representing the ratio between the number of occurrences of a topic to the total number of occurrences of the top topics.
- $Auth(AB_{Ag_i})$, is a unary relationship on the buffer AB of agent Ag_i to represent that Ag_i is authorized to clear its buffer. An agent cannot clear its buffer unless it is authorized to do so. The value of $Auth(AB_{Ag_i})$ is either true or false.

The rest of this chapter is devoted to providing a detailed description of developing agent-based system architecture for cooperative information gathering. The objective is to design cooperative, coordinated, intelligent, rational agents for the environment described above.

4.2 Cooperative Information Gathering System Architecture

A cooperative information gathering system (CIGS) is a multi-agent system, in which each agent is autonomous, cooperative, coordinated, intelligent, rational and

able to communicate with other agents to fulfill the user's needs. The architecture of the system is designed to help users to locate, retrieve and integrate information from distributed resources.

A CIG system consists of three types of agents: User Agents, Broker Agents and Resource Agents, as shown in Figure 4.1. These agents communicate using Knowledge Query Manipulation Language (KQML) [37]. The user interacts with the system through a graphical user interface to submit queries and specify requests.

One appropriate system architecture for cooperative information gathering in dynamic and open environments is the three-tier architecture. At the front end, the agents (User Agents) keep track of the dynamic nature of the users. At the back end, the agents (Resource Agents) monitor the changes in the information resources content. In an open environment, new agents might join the system while existing ones might disjoin it. At the middle tier, the agents (Broker Agents) act as a 'middleman' between the agents of the other tiers and keep track of the agents that exist. The functionality of each of the agents are described in the following subsections.

4.2.1 The User Agent

The User Agent is viewed as a facility that allows the user to interact with the information system environment. The responsibility of the User Agent is to receive queries submitted from its user or other User Agents and to provide them with the relevant information. This information might be retrieved from local and/or remote resources through User Agents, Resource Agents or both.

The User Agent is capable to access local information resources, to store information for the user's future queries, and to maintain models of the other agents.

Because of these capabilities, the User Agent might be considered an information resource to other agents. The User Agent is persistent in retrieving information, in the sense that it continuously searches for more information in the user's absence.

The User Agent provides an interface to the user to interact with the information system environment. This interface allows the user to submit both the query and the desired constraints, to provide feedback, and to display the results. The User Agent accepts a query in a form described by a set of words that includes the user's topic of interest and the associated constraints. The user's interest includes both information quality and response time. The query is represented as a goal to be achieved and the associated constraints. The User Agent then generates a solution for achieving this goal which might be decomposed into sub-goals. A goal or a sub-goal might be locally achievable or require interaction with the user and/or other agents.

The User Agent interacts with the user, to learn about the user's topic of interest and preferences. The term 'user's topic of interest' refers to how the topic might be defined by the user in terms of a set of concepts, whereas the user's 'preferences' refers to the importance of each concept that defines the topic. The User Agent accordingly chooses a solution that meets the user's interest. The User Agent also learns about other agents' capabilities in terms of the topic and quality of information that they can provide as well as the time required to deliver the information.

By allowing the User Agent to interact directly with the Resource Agents or indirectly through the Broker Agents, enhances the efficiency of the CIGS when retrieves information for similar queries, and prevents CIGS from collapsing if the Broker Agent malfunctions or disappears.

4.2.2 The Broker Agent

The Broker Agent acts as a 'middleman' that pairs agents of both the front end and the back end tiers. The main responsibility of the Broker Agent is to identify and match agents based on agents' goals and interests. User and Resource Agents advertise their capabilities to the Broker Agents as they join the system. The Broker Agent accepts advertisements from the agents to confirm their existence and capabilities, and organizes those agents into groups based on topic of interest. This organization allows the agents to direct their messages and requests to the interested agents only, though an agent might associate with more than one group.

The Broker Agent also accepts notifications from the User and the Resource Agents of their unavailability at any time. The Broker Agent knows the unavailable agent either by receiving a notification message or by setting an expiration time for acknowledging the agents' existence during interaction.

The Broker Agent's responsibility has a very significant impact on the CIGS functionality. It is assumed that the Broker Agents do not interact with users. This assumption supports the *transparency* aspect of the system. Thus, neither the user nor the User Agent needs to know which are the relevant information resources. The Broker Agent knows when a new agent joins and when a member intends to disjoin the system; more agents allowed to join the system implies more information resources can exist and old ones allowed to disjoin implies information resources will not exist. Allowing agents to join and disjoin the system at any time makes the system cope well with open environments.

4.2.3 The Resource Agent

The Resource Agent acts as an information source or provider that has direct access to the information resources. The responsibility of the Resource Agent is to receive queries submitted by the User Agent(s) and return the relevant information.

Due to the dynamic nature of the environment, the contents of the information resources are constantly changing through the addition of new information, deletion of old information or modification of existing information. The Resource Agent is able to handle event notifications on the information resource updates; this means that the CIGS is able to cope with the dynamic nature of the information resources. The Resource Agent periodically monitors any change that might happen to the information resources based on a predefined time frame. This capability allows the Resource Agent to provide the User Agents with up-to-date information.

The Resource Agent accepts a query similar to that of the User Agent. This query is formulated by the User Agent and described by a set of words that include the User Agent's topic of interest and the associated constraints. The constraints might include the quality of information and the response time at which the information should be available. The relevance of the information is based on the set of concepts and the associated qualities as specified by the User Agent.

The Resource Agent might utilize the search engines as a metadata for retrieving relevant information. The relevance of the information is based on a set of concepts that define the topic of the resource. From those resources, the agent then selects those that match the user's topic of interest. The Resource Agent accomplishes this task by identifying the information resources, downloading and then extracting the semantic features (i.e., the topics and their qualities).

4.3 Agent's Architecture

The architecture of each individual agent is based on the CIR-Agent model [44]. This model provides a generic agent model for cooperative distributed systems that is appropriate for designing the CIGS. In the CIR-Agent model, an agent can be described in terms of its knowledge and capabilities.

4.3.1 The Agent's Knowledge

The knowledge of an agent includes the information that an agent has in its memory about the environment, including the self-model, the other agents' model and the domain specification.

- Self model, SK^{Ag_i}, is defined by what the agent Ag_i knows about itself, for instance the reasoning capability.
- Models of the other agents are denoted by $MK^{Ag_i} = \{M^{Ag_i}_{Ag_l} | 1 \leq l \leq m, i \neq l\}$ where $M^{Ag_i}_{Ag_l} \stackrel{def}{=} \langle X^{Ag_i}_1, X^{Ag_i}_2, \ldots, X^{Ag_i}_x \rangle$ —the definition of the Xs defines the parameters that agent Ag_i might know about agent Ag_l. These parameters might include the capability of managing information and the mental status of an agent toward achieving its goals.

Further, the domain specification includes:

- A local history of the world that consists of all possible local views for an agent Ag_i at time T denoted by $W^{Ag_i}_T$;
- A set of possible goals $G^{Ag_i} = \{G^{Ag_i}_1, \ldots, G^{Ag_i}_n\}$ where n is the number of sub-goals that might result from decomposing G^{Ag_i} —a goal $G \in G^{Ag_i}$ is a

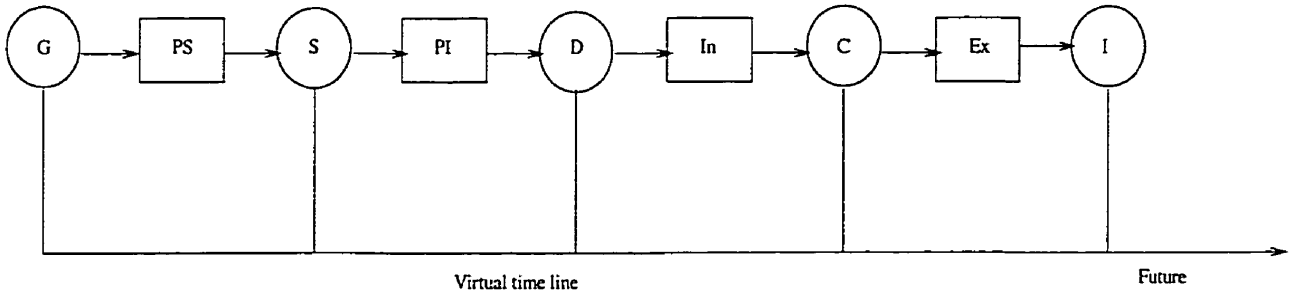

Figure 4.2: An agent's mental states.

condition that needs to be satisfied, where a condition is a local view of the world;

- A set of possible solutions, $S_G^{Ag_i}$, generated by Ag_i toward achieving a goal $G \in G^{Ag_i}$;
- An agent's, Ag_i, desires, $D_G^{Ag_i}$, toward achieving $G \in G^{Ag_i}$;
- An agent's, Ag_i, commitments, $C_G^{Ag_i}$, toward achieving $G \in G^{Ag_i}$;
- An agent's, Ag_i, intentions, $I_G^{Ag_i}$, toward achieving $G \in G^{Ag_i}$.

The Agent's Mental State

An agent's mental state refers to the agent's internal structural representation for coordination knowledge[1], local history, goals, and the reasoning activities toward achieving goals at a given time. The agent's mental state regarding the reasoning about achieving a goal must be at one of the following states: (1) Problem solving –to determine the possible solutions for achieving a goal; (2) Pre-interaction –to determine the number and the type of interdependencies; (3) Interaction –to resolve

[1]The knowledge required to identify the existing and handling types of interdependencies.

the problems associated with the type of interdependencies; and (4) Execution –to affect the world. Each of these states represents the behavior of the corresponding component of the agent, as shown in Figure 4.2.

Upon the generation of a goal from an agent or the arrival of a goal from another agent, this goal is assigned into a goal state (G). The problem solver (PS) processes this goal and transforms its mental state from a goal state into a solution state (S). That state (S) is in turn transformed into a desire state (D) by the pre-interaction (PI), then into a commitment state (C) by the interaction. Finally, this mental state is transformed into an intention state (I) by the execution (Ex). With the exception of a solution state, the mental states of the goal are defined explicitly by the agent against a virtual time line[2].

4.3.2 The Agent's Capabilities

The capabilities of the agent include communication, reasoning, and domain actions. The communication capability provides the agent with the capability of sending, receiving and interpreting messages with the other elements of the environment. The reasoning capabilities include: problem solving, pre-interaction and interaction devices.

(1) A Problem Solver —provides the agent with the capability of reasoning about its knowledge to generate the appropriate solution that is directed to satisfy a goal.

(2) A Pre-Interaction —provides the agent with the capability to determine the type and the number of interdependencies involved with each solution, and

[2]A virtual time line is a mental representation of time.

to identify the appropriate interaction devices for each type.

(3) Interaction devices —provide the agent with the capability of interacting with other agents including the user, through the communication component. These devices include:

- an assignment device —which provides the agent with the capability of delegating to other agents, goal(s) that cannot be achieved on its own;
- a redundancy avoidance device —which provides the agent with the capability of avoiding the efforts of achieving some goals that are being or could be achieved by some other agents;
- a knowledge update device —which provides the agent with the capability of learning about the other agent's capabilities and interests. This device also provides the agent with up-to-date information about the environment and the world, for which the agent knowledge will be affected.

4.3.2.1 The Problem Solver

The agents' problem solver for CIGS is designed as a goal-driven solver that hides the heterogeneity of the agent's internal structure. However, this approach assumes that the agents share partial knowledge or awareness of the goals. A goal-driven approach for an information gathering domain has a number of advantages. First, the agent might be delegated by another agent to achieve a goal without specifying how the goal can be achieved. Second, a goal-driven solver allows the agent to be equipped with different solutions for achieving a particular goal, relying on its knowledge. Third, if one solution for achieving a goal fails unexpectedly, the agent can dynamically recover and try a different solution.

A goal-driven problem solving approach can be viewed as a planning process [9, 79, 105]. In these views, the problem solver is composed of two main modules: decomposition and planning. Knoblock *et al.* [57, 58] have developed a planner called *Sage* for processing queries. *Sage* is a modified version of partial order planning that can handle conflict about reusable resources.

An agent's problem solver capability can be defined as the ability to reason about the goal with respect to the agent's domain actions and local history. Formally, the problem solver is a function that maps goals, information gathering actions, and local history into solutions ($PS : G^{Ag_i} \times Ac^{Ag_i} \times W_T^{Ag_i} \longrightarrow S_G^{Ag_i}$), where a solution is a temporal ordered list of information gathering actions.

The agents are characterized by a set of domain actions, denoted by $Ac = \{a_1, \ldots, a_l\}$. These actions are based on the view of information discussed in section 3.1 and 3.2. There are four domain actions that are necessary and sufficient for the information gathering domain. These actions include: retrieve, save, clear and decompose.

(1) *Retrieve* —enables an agent to make the topic and the quality characteristics of its buffer equal to the specified ones.

(2) *Save* —enables an agent to make the topic and the quality characteristics of some information resource equal to that of its buffer.

(3) $Clear_{AB_{Ag_i}}$ —enables an agent Ag_i to make the topic characteristic of its buffer AB equal to '$EMPTY$'. $EMPTY$ is a predefined constant.

(4) $Decompose_T$ —enables an agent to define a topic into subtopics using a binary tree structure.

Domain-action	Preconditions	Postconditions
Retrieve	$Topic(IR) = T$, $Quality(IR, T) = Q$ $Topic(AB_{Ag_i}) = EMPTY$	$Topic(AB_{Ag_i}) = T$, $Quality(AB_{Ag_i}, T) = Q$
Save	$Topic(AB_{Ag_i}) = T$ $Quality(AB_{Ag_i}, T) = Q$	$Topic(IR) = T$, $Quality(IR, T) = Q$
$Clear_{AB_{Ag_i}}$	$Auth(AB_{Ag_i})$	$\overline{Auth(AB_{Ag_i})}$, $Topic(AB_{Ag_i}) = EMPTY$
$Decompose_T$	$Topic(IR_1) = T_1$, $Topic(IR_2) = T_2$	$Topic(IR) = T$, $Quality(IR) = Q$

Table 4.1: A formal description for information gathering actions for agent Ag_i.

A formal description for each of these actions is given in Table 4.1. Executing a domain action results in a change of the worldview, where a domain action $a_j^{Ag_i}$ is read as action j of agent Ag_i. The change of the world is described by the elements of the postconditions of the action $a_j^{Ag_i}$. Note that the $Decompose_T$ action assumes the structure of a decomposable query represented as a binary tree. The information gathering goals are classified into two types as summarized below.

(1) *Agent goal* (A-goal) is denoted by $Topic(IR) = T$. A goal may or may not be locally achievable.

(2) *User goal* (U-goal): this type of goal cannot be satisfied by an agent, such as authorize to clear the agent's buffer, $Auth(AB_{Ag_i})$.

The problem solver consists of two main parts which: (a) identify the type of the goal as A-goal or U-goal; (b) determine the required actions for achieving each goal of type A-goal or U-goal. A solution for information gathering goal, G, by agent, Ag_i, is a temporal ordered list of information gathering actions, denoted by $S_G^{Ag_i} = \{a_1^{Ag_i}, a_2^{Ag_i}, \ldots \mid a_j^{Ag_i} \in Ac^{Ag_i}\}$.

4.3.2.2 The Pre-Interaction

The pre-interaction process of the agent estimates the cost and the possible scheduling time for each solution and transforms the plan (solution) that minimizes the expected cost into a desire. The selection process is based on the number, the type of interdependencies, and the characteristics of the interaction devices that will be used for resolving the problems associated with each type of interdependency.

In the pre-interaction process, an agent identifies the number and type of interdependencies involved for each solution candidate, i.e., reasoning about '*which*'. Then, an agent rationally anticipates the characteristics of the devices that will be used for resolving the problems associated with each type of interdependency, i.e., reasoning about '*how*'. For example, when an agent, Ag_i, problem solver processes a goal G (i.e., $Topic(IR) = weather$), to be transformed into a solution $S_G^{Ag_i} = \{Retrieve, Clear_{AB_{Ag_i}}\}$, the reasoning about '*which*' might indicate the existence of capability interdependency for which this goal should be delegated to another agent. Since actions are represented in terms of preconditions and postconditions, the existence and the number of interdependencies and their associated types can be identified during the process of reasoning about the elements of the preconditions and postconditions of the actions. After this process of reasoning that generates the number and type of the interaction devices, reasoning about '*how*' determines the types of heuristics that will be used by the interaction devices identified.

4.3.2.3 The Interaction

In this stage, the agent interacts with other agents to resolve problems of interdependencies that may arise due to different capabilities of agents, decomposition of

goals, and common goals that are associated with the agent's desires.

In the CIG domain the agents are equipped with three types of interaction devices: the assignment, the redundancy avoidance and the knowledge update. These devices are bounded in duration, in the sense that the device is restricted to find a solution against a query within a specified duration. These devices are also bounded in solution quality (i.e. the device is restricted to finding a solution against a query with a desired quality). Each type of these devices is identified in terms of three basic characteristics: (1) *problem specification*, describing the type of problems that need to be resolved; (2) *evaluation parameters*, describing the measures for the possible solutions; and (3) *sub-processes*, involved in finding an appropriate solution for the interdependency problem. The following subsections describe the assignment, the redundancy avoidance and the knowledge update devices that an agent is equipped with.

4.3.2.4 The Assignment Device

In the CIGE, it is possible for an agent to have limited capabilities related to gathering information; consequently, that agent may require assistance from other agents. This assistance can take the form of the delegation of an appropriate agent to achieve a goal. The characteristics of the assignment device are described below.

The problem specification is described as a query to be assigned to another agent. A query is a set of attributes that include the goal and the associated constraints. The goal is described in terms of the topic and quality of information. The time constraints might include desired satisfying time at which the information should be available and the expiration time after which the information is not valuable.

The evaluation parameters describe the characteristics of the other agents (models of the others) that are related to the problem specification. The evaluation parameters can be determined using the *contracting* approach [21, 95] in which the following two phases are required.

- The announcement phase, is the phase in which the agent (manager) sends out the problem specification to the other agents (or the potential contractors) as an announcement. One possible strategy for announcement is *focusing*. This strategy assumes that an agent is capable of identifying the set of potential contractors. Another possible strategy for announcement is *broadcasting*. Using this strategy, all other agents are considered as potential contractors.

- The bidding phase, is the phase in which the manager receives from each potential contractor a bid to be used as evaluation parameters. At the same time, each potential contractor tags the solution associated with that goal as a desire in their local schedules. The evaluation parameters might include: the quality of information to be satisfied; the possible starting time that represents the earliest possible time for a contractor to satisfy the query; the reservation time or the time interval required by an agent to achieve the contracted query to be reserved as a time frame specified by the earliest possible starting time within its local schedule; the cost that needs to be paid to the contractor for achieving the goal.

The subprocesses involve the selection of the appropriate contractor. The selection heuristics are based on assigning aspiration levels for the quality of information, reservation time, possible starting time, and the cost. The agent assigns priority levels to these heuristics. In one possible situation, the manager assigns the highest priority to meeting the reservation time, if the bids submitted by the contractors

have a reservation time that cannot be met by the manager, then they are rejected regardless of the information quality, possible starting time, or cost.

The manager selects the contractors with bids that have acceptable quality. If more than one contractor has been selected, then the selection will proceed based on the possible starting time, reservation time, and so on for the rest of the heuristics used. If there is no time left for the assignment, the agent selects the contractor randomly. Finally, the agent sends out an award message to the selected (winner) agent and a dismiss message to the unselected (loser) agent(s). When the winner agent receives the confirmation message, the status of its mental state of the contracted query is transformed from a desire state to a commitment state. Whereas the loser agent deletes the contracted query and invokes the local scheduler to free the time frame reserved for the contracted query as a desire.

4.3.2.5 Redundancy Avoidance Device

In the CIGE it is possible that more than one agent will attempt to achieve the same goal. Consequently, an agent might achieve a goal that will be achieved by other agents. To avoid the efforts of achieving goals that are being, or could be achieved by some other agents is the main function of the redundancy avoidance device. Although the main function of the redundancy avoidance device is similar to that of the assignment device, the goals in the former can and possibly will be achieved locally. This may require an agent to select one of the other agents (partners) to achieve the goal. The selection process can be based on negotiation. The characteristics of redundancy avoidance are discussed below.

The problem specification is described as a query that should be sent to other agents. The time constraints of this query might include the desired satisfying time

and holding period during which the goal remains satisfied.

The evaluation parameters for the above problem specifications might include the other agents' desires during the interval specified by the desired satisfying time and holding period. There are two phases required to determine the evaluation parameters.

- In the first phase, the agent sends out the problem specification to the other agents (or the potential partners) as a request, which might include topic, quality, desired satisfying time, holding period, and 'desired'[3].
- In the second phase, each potential partner responds back after it determines its part of the evaluation parameters that might include topic, quality, and status, where status $\in$ {(interested, mental state), (not interested)}.

The subprocesses involve the selection of the appropriate partner. After receiving the evaluation parameters, the agent determines its partners that are interested in achieving the goal and that have a desire as a mental state. Then, the agents engage in the process of negotiation. Since agents are characterized as being cooperative (i.e. they have the willingness to help each other whenever it is both possible and beneficial) one negotiation strategy that can be used is to help the others with the right to *opt out* of the negotiation. This strategy allows the agent to pursue its desires locally.

4.3.2.6 Knowledge Update Device

In CIGE it is possible due to the nature of this environment (i.e., dynamic and open), for an agent to have outdated information over time. There might be a

[3]desired is a keyword to indicate the mental state of the agent toward achieving the corresponding goal.

change in the agent knowledge about the information resources, other agents' capabilities and goals or the user's topic of interest. Thus, in order for an agent to cope with this environment the agent is required to update its knowledge; this update might have two different forms. In the first form, an agent learns about the elements of the environment when it interacts with those elements using assignment and/or redundancy avoidance. For example, agents know other agents' interest when they receive requests from them. In the second form, an agent tries to seek information from other agents (partners). The characteristics of the knowledge update device are described as follows.

The problem specification is described as a query to be sent to other agents. The goal is described in terms of the topic whereas the constraints of this query might include either information quality or other agent's goals.

The evaluation parameters that are related to the above problem specification can be determined using the following two-phase approach:

- the seeking phase, in which the agent sends out the problem specification to the other agents (partners) as a request, which might include the topic and the quality of information;
- the informing phase, in which each partner responds after it determines its part of the evaluation parameters that might include topic, quality and status, where status $\in$ { (exist, quality), not exist}.

The subprocesses involve the following processes. After receiving the evaluation parameters, the agent determines the status of the information. Then, the agent sends out an acknowledgement message to the partner that responded, at the same time an update on the information is performed.

Recall that actions are represented by a set of relationships as preconditions and postconditions respectively. Any domain action performed by an agent causes a change in the status of these conditions. The knowledge update accordingly updates the agent's knowledge about the current view of the world.

4.3.3 The Agent's Local Scheduler

A local schedule is a time-indexed agenda for the agent's desires, commitments and intentions. A local scheduler then, is a mental processor that enables the agent to assign or allocate the mental states of goals on the virtual-time line and produces a local schedule. For instance, when the problem solver produces the set of possible solutions for a goal, it interacts with the local scheduler to determine the possible scheduling time for each solution. Then by selecting the best solution as a desire the pre-interaction invokes the local scheduler to update the local schedule accordingly. The selected solution is tagged as a desire by assigning a time frame on the local schedule according to the time stamp associated with it. The time stamp represents the possible location on the local schedule to the chosen solution as a desire. If this desire was pursued later during interaction, its time frame might be adjusted because of its interaction with other agents' desires and commitments, and then tagged as commitment. Finally, an agent during the execution state affects the world by transforming the commitment into intention.

4.3.4 Execution

The execution is a mental processor that enables the agent to operate on the local schedule. An agent uses this processor to check the time stamps put on the local scheduler against the real time clock. For example, if the time stamp of a goal that

is tagged as a desire on the top of the schedule equals the value of the real time clock, then the desire is destroyed, allowing the time reserved for it to be freed. If the time stamp of the goal is tagged as commitment and equals the value of the real time clock, then the execution starts and the commitment is transformed into intention for which the world will be affected.

4.3.5 Communication

Communication is the essential means by which the agents cooperate and coordinate in order to achieve their goals. In a decentralized, interconnected entities[4] environment, there are different types of communication in multi-agent systems: Agent to Human, Agent to Agent, Agent to Non-Agent, and Agent to Environment.

- Agent to Human —agents might communicate with a human through different forms, such as textual dialogs. Agents that communicate with humans are usually named as interface or user agents. They serve the users by accepting queries and getting the results back to them.
- Agent to Agent —communication between agents can be established through the communication components of their architecture. They exchange messages using predefined mechanisms and protocols.
- Agent to Non-Agent —agents might also be able to communicate with non-agents (objects), such as databases through their names and addresses.
- Agent to Environment —the capability of an agent to communicate with the environment that might include the type of operating system that it runs on.

[4]Entity refers to software agent, human agent or object

The above description makes it apparent that there are three distinct issues that are fundamental for the agents' interaction with other types of entities: a common communication language to establish the communication between the entities; a common understanding that is necessary (i.e., the ontological commitments between them); and communication protocols and mechanisms to enable the multi-agent system to exchange information for which the agents coordinate and cooperate with each other. The communication component of an agent is divided into four layers in a way well-suited to an agent's interactions with other entities, as shown in Figure 4.3. The description of these layers follows.

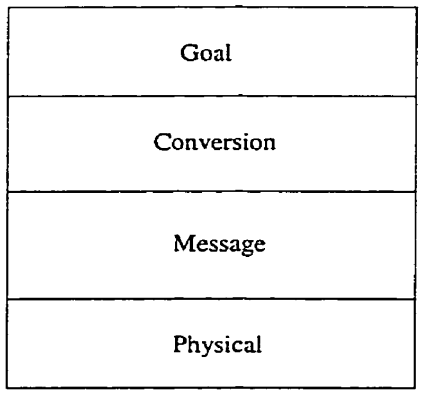

Figure 4.3: An abstract model for communication layers.

The Physical Layer —provides a uniform interface to the underlying layer, which is the physical layer. The interface hides the low-level details of the connection by providing, to the upper layers, an abstract view of several basic operations needed by the agent. The physical layer receives all messages sent by other entities of the environment and sends messages to other agents formed at the message layer.

The Message Layer —maps between the physical layer and the conversion layer, dis-

cussed next. The CIGS agents use KQML for inter-agent communication because it has been proposed as a standard communication language and is commonly used in the multi-agent community [36]. In this layer, there are two main operations that can be performed by the agent. First, the agent constructs the outgoing messages in the form of KQML. Second, the agent parses the incoming messages from the physical layer as a result of messages exchanged with others. There are three functionalities that are encapsulated during these two operations, which are described as follows: (1) consistent use of KQML syntax by the agents involved to (a) ensure the creation of valid KQML messages requested by the conversion layer, and (b) ensure valid KQML messages received from others through the physical layer; (2) handling failures during parsing by generating error messages; (3) generating the appropriate messages for the type of communication protocol to be used.

The Conversion Layer —provides a well-formed language (in terms of structure) that will ensure the sending and receiving of the intended messages between the agents. The main function of this layer is to determine how to transmit/accept messages effectively without misunderstanding. A set of standard messages is used to serve as the bases for carrying the dialog of conversion. The conversion policy might take the following two different forms: (1) a predefined conversion, which is defined a priori and forces the agents' communication to follow a very specific order of exchanging messages; (2) an emergent conversion, in which the agent uses a dynamic order of messages, based on the interpretation of the received messages [38].

A Goal Layer —maps between the internal/external goal(s) to be achieved by the agent and the conversion layer. As soon as the agent identifies the type of the goal, the type of the interaction device (assignment, redundancy avoidance or knowledge

update) is also identified. Accordingly, the conversion mechanism becomes available at the conversion layer as related to the device to be invoked. The semantics of the speech acts that are used by the interaction devices of the agents in CIGS are described below:

- Announce —used to send the problem specification to one or more agents that might be able to achieve the goal, when the assignment device is invoked.
- Request —used to send the problem specification to one or more agents that might be able to achieve a goal, when the redundancy avoidance is invoked. It is also used to seek information when the knowledge update is invoked.
- Offer —used to send a bid for achieving a goal to a previously announced problem specification (assignment device), or a response for achieving a goal to a previous request (redundancy avoidance).
- Counter-Offer —used to refine a previous offer in an ongoing negotiation.
- Accept —used to signal an acceptance of a previously received offer.
- Opt-Out —used to signal rejection of a previously received offer during negotiation.
- Reject —used to signal rejection of an announcement or a request.
- Award —used to send a confirmation to an agent that has been awarded to commit itself to pursue achieving a goal.
- Dismiss —used to send a cancellation to agent(s) that were not awarded for a previously received bids.

- Inform —used to send a response to previously received request –knowledge update.
- Success —used to confirm the achievement of a previously committed goal.
- Fail —used to signal the failure in achieving a goal.

The CIGS uses finite-state machines (FSMs) to specify the conversion patterns between the agents. All conversion polices are identified based on the interaction devices and the initial message used. Other researchers have used similar approaches [18, 38, 78]. Given that the types of interaction devices at the communication level are known, and given that every possible conversion for each type an agent engages in is also known, the finite state machine (FSM) to model these conversions can be then represented.

The FSMs representation for the assignment device are shown in Figure 4.4 and Figure 4.5. Figure 4.4 shows the FSMs representation for an agent when it acts as a manager and Figure 4.5 shows the FSMs when an agent acts as a potential contractor. The FSMs representation for the redundancy avoidance device while an agent acts as a requester is shown in Figure 4.6, whereas an agent acting as a partner is shown in Figure 4.7. The FSMs representation for the knowledge update device while an agent seeking information from other agents is shown in Figure 4.8, whereas Figure 4.9 shows the FSMs for an agent when it is informing other agents. Following the approaches used in [7, 10, 40] all the states of the finite state machine representations are mapped into a set of rules. These rules are descriptions of what an agent does in a certain situations based on certain events.

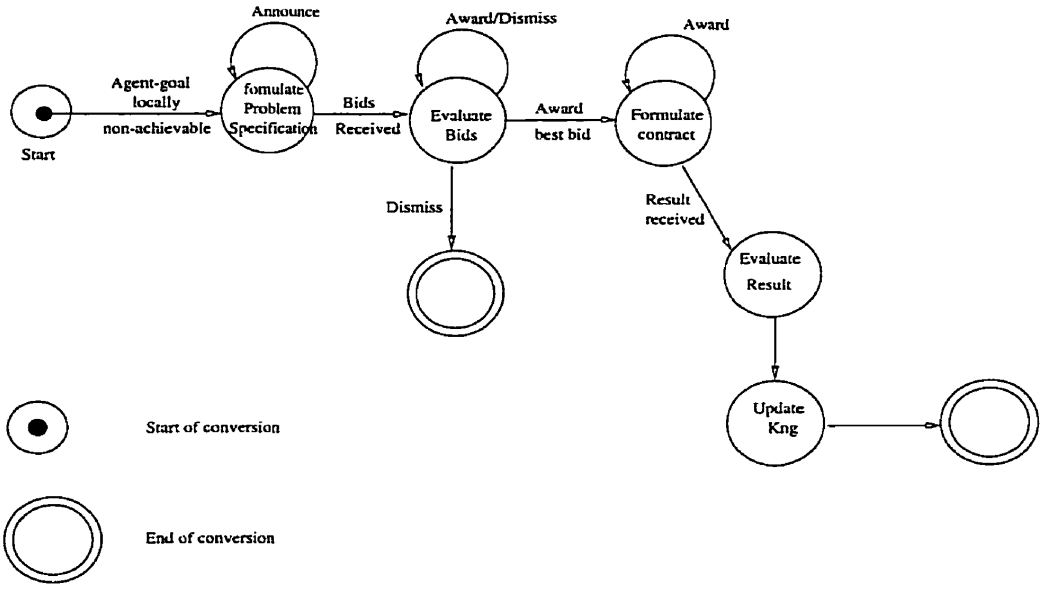

Figure 4.4: Finite State Representation of a Manager.

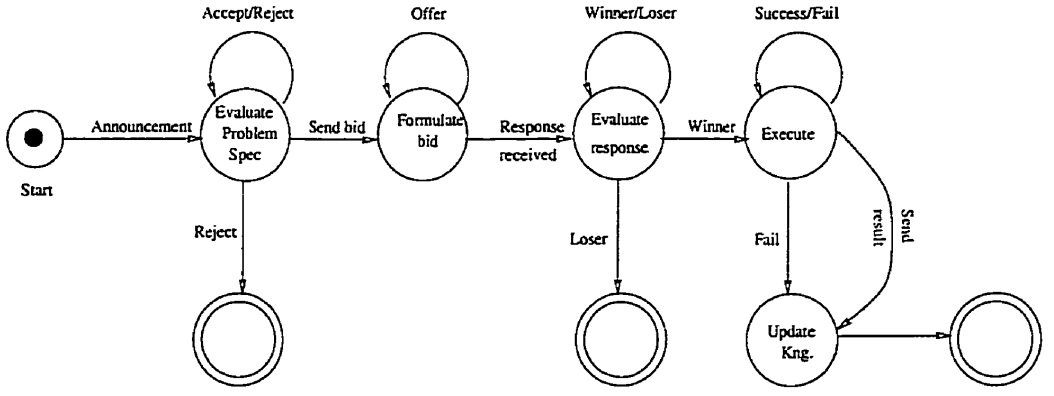

Figure 4.5: Finite State Representation of a Contractor.

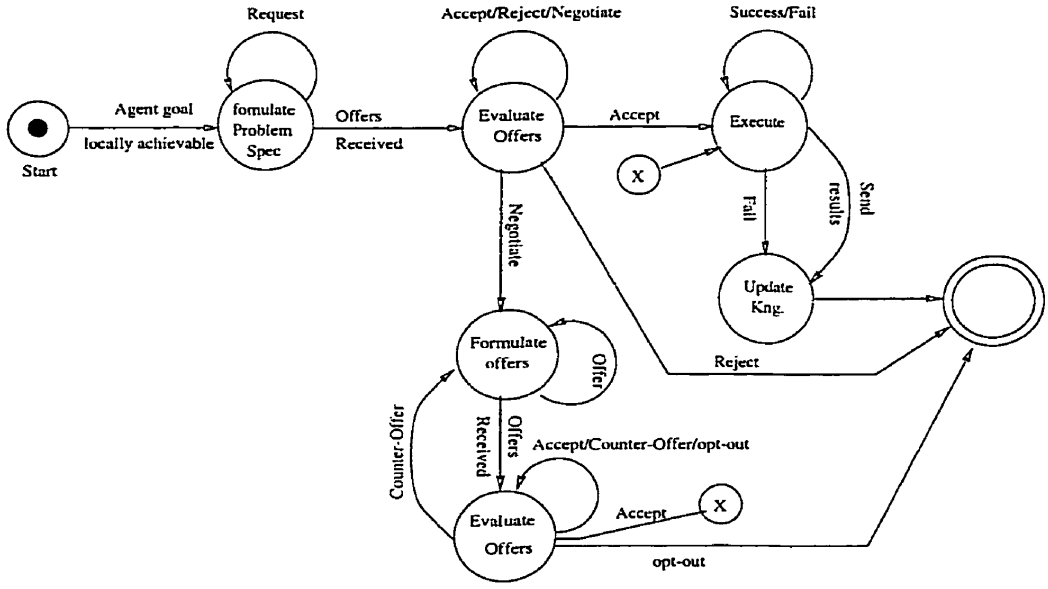

Figure 4.6: Finite State Representation of a Requester.

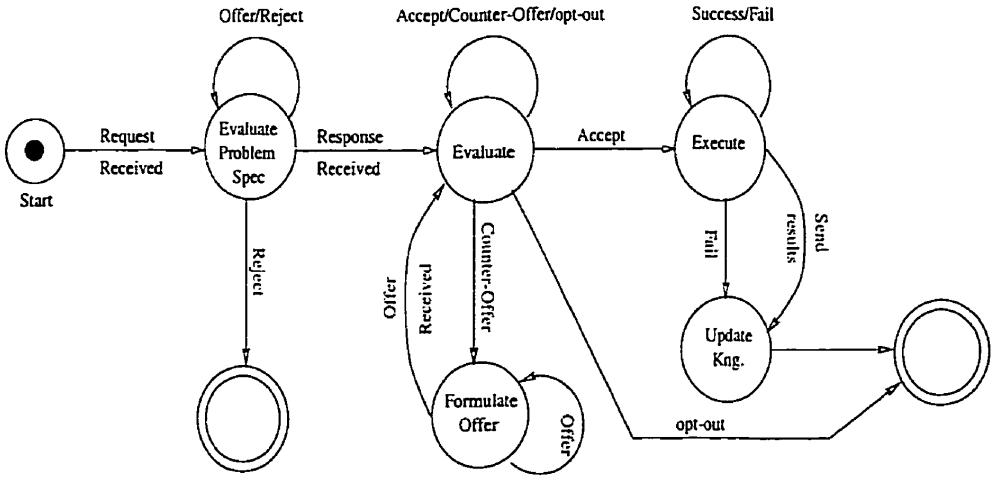

Figure 4.7: Finite State Representation of a Partner.

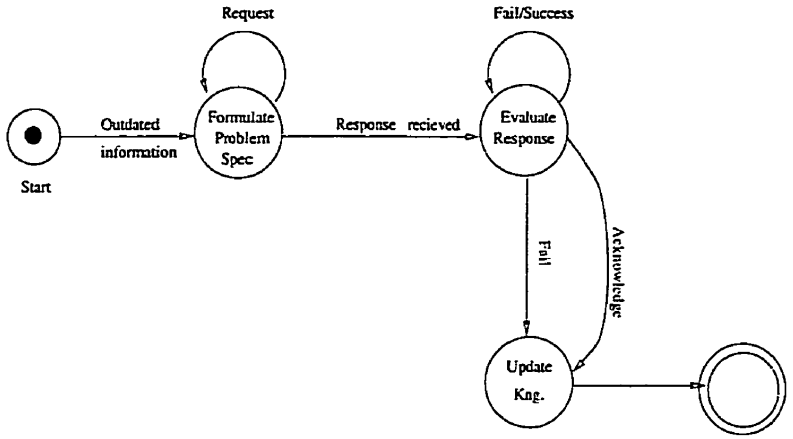

Figure 4.8: Finite State Representation of a Seeker.

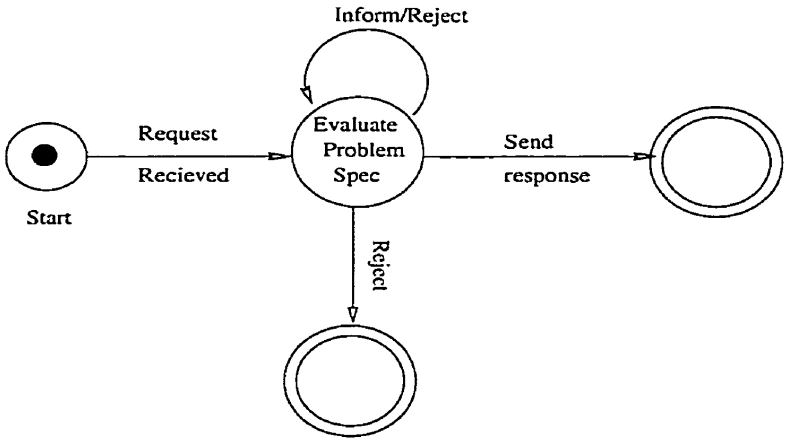

Figure 4.9: Finite State Representation of a Informer.

4.4 Cooperation

Cooperation is essential in multi-agent systems to ensure the behavior that governs the interaction between the agents. Different strategies can be used to provide the agent with the ability to exhibit the desired behavior. To design such strategies, the types of the agent's behaviors are explained.

- During the assignment, an agent (requester) might be required to assign a goal to other agents (providers).

  A requester agent using *Selfish-driven* strategy:

  - If ($MK \neq \phi$) (* i.e. the agent has experience with other agents*) Then
    - Use focusing strategy for announcing problem specification
    - Select agent(s) that can provide a utility higher than or equal to the expected utility B
  - Else
    - Use multicasting strategy for announcing problem specification
    - Select agent(s) that can maximize the local utility.

  A provider agent using *Selfish-driven* strategy:

  - If (the local utility can be maximized) Then
    - Set the local utility (* The local utility is set based on the expected utility H of the requester agents *)
    - Make a bid with the local utility
    - Allocate goals at the tail of the scheduler

- Else
  - * Ignore the request.

A requester agent using *Benevolent-driven* strategy:

- If ($MK \neq \phi$) Then
  - * Use focusing strategy for announcing problem specification
  - * Select agent(s) that can provide a utility higher than or equal to the expected utility B
- Else
  - * Use multicasting strategy for announcing problem specification
  - * Select agent(s) that maximize the local utility B.

A provider agent using *Benevolent-driven* strategy:

- Determine the expected utility H of the requester
- Set a utility that satisfies the requester's desire
- Make a bid that maximizes the requester's utility
- Allocate goals on top of the local scheduler as they arrive.

A requester agent using *Cooperative-driven* strategy:

- If ($MK \neq \phi$) Then
  - * Use focusing strategy for announcing problem specification
  - * Select agent(s) that can provide a utility higher than or equal to the expected utility B
- Else

* Use multicasting strategy for announcing problem specification
* Select agent(s) that maximize the local utility as well as the others' utility $B + H$.

A provider agent using *Cooperative-driven* strategy:

- Determine the utility values of the requesters
- If (the local utility as well as the others' utilities can be maximized) Then
  * Set the local utility based on the others' utilities.
  * Make a bid with the expected utility (* i.e. maximizes local and others' utility $H + B$ *)
  * Allocate goals at the desired satisfying time on the local scheduler
- Else-if (others' utilities can be maximized) Then
  * If ($MK \neq \phi$ and history = 'good[5]') Then
    * Set the expected utility
    * Make a bid with the expected utility
    * Allocate goals at the desired satisfying time on the local scheduler
  * Else-if ($MK \neq \phi$ and history $\neq$ 'good') Then
    * Ignore the request
  * Else
    * Make a bid with the expected utility

[5]good –is an indicator to the agent's self-satisfaction about another agent in terms of help provided.

    * Allocate goals at the earliest possible free time slot on the local scheduler

  - Else Reject.

- During redundancy avoidance, an agent might delegate some goals to other agents to avoid redundant effort. An agent invokes this device when it is aware that the other agent(s) might be interested in achieving the same goal.

A requester agent using *Selfish-driven* strategy:

  - Determine the others' utilities H
  - Select agent(s) that can provide the highest local utility B
  - Make an offer based on others' utility
  - If (Offer received contains the expected utility or higher and there is no time left to make a counter offer) Then
    * Accept the offer
    * Allocate goals at the desired satisfying time on the local scheduler
  - Else-if (there is enough time to negotiate)
    * Repeat
    * Generate a counter offer using discounting strategy
    * If (Offer received contains the expected utility) Then
      · Accept
      · Allocate the goal at the desired satisfying time
    * Until the time set for negotiation expires

- Else
  - Generate a counter offer with a minimum utility and use take-it-or-leave-it strategy
  - If (Offer received contains the expected utility) Then
    - Accept
    - Allocate the goal at the desired satisfying time
  - Else Reject.

A partner agent using *Selfish-driven* strategy:

- If (Offer received contains the expected utility or higher and there is no time left to make a counter offer) Then
  - Accept the offer
  - Allocate goals at the desired satisfying time on the local scheduler
- Else-if (there is enough time to negotiate)
  - Repeat
  - Generate a counter offer using discounting strategy
  - If (Offer received contains the expected utility) Then
    - Accept
    - Allocate the goal at the desired satisfying time
  - Until the time set for negotiation expires
  - Else Reject
- Else
  - Generate a counter offer with a minimum utility and use take-it-or-leave-it strategy

    * If (Offer received contains the expected utility) Then
        - Accept
        - Allocate the goal at the desired satisfying time
    * Else Reject.

A requester agent using *Benevolent-driven* strategy:

- Make an offer based on the local utility
- If (offer rejected) Then Pursue the goal locally
- Else
    * Accept the offer
    * Allocate goals at the desired satisfying time on the local scheduler.

Note that, in the benevolent behavior, if an agent selects an agent as partner and receives an offer it accepts that offer without negotiation.

A partner agent using *Benevolent-driven* strategy:

- Accept offers without negotiation
- Allocate goals at the desired satisfying time on the local scheduler as specified by the requester agent.

A requester agent using *Cooperative-driven* strategy:

- Determine the others' utilities H
- Select agent(s) that can provide the highest local utility B
- Make an offer based on others' utilities
- If (Offer received contains the expected utility or higher and there is no time left to make a counter offer) Then

    * Accept the offer
    * Allocate goals at the desired satisfying time on the local scheduler
  - Else-if (there is enough time to negotiate)
    * Repeat
    * Generate a counter offer using discounting strategy
    * If (Offer received contains the expected utility) Then
      · Accept
      · Allocate the goal at the desired satisfying time
    * Until the time set for negotiation expires
  - Else
    * Opt out
    * pursue the goal locally.

A partner agent using *Cooperative-driven* strategy:

  - If (Offer contains the expected utility or higher and there is no time left to make a counter offer) Then
    * Accept the offer
    * Allocate goals at the desired satisfying time on the local scheduler
  - Else-if (there is enough time to negotiate)
    * Repeat
    * Generate a counter offer using discounting strategy
    * If (Offer received contains the expected utility) Then
      · Accept
      · Allocate the goal at the desired satisfying time

* Until the time set for negotiation expires

- Else
  * Opt out
  * pursue the goal locally.

4.5 User Agent Architecture

The main characteristics of the User Agent (UA), shown in Figure 4.10, are knowledge and capabilities. The knowledge of UA includes self-model, models of the other agents and the domain specification. The models of the other agents (User

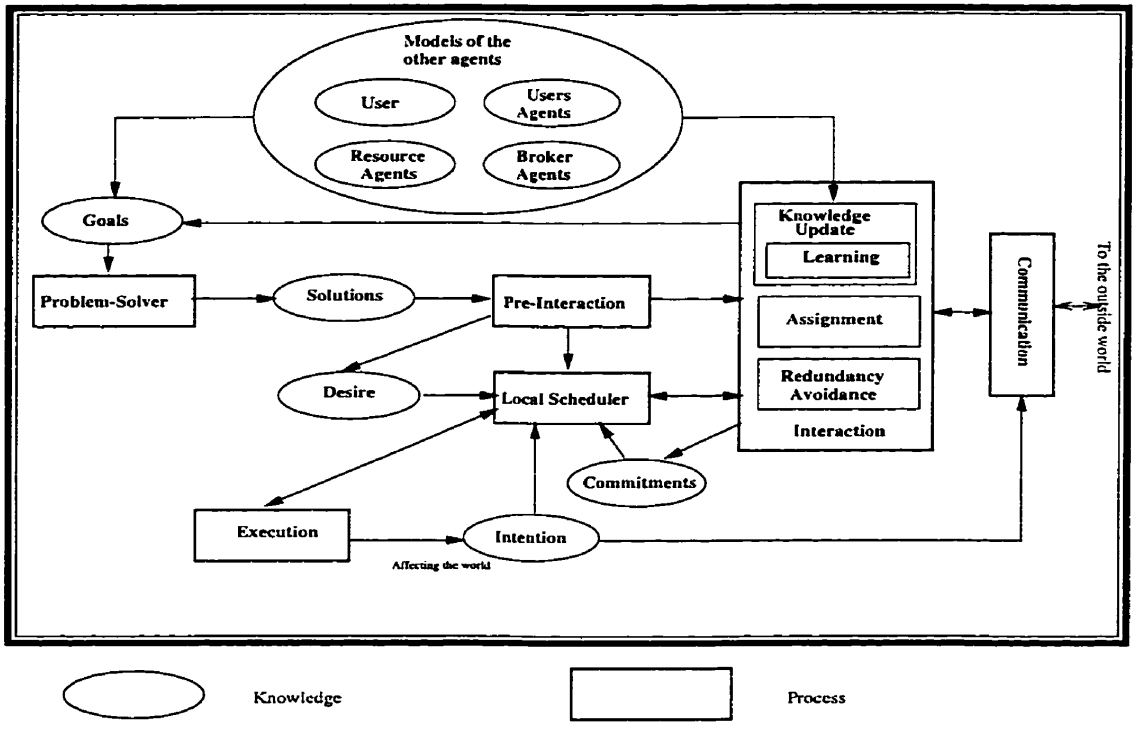

Figure 4.10: User Agent Architecture.

Agents, Broker Agents and Resource Agents) can be represented in terms of their

capabilities, whereas the model of the user is considered in terms of his/her topics of interests and preferences. The domain specification includes:

- The local history, denoted by W_T^{UA}, at a given instance of time includes the information available locally and the user's topic of interest and preferences.
- A set of goals denoted by $G^{UA} = \{G_1^{UA}, \ldots, G_n^{UA}\}$. These goals are directed toward gathering information. For example, when a user submits a query to plan for a vacation, the goal can be represented as $Topic(AB_{UA}) = vacation$. The UA then decomposes the goal into sub-goals during the problem solving based on how vacation is represented.
- A set of possible solutions, S_G^{UA}, toward achieving a goal $G \in G^{UA}$. Once the goal has been identified the possible solutions for achieving this goal are generated.

The UA acquires and builds a model of the user in terms of the user's topic of interest. The user's topic of interest can be represented as similar to that of the information resources, explained in section 3.2. The user's interest in a specific concept within the topic is represented as a set of concepts (concepts-qualities vector) and the degree of relevance that identifies the user preferences over the concept. Formally, the user's topic of interest can be represented as:

$$U =< (C_1, Q_1), \ldots, (C_j, Q_j) > . \tag{4.1}$$

The UA adapts to the dynamics of users by monitoring the changes of the user's topic of interests and reacts accordingly by modifying the concepts-qualities vector that represents the user's interest in a topic. When the user's interests change, the content of this vector is replaced by updated concepts and qualities, which will serve the new user interests.

The feedback mechanism used for updating the concepts-qualities vector is similar to that used in [94]. The user provides feedback in response to the agent request, which is positive or negative. Then, the concepts-qualities vector is modified utilizing reinforcement learning approach. These qualities are modified based on the learning rate α and the user feedback f as follows:

$$Q^U_{C_i} = Q^U_{C_i} + f \times \alpha \times Q^{IR}_{C_i}, \tag{4.2}$$

where $Q^U_{C_i}$ is the quality of concept C_i as described by the concepts-qualities vector of the user's interest and $Q^{IR}_{C_i}$ is the quality of the same concept in the concepts-qualities vector describing the information resource IR.

In order to compute the learning rate automatically, the agent monitors the number of queries containing the concept C_i, and the number of times a user expresses an interest in C_i within a predefined number of queries, η. Then, the agent calculates α online as follows:

$$\alpha = \frac{1}{1 + e^{-\rho\eta}}, \tag{4.3}$$

where ρ indicates the sensitivity of the learning rate related to the number of times a user expresses interest in a concept, and α indicates the sensitivity of qualities as related to the user feedback. Preferences of concepts can be then based on the combined values of $f\alpha$. This gives a sigmoid function in which the learning rate increases as the user interest on specific concept increases. For example, for a given positive f, if the value of α is α_1 for a concept C_1 and α_2 for a concept C_2 where $\alpha_1 > \alpha_2$ then $C_1 \succ C_2$ (read as C_1 is preferred over C_2). Conversely, for a given negative f with the value of α is α_1 for a concept C_1 and α_2 for a concept C_2 where $\alpha_1 > \alpha_2$ then $C_2 \succ C_1$.

The UA is constrained in delivering information within a bounded time, for which it needs to model other agents in terms of their response time to its query.

The UA also builds its belief about other agents' capability in terms of the information that they might provide. When the UA receives a query from another agent, it records the information required as the requester agent's interest. To elaborate, assume that UA_1 receives a query from UA_2 requesting information about 'transportation' with a quality of x. Given that this information is not available at the local resources of UA_1, UA_1 presumes that UA_2 will have this information later from some other agents.

The problem solver of the UA obtains the goal from the submitted query or the goals generated from decomposable ones and arrives at a solution that best fits the user's needs. The problem solver consists of two main parts. In the first part, the type of the goal is identified as A-goal or U-goal. In the second part, the required actions are determined. The UA is characterized by the following domain actions:

(1) *Retrieve*: to make the topic and the quality of the UA's buffer equal to the topic and quality identified in the query;

(2) *Save*: to make the topic and the quality of the retrieved information to be equal to the topic and quality of the UA's buffer;

(3) $Clear_{AB}$: to make the topic of the agent's buffer equal to '$EMPTY$';

(4) $Decompose_T$: to decompose a topic T requested by a user or other agents into subtopics based on the other agents' topic of interest.

For example, consider that the goal of UA is to retrieve information about the topic 'weather', i.e., $Topic(AB_{UA}) = weather$. This type of goal is identified as an A-goal and the UA would apply *Retrieve* action to achieve it. But there is an unsatisfied precondition (i.e., $Topic(UA_{AB}) = EMPTY$). In order to satisfy this precondition, a $Clear$ action is required to make the topic characteristic of its buffer

equal to $EMPTY$. Now it is clear that for the UA agent to apply *Clear* action, it needs to satisfy the precondition $Auth(AB_{UA})$ for which an authorization from the user is required, and the type of goal is identified as a U-goal.

The UA consists of three interaction devices namely assignment, redundancy avoidance and knowledge update.

- The assignment device is invoked when the UA cannot achieve the goal of type A-goal by itself. Thus, the UA agent delegates the goal to another agent. This can be accomplished either by announcing the problem specifications to specific agents (Resource Agents and UAs), or broadcasting through the Broker Agent.

- The redundancy avoidance is invoked when the UA can achieve the goal by itself, but it is possibly more beneficial when achieved by another UA that might or will achieve the same goal. For example, let UA_1 be responsible to gather information about 'weather' at time t. Also, UA_1 is aware that UA_2 is interested in the same information, which can be denoted by $M_{UA_2}^{UA_1} = \{topic(IR) = weather, t\}$. Hence, UA_1 and UA_2 might coordinate their efforts to avoid the redundant effort.

- The knowledge update is invoked when the world is affected. For example, when UA executes a *Retrieve* action the characteristic of the buffer is changed from $EMPTY$ to those describing the information retrieved. UA also invokes the knowledge update to capture new or updated information about other agents.

UA communicates with users through a graphical interface, using simple dialogues that handle predefined set of messages. It communicates with other agents using KQML.

4.6 Broker Agent Architecture

The Broker Agent (BA) characteristics, as shown in Figure 4.11, include knowledge and capabilities. The knowledge of BA includes self-model, models of the other agents and the domain specification. BA might have models of the other agents in

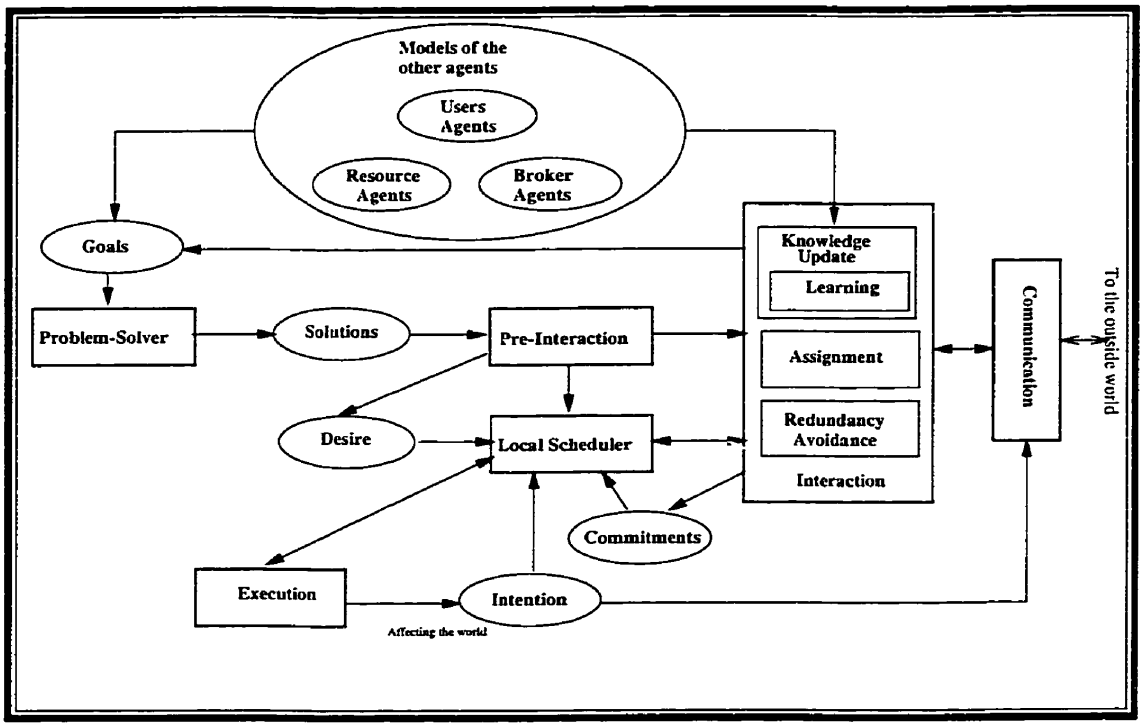

Figure 4.11: Broker Agent Architecture.

terms of their type and capabilities. These agents include UA, BA and Resource agents. The domain specification includes:

- The local history, denoted by W_T^{BA}, which might include the information about the existing UAs, BAs and Resource Agents;
- A set of A-goals, denoted by $G^{BA} = \{G_1^{BA}, \ldots, G_n^{BA}\}$, and directed to identifying the capable agents to achieve them;

- A set of possible solutions, S_G^{BA}, toward achieving a goal $G \in G^{BA}$.

The problem solver of the BA can be viewed as a search algorithm to determine the available User and/or Resource Agents who are able to carry on the requested goal. The BA consists of the following interaction devices.

- The assignment device is invoked when the BA cannot achieve the goal by its own, i.e., neither Resource Agents nor User Agents have advertised their capabilities to achieve the requested goal. Accordingly, it might assign this goal to another BA.
- The redundancy avoidance is invoked when the BA is aware that another BA might be interested to achieve the same goal, given there is enough time for the BA to be involved in negotiation.
- The knowledge update is invoked when the world is affected, e.g., new agents joined or disjoin the system.

The communication component enables BA to receive, send and interpret all messages. The BA communicates with other agents including UAs, BAs and Resource Agents, using KQML.

4.7 Resource Agent Architecture

The Resource Agent (RA), as shown in Figure 4.12, has characteristics which include knowledge and capabilities. The knowledge of RA includes self-model, models of the other agents and the domain specification. The RA might have models of the other agents based on their type and capabilities. These agents include BAs that

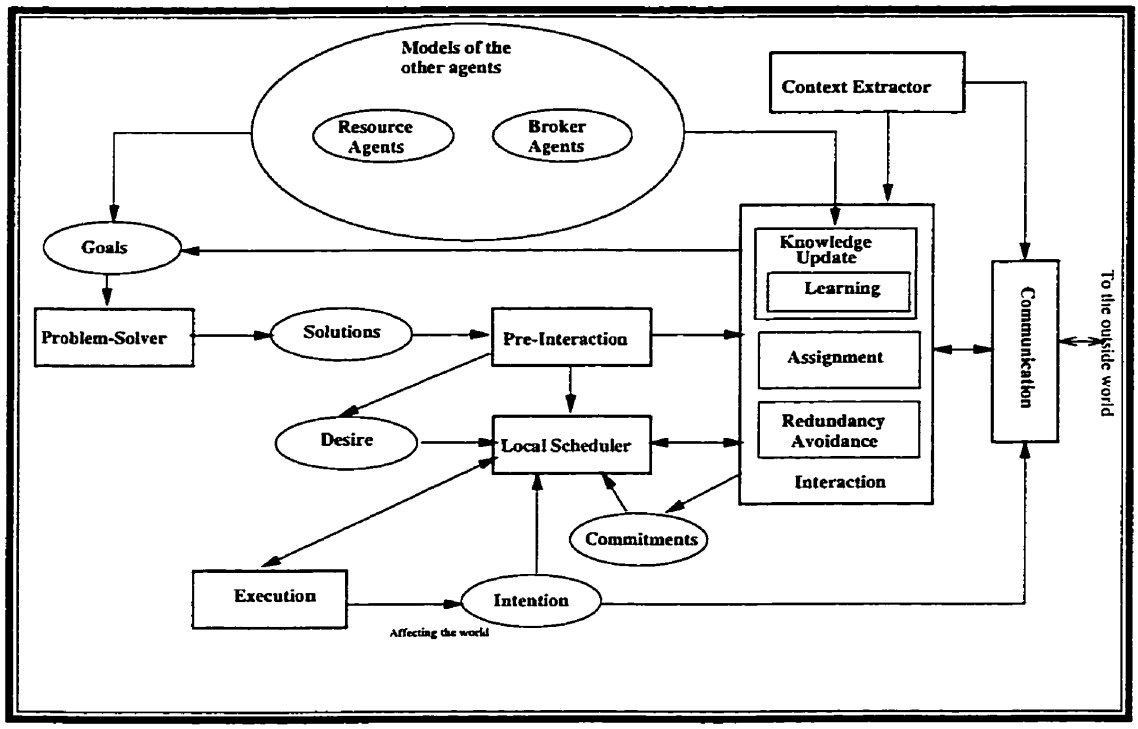

Figure 4.12: Resource Agent Architecture.

are needed to advertise to them their existence and capabilities, and other RAs that might be able to provide help for achieving some goals. RA has models of the objects that are able to access them such as databases. The domain specification includes:

- The local history, denoted by W_T^{RA}, at given instance of time includes the information resources available locally;
- A set of A-goals, denoted by $G^{RA} = \{G_1^{RA}, \ldots, G_n^{RA}\}$, and directed toward retrieving information;
- A set of possible solutions S_G^{RA} toward achieving a goal $G \in G^{RA}$.

RA consists of a context extractor as one of its components. This component provides the RA with the capability of extracting documents representations that might match the user's topic of interest. Each retrieved document is represented by a concepts-qualities vector, as described in section 3.2. This vector to be obtained by the RA through a full analysis of the document. The quality of each concept depends on its frequency in the document and the number of documents it appears in. Then, the RA builds a model of the document (object) in terms of topics and qualities, and other attributes such as last-update time stamp and location. Recall that, the user's topic of interest can be represented as:

$$U =< (C_1, Q_{C_1}), \ldots, (C_j, Q_{C_j}) > . \tag{4.4}$$

Using this representation, the similarity S between the information resource IR and the user's topic of interest U can be measured in terms of the scalar product of their concepts-qualities vectors as:

$$S(IR, U) \quad = \sum_j Q^{IR}_{C_j} \bullet Q^{U}_{C_j}. \tag{4.5}$$

In order to have a unified scale to compare and add the qualities between the user's topic of interest and different information resources, each quality of both vectors is normalized using the form $\frac{Q_{C_j}}{\sqrt{\sum_{vector}(Q_{C_j})^2}}$. With the normalization, the similarity function produces values from 0 to 1. This function produces the highest value ($S(IR, U) = 1$) only when the user's topic of interest and the information resource representations are identical –the information resource is 100% relevant to the user's topic of interest.

RA problem solver consists of a module that is required to determine the required actions for achieving A-goals, denoted by $Topic(IR) = T$. The RA is characterized by the following two domain-actions:

(1) *Retrieve*: to make the topic and the quality of the RA's buffer equal to the topic and quality required;

(2) *Save*: to make the topic and the quality of information resource to be equal to topic and quality of its buffer.

The interaction component that RA has consists of three devices: the assignment, redundancy and knowledge update.

- The assignment device is invoked when the RA is unable to achieve a goal on its own, for which it requires assistance from other agents. Due to the main functionality of this agent, this device is more likely to be invoked when UAs require assistance in providing information.
- The redundancy avoidance is invoked when the RA possesses the knowledge that another Resource Agent might be interested in achieving the same goal. This device is also invoked when other RAs send requests to avoid redundant effort.
- The knowledge update is invoked when the world is affected (e.g., when a manipulation of an information resource has occurred).

The RA is able to send, receive and interpret messages through the communication component. The RA communicates with other agents including UAs, BAs and RAs, using KQML. The RA also communicates with objects using their native language, such as SQL databases.

4.8 Summary

To deal with the dynamic, distribution and open environment for information gathering, this chapter discussed a cooperative information gathering environment. Then, multi-tier agent-based system architecture was described. This architecture consists of three-tiers. At the front end, the User Agents keep track of the dynamic nature of the users. At the back end, the Resource Agents monitor the changes in the information resources content. At the middle tier, the Broker Agents act as a 'middleman' between the agents of the other tiers.

The functionality, design and architecture of each agent's type were described. The architecture of all agents is based on CIR-Agent model, each of them exemplifies a particular architecture to reflect its functionality. Each agent was described in terms of its knowledge and capabilities. The agent's knowledge contains the information that it has in its memory about the environment and the expected world. The agent's capabilities include communication and reasoning. The reasoning capabilities include the following components: (1) problem solver, (2) pre-interaction, (3) interaction, and (4) execution. These components act as transformation process on the mental state of the goal. Furthermore, the interaction component consists of three devices: assignment, redundancy avoidance and knowledge update.

Chapter 5

Implementation and Results

This chapter presents a detailed description of a prototype implementation of the agents' knowledge and capabilities in CIGS. Then, a set of scenarios of a distributed information systems environment is used to demonstrate the feasibility of the proposed architecture for CIGS.

5.1 Introduction

The agents of the proposed architecture are implemented using the IBM Agent Builder Environment (ABE) [13, 14]. The ABE provides the essential tools that are required to develop and build some of the components of the agents of the CIGS. The ABE supports a rule based, forward chaining inference engine as the reasoning mechanism; it also provides a library of flexible functionalities to create and maintain rules and facts. The knowledge base representation is based on (Knowledge Interchange Format) KIF [41], which facilitates knowledge interchange independent of the implementation technology.

The ABE also provides a set of adapters that are ready-to-use that can be used for specific requirements, such as event detection facility, time alarm to trigger rules for time constraints, and a file adapter that facilitates monitoring and manipulating files.

All the agents in CIGS are implemented as a stand-alone Java application. To achieve our goal in providing a high performance system, explicit thread management is used to support concurrency of the agents' components on the same process space. Thus, the User Agent can perform other activity, for example accepting another query from the user while waiting for responses from other agents.

5.2 Knowledge

The agents of CIGS are written in Java and provide access to relational databases via the Java Database Connectivity (JDBC) and Microsoft's Open Database Connectivity (ODBC). This allows the agents to query and access information from local or remote relational databases through one common platform-independent interface. The definition of topics that are related to the user's view and preferences, models of the information resources and models of other agents are implemented as tables. Each agent is able to access and manipulate a set of tables including:

- $resourcemodel^A$: contains topic name, topic quality, constraints on topic, the address of the topic, last-update and summary from the document[1];
- $definition^U$: contains topic name, the user's view and preferences toward the topic[2];

[1] A indicates each individual agent has this type of table.

[2] U indicates only the user agent has this table.

- $otheragentsmodels^A$: contains agent name, agent address, benefit, cost, exchange times, average quality, and average response time;
- $otheragentsgoals^A$: contains agent name, agent address, topic name, desired time for achieving a goal;
- $otheragentsinterests^A$: contains topic name and address;
- $querymonitor^U$: topic name and number of times the user reflect an interest on a topic.

The local history of the world is implemented as lists of events that consists of the postconditions of the executed actions and time that can be used to describe the status of an object, for example $(Topic(IR) = weather, Quality(IR) = 50, Time = 1:00am)$ or the status of an agent buffer as $(Topic(AB) = EMPTY, Quality = 0, Time = 2:30)$.

The implementation of the algorithms used to extract information resources representations are performed at different stages as follows:

1. **For** i = 1 **to** N_D
   (a) Read IR_i
   (b) Parse IR_i
   (c) Remove all punctuation marks
   (d) Eliminate commonly used words, such as *for* and *the* using stop-words list
   (e) Extract all concepts (C^{IR_i}) in IR_i and the frequency of each concept (Cf^{IR_i}) and store;

End-i

2. **For** i = 1 **to** N_D

   For j = 1 **to** C^{IR_i}

   (a) Compute the value of $n_{C_j^{IR_i}}$, the number of documents that each concept $C_j^{IR_i}$ appears in

   (b) Evaluate idf using this formula $idf_{C_j^{IR_i}} = log(\frac{N_D}{n_{C_j^{IR_i}}})$

   (c) Calculate Q_{IR_i,C_j} as $Q_{C_j}^{IR_i} = Cf_{C_j}^{IR_i} \times idf_{C_j^{IR_i}}$

   End-j

   (a) Sort Q^{IR_i} in descending order

   (b) Truncate to the top m[3] concepts and normalize, for all truncated concepts

   End-i

3. **For** i = 1 **to** N_D

   For j = 1 **to** C_m^{IR}

   For k = 1 **to** C_n^U

   (a) Calculate the similarity such as $S(IR_i, U) = \sum Q_{C_j}^{IR_i} \bullet Q_{C_k}^U$, only if $C_j^{IR_i} = C_k^U$.

[3]For computations reasons the vectors truncated to 20 from the highest weighted concepts.

5.3 The Problem Solver

The problem solvers are implemented as a rule-based system utilizing the RAISE engine provided by the ABE. Rules and facts are all represented using KIF. An example of a rule and the required conditions to fire the *Retrieve* action is shown in Figure 5.1. The facts are of three types. Long-term facts are built and stored persistently. Examples of this type are related to the definitions of a concept provided by the user. Short-term facts are originated from adapters through trigger events or sensors such as user's current action or the time allowed for search. The third type is derived facts, which are generated from a successfully fired rule such as the required topic of information resource and/or its associated quality.

```
=> (AND (AND (AND (AND (AND (AND (AND (AND (AND (AND (AND      (EventName \"AGENT:CSCHANGE") " +
"(UserName ?name)) " +  "(UserActivity ?request))" +  "(UserActivityCheck ?request \"STARTSEARCH\")) " +
"(topicRequested ?tReq)) " +  "(qualityRequested ?qReq)) " +  "(topicAvailable ?tAva)) " +  "(qualityAva ?qAva)) " +
"(queryConstraint ?qConst)) " +  "(topicCompare ?tReq ?tAva)) " +  "(qualityCompare ?qReq ?qAva)) " +
"(Retrieve ?tAva ?qAva ?qConst))" );
```

Figure 5.1: A Retrieve action.

The problem solver of the CIGS agents determines the required actions for achieving the goal. With the exception of the User Agent's problem solver has to identify not only the required actions but also the type of the goal either as user or agent goal.

5.4 The Pre-Interaction

The pre-interaction component selects the *solution*, which minimizes the expected cost as a *desire* for achieving the goal. This component has been implemented as a

class. In this implementation it consists of the following: the *Which* method that is responsible for determining the type and the number of the interdependencies involved for each solution. This method is activated during the problem solving for each solution. The *how* method is responsible for identifying the type of the interaction devices and the desired heuristics to be used for resolving the problems associated with each type of interdependency. This method is activated after the *which* method is completed.

5.5 Interaction

The interaction of the User Agent with the user is based on a set of dialogues. These dialogs are build using Java's Abstract Windowing Toolkit (AWT) components. On the other hand, agents interact with each other through assignment, redundancy avoidance and knowledge update devices. Each device is identified in terms of three basic characteristics: (1) *problem specification* describing the type of problems that need to be resolved, (2) *evaluation parameters* describing the measures for the possible solution and (3) *sub-processes* involved in finding an appropriate solution for the interdependency problem.

5.5.1 Assignment Device

The implementation technique of the assignment is based on the modeling approach for the other agent's capabilities and the soliciting approach for determining the local schedule of the other agents, utilizing the contracting approach [95]. The assignment device is implemented as a class that extends Thread. This provides the agent with the flexibility to invoke this device and engage in interaction with

Topic	Quality	Starting-time	Reservation-time	Cost

Table 5.1: Evaluation parameters for the assignment.

multiple agents at the same time. In this device, the following functionalities are implemented.

The *outgoing* method formulates the problem specifications as a query that includes $Topic(IR)$=T, where T takes a topic such as 'weather', the $Quality(IR) = Q$, where $Q \in (0, 1)$, the desired satisfying time, and the expiration time for achieving the goal. Focusing strategy is used for announcing the problem specifications, if the potential contractors are determined using models of the other agents' capabilities. Otherwise multicasting strategy is used for announcing the problem specifications. In both cases, this method sets an alarm. In the first, the alarm is activated by either the arrival of the corresponding bids or the expiration time, whereas in the second the alarm is activated only by the expiration time. Then, all received bids are pooled into a bids-list. As soon as the method is activated it selects the best bid. Using the evaluation parameters shown in Table 5.1, the selection heuristic of the best bid for an agent is as follows:

(1) Remove all bids from the bids-list with starting time greater than or equal the expiration time;

(2) Remove all bids with reservation time that are less than or equal a threshold from the bids-list;

(3) Remove all bids from the bids-list with topic not equal to the topic of interest;

(4) Remove all bids from the bids-list with quality less than the quality of interest;

(5) Select bids with the least cost;

(6) If the cardinality of the bids-list greater than one, then select the highest quality bid(s) as the bids-list;

(7) Pick the first bid in the bids-list as the winner.

The *incoming* method is responsible for receiving queries. It enables the scheduler to assign the appropriate time frame to the goal. Then, the goal is passed to the problem solver and the incoming process goes to sleep. As soon as the pre-interaction determines the estimated cost for this goal, it awakens the corresponding incoming process. If this goal can possibly be achieved within the desired satisfying time, before the expiration-time, then the incoming method submits the bid back accordingly and sets the mental state for this goal as 'desire'.

The *award/dismiss* method is activated based on the outgoing process concerning the selection of the best bid. In this method, dismiss message is issued for each unselected bidder (loser). For the selected bidder (winner) a contract form is created, and then an award message is sent to the corresponding contractor.

A *winning* process is created when the award message is received. When an agent receives an award message it pulls out the corresponding contract and the mental state status of the goal is transformed from 'desire' to 'commitment'.

A *dismiss* process is created at the time when the dismissing message arrives. When an agent receives a dismiss message, the contract that is related to this message is destroyed, at the same time the local scheduler is invoked to free the time frame reserved for the desire of the corresponding contract.

Topic	Quality	Status	Desired-satisfying-time	Holding-period	Cost

Table 5.2: Evaluation parameters for the redundancy avoidance.

5.5.2 Redundancy Avoidance Device

The implementation technique of the redundancy avoidance is negotiation based. This device is implemented as a class that extends Thread, which provides the agent with the flexibility to engage in interaction with more than one agent at the same time. In this device, the following functionalities are implemented.

The *outgoing* method formulates the problem specifications as a query that includes $Topic(IR) = T$, the $Quality(IR) = Q$, the desired satisfying time, the holding period, and the state of the goal as a *desired* for achieving the goal. Then, the query is sent out in the form of the problem specifications to the potential partners. The potential partners are determined using models of other agent(s) goals. This method sets an alarm that might be activated by either the arrival of the responses or the expiration time. All received responses are pooled in a responses-list. As soon as the process is activated, it selects the appropriate partners at which the negotiation process is activated. Using the evaluation parameters shown in Table 5.2, the selection heuristic of the appropriate partners for an agent is as follows:

(1) Remove all responses with status not interested or interested with a mental state not-desired from the responses-list;

(2) Remove all responses from the responses-list with desired-satisfying-time later than the desired;

(3) Remove all responses from the responses-list with a holding-period smaller than that of the agent;

(4) Remove all responses from the responses-list with a topic not equal to that of the interest;

(5) Remove all responses from the responses-list with a quality less than that of the quality of interest.

The *incoming* method is responsible for receiving queries. It enables the agent to evaluate its part of the evaluation parameters that includes $Topic(IR)=T$, $Quality$ $(\ IR\ ) = Q$ and its status. As soon as the status of the goal is determined it sends the response back accordingly.

The *negotiation* must find an offer acceptable to all partners. The adopted negotiation strategy reflects the agents' cooperative behavior, as discussed in section 4.4. In this implementation, the strategy used is based on the willingness to help with the right to opt-out from negotiation. Agents engage in the negotiation process for who will carry on the job. An opt-out agreement is reached when no offer is 'acceptable'. After an agreement is reached, the mental state of the goal is transformed from 'desire' to 'commitment'.

The discount factor is determined using the time left for negotiation, the time required for sending a counter offer, the time required to be received, and the time required to evaluate the offer. The discount heuristics for counter offers are dynamic, because these values change depending on the status of the communication channels, the time required from the partners to respond and the time left for negotiation.

5.5.3 Knowledge Update Device

The knowledge update device is implemented with two main classes. *kngUpdateT* class extends a Thread and responsible for seeking and informing other agents. *kngUpdate* class is responsible for updating the agent knowledge.

The implementation of *kngUpdateT* consists of the following two constructs. The *seeking* construct is implemented with a set of functionalities and responsible for formulating the problem specification as a request and sending it out. Depending on the information required, the corresponding method will be activated. These methods set an alarm based on a predefined time set for knowledge updates. The alarm may be activated either by the arrival of a response or the expiration time. As soon as this method is activated, it initiates the corresponding method in *kngUpdate* to update the appropriate parameters.

The *informing* construct is responsible for receiving requests. Based on the type of the request the corresponding method is activated, for which the agent will be able to determine its part of the parameters identified in the request. As soon as the status of these parameters is determined the response is sent back.

The implementation of the *kngUpdate* class consists of the following methods. The *updateDefinition* method is responsible for updating the agent knowledge with the user's topic of interest and preferences. This method is activated during the agent learning mode. The *updateResource* method enables the agent to update the local view of the information resources. The assignment and/or the redundancy avoidance devices might activate this method. The *updateOthersInterest* method is responsible for updating the agent's knowledge with the other agent's topic of interest. The *updateOthersGoals* method is responsible for updating the agent's knowledge with the models of the other agent's goals. This method is activated

when another agent, for example, informs an agent about its schedule. The redundancy avoidance device activates this method. The *updateOthers* method enables the agent to modify all the parameters that are required about other agents. The *updateMonitor* is activated when the agent requests feedback from the user.

5.6 Local Scheduler

Scheduling the agent's activities is implemented based on its mental states. Recall that during pre-interaction process, each alternative solution is attached to a possible scheduling time. The schedule consists of time frames that are attached to solutions based on the mental state of the goal. All mental states are assigned time frames on the local schedule during the process of a goal. During the process of achieving the goal the states change from 'desire', to 'commitment', then to 'execution'. The desired satisfying time of achieving a goal is used to determine the possible scheduling time for the solution.

There are two types of solutions that are generated by the problem solver. The first type is a solution that consists of pure local domain actions for which the time frame to be assigned by the scheduler is sufficient to carry on the solution with a starting time closest to the desired satisfying time associated with the goal. The second type of solution consists of domain actions that are associated with interdependencies for which the assignment or redundancy avoidance device or both are required. The possible scheduling time for the second type is assigned at the tail of the schedule.

The agent's desires toward achieving goals are assigned a time frame on the local schedule. The type of the time frame depends on the domain actions involved on the desire. Time frames for desires with local domain actions are closed ended,

whereas those associated with interdependencies are open ended. During assignment interaction, for example, if the agent receives an award message for a contract, the mental state is transformed from 'desire' to 'commitment'. In contrast, if the agent receives a dismiss message for the contract, the reserved time frame for the corresponding contract becomes free and available for allocation.

5.7 Communication

The communication component is implemented as a set of object-oriented classes using the layering approach, discussed in section 4.3.5. In order for an agent to communicate with other entities, different communication protocols have been used at the physical layer. An agent in CIGS can communicate using one of the following methods: (1) through TCP/IP SMTP protocol an agent is able to send mail messages to users; (2) through TCP/IP an agent is able to send messages directed to one agent, for which the sender should know the address of the receiver agent; (3) through UDP protocol the agent can broadcast messages, for which the agent can send messages to a group of agents.

The messages that are intended to be sent by an agent as electronic mails are implemented through the utilization of the mail adapter provided by ABE. Using this adapter, the message, the email address of the sender and the email address of the receiver are known by the agent. Also, the set of rules and facts to drive these rules for sending emails are appropriately constructed.

Two modes of communication are implemented using sockets: namely, direct and multicast. Direct communication is implemented as a class *DirectCom* and utilizes the TCP/IP stream sockets. This class implements the Runnable interface that provides the system with the flexibility to run in a separate thread. Also,

it is daemon and reachable through TCP/IP protocol for sending and receiving messages. This class consists of the following methods: the SendMsg, the ReceivedMsg, and the Interpreter. The *SendMsg* is responsible for sending messages to other agents. Other processes of the agent's components that require sending messages activate this process. In this method, the receiver agent address is identified and the message sent to its destination. The *ReceivedMsg* is daemon and listens at a specified socket number that is designated to it and used as its address. This process is responsible for detecting the arrival of any message from other agents. As soon as this process detects a message the interpreter method is activated. The *Interpreter* is activated by the arrival of a message through the SendMsg method. The main function of this method is to identify the performative of the message and the type of its content. When the performative and the type of the message are identified, the corresponding process is activated.

The message is implemented as a class and consists of the following methods. The *Message* method, the main functions of this method are to generate messages in KQML syntax and to generate error messages during parsing for incorrect messages. It accepts the messages from other processes in a predefined string format and reconstructs the message in a KQML format. Every instance of this class has a *performative* variable to specify the kind of speech act message. Not all slots of the message are required for every message to be sent by the agent and accordingly the Message class constructs only the required ones based on the requested process. This approach reduces the parsing time. Other sets of methods are implemented in the Message class, including *setSender* and *setRecievер* specifies the sender and the receiver agents. The *setDeviceName* specifies the type of the interaction device invoked. The *setTimeSend* and *setTimeRecvd* specify the time stamps at which the message is sent and received. The *setContent*, to define the message content,

contains the information to be sent.

Multicast communication is implemented as a class *MulticastCom* and utilizes UDP protocol. This class uses *MuticastSocket*, which is a UDP socket and extends Thread. It allows the agent to send and receive IP multicast packets. This class consists of the following methods. The *joinGroup* method is responsible for enabling the agent to start listening and receiving to all messages through broadcasting. The *leaveGroup* method is responsible to end receiving messages through broadcasting.

5.8 Time Synchronization

In the implementation of the agents in the CIGS, some of the time-based issues are considered, such as universal-time clock, execution time and time zones. The *universal time clock* provides a coherency in the time line in the situations where the agents are interdependent and allows each agent to have its own real-time clock. The value of the local time line needs to be measured along a universal-time line (reference point) only during the interaction. The *execution time* is a daemon process that operates on the local schedule utilizing a 'look-ahead' timing strategy. This strategy provides the transformation of time from virtual-time line to real-time clock as follows

- If the top of the schedule is *desire*, then
  - If the *time-stamp of the desire* = the value of *the real-time clock* + *look-ahead time*

    Then the *desire* is destroyed;
- If the top of the schedule is *commitment*, then

– If the *time-stamp of the commitment* = the value of *the real-time clock* + *look-ahead time*

Then the *commitment* is transferred to *intention.*

The *reservation-time* is a daemon process that calculates the reservation-time as the time rate per goal, or $\frac{\Delta t}{in\text{-}load}$, where Δt is a prespecified value for the time interval that is required to count the number of incoming goals (*in-load*).

The *contract-table update* is a 'look-ahead' daemon process which operates on the contract-table. If the reservation-time for a contract has expired then it destroys the contract and invokes the scheduler to free the time-frame reserved for the corresponding *desire.*

The *TimeZone* is a class that represnts a time zone offset, which consists of the following methods. The *getDefault* generates a time zone based on the time zone where the agent is running. The *getTimeZone* generates a time zone based on a time zone ID, where each time zone, if it is supported, has its unique ID.

5.9 Results

In the following sections, four different scenarios were constructed to demonstrate how the agents interact with each other to handle a user query. The rationale behind these scenarios is as follows. Firstly, to show the importance of using a user model that allows the system to provide more relevant information to the users based on their topic of interest. Secondly, to show the system ability in coping with open environments when new information resources join the environment. Lastly, to show how the system performance can be improved in terms of computational time by achieving goals simultaneously. A distributed information systems environment, shown in Figure 5.2, is used throughout all the scenarios. This environment is viewed in terms of the following entities.

- A set of agents denoted by $\mathbf{AG} = \{U, U_1, U_2, UA_1, UA_2, BA, RA_1, RA_2, RA_3\}$, where U denotes a user who is able to access the network and has no User Agent. UA_1 and UA_2 represent two User Agents. BA represents a Broker Agent and RA_1, RA_2 and RA_3 represent three Resource Agents, where RA_1 and RA_2 are able to utilize AltaVista and GoTo search engines respectively.
- A set of objects, $\mathbf{OB} = IR \cup AD \cup AB$, where
  - $IR = \{IR_U, IR_{UA_1}, IR_{UA_2}, IR_{BA}, IR_{RA_1}, IR_{RA_2}, IR_{RA_3}\}$, denote information resources and they can be accessed through their respective agents specified by their names.
  - $AD = \{AD_{UA_1}, AD_{UA_2}, AD_{BA}, AD_{RA_1}, AD_{RA_2}, AD_{RA_3}\}$, denote information resource or a buffer. This buffer can be utilized by agent j, $j \in \mathbf{AG}$ to send and receive information to/from agent k, $k \in \mathbf{AG}$ and $k \neq j$.

- $AB = \{AB_{UA_1}, AB_{UA_2}, AB_{BA}, AB_{RA_1}, AB_{RA_2}, AB_{RA_3}\}$, and AB_m, $m = \{UA_1, UA_2, BA, RA_1, RA_2, RA_3\}$, denote information resource or agent buffer that can only be processed by agent k, $k \in m$. This buffer is used to transfer reachable information by agent k from resource r, where $r \in IR \cup AD$.

Before the agents engage in the interaction with each other, it is assumed that each agent is configured, such as their addresses, during the startup process. Upon the initialization, each agent might advertise its capability to the Broker Agent to join groups of its interest. For example, RA_1 advertise its capability in terms of topic, say *places*, and gets the replay back with the address of the interested group as shown in Figure 5.3.

For agent i, let models of the other agent j be $M_j^i =< M_{G_j}^i, M_{lsc_j}^i, M_{Cap_j}^i >$, where $i, j \in \mathbf{AG}$. For example, $M_{G_{UA_2}}^{UA_1} = \{weather\}$ is the model of the UA_2's goals set by the UA_1 with its local scheduler $M_{lsc_{UA_2}}^{UA_1} = \phi$ at $t = 0$, and $M_{Cap_{UA_2}}^{UA_1} = \{accommodations\}$ is the model of the agent UA_2's capabilitities set by UA_1 regarding the set of information that agent UA_2 can manage. Also, let a query $q = (G, \pi)$ where G is the goal to be achieved by the agent and π is the set of associated constraints. The goal, G, might be decomposable into a set of sub-goals, $G = \{G_1, \ldots, G_n\}$, such that the individual sub-goal might be delegated to other agents. At the initial state, or at time equal to zero, it is assumed that none of the agents has developed desires, commitments, or intentions toward any goal yet (i.e., $D_{G_j}^i = C_{G_j}^i = I_{G_j}^i = \phi$). Furthermore, the local history of the world for each agent is shown in Table 5.3.

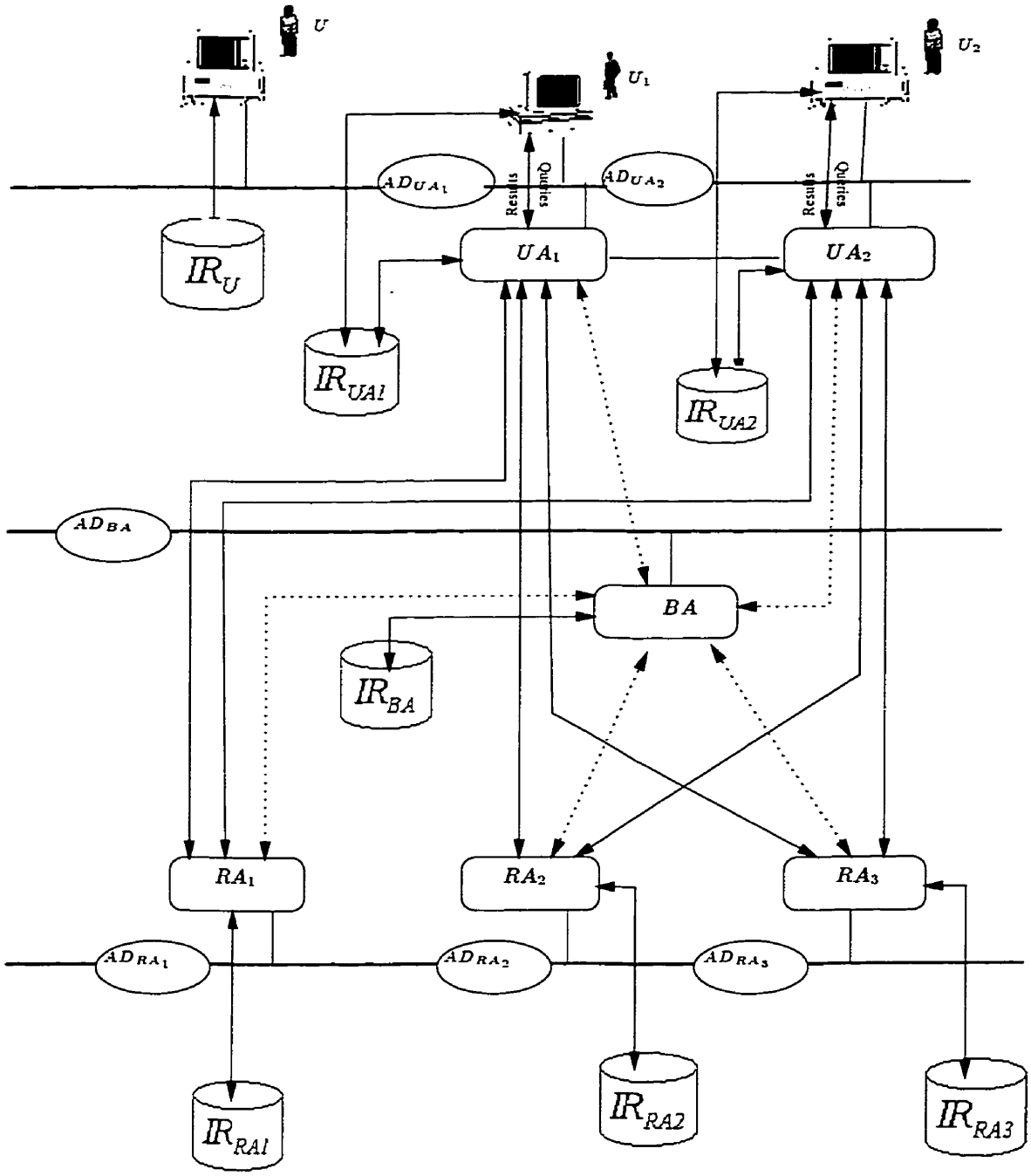

Figure 5.2: Cooperative Information Gathering Environment.

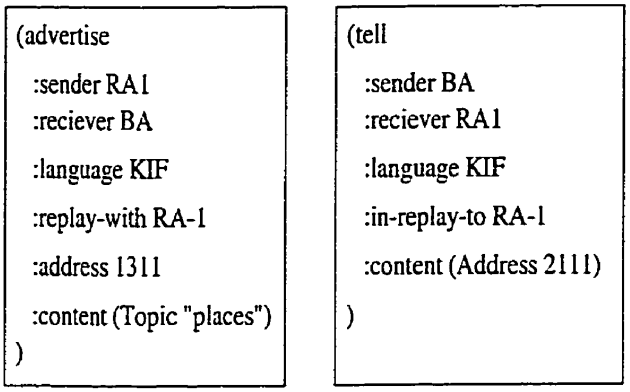

Figure 5.3: Messages sent by RA_1 and BA.

Agent name	Local history
UA_1	$W_t^{UA_1} = \{Topic(IR_{1_{UA_1}}) = sports,\ Quality(IR_{1_{UA_1}}, sports) = 70\}$, $\{Topic(IR_{2_{UA_1}}) = weather,\ Quality(IR_{2_{UA_1}}, weather) = 65\}$
UA_2	$W_t^{UA_2} = \{Topic(IR_{1_{UA_2}}) = accommodations,$ $Quality(IR_{1_{UA_2}}, accommodations) = 60\}$
BA	$W_t^{BA} = \{Capable(RA_1, places)\}$, $\{Capable(RA_2, sports)\}$, $\{Capable(RA_3, places)\}$
RA_1	$W_t^{RA_1} = \{Topic(IR_{1_{RA_1}}) = agents,\ Quality(IR_{1_{RA_1}}, agents) = 70\}$, $\{Topic(IR_{2_{RA_1}}) = places,\ Quality(IR_{2_{RA_1}}, places) = 80\}$
RA_2	$W_t^{RA_2} = \{Topic(IR_{1_{RA_2}}) = computers,$ $Quality(IR_{1_{RA_2}}, computers) = 70\}$,
RA_3	$W_t^{RA_3} = \{Topic(IR_{1_{RA_3}}) = sports, Quality(IR_{1_{RA_3}}, sports) = 70\}$, $\{Topic(IR_{2_{RA_3}}) = places, Quality(IR_{2_{RA_3}}, places) = 90\}$

Table 5.3: The local history of the world for each agent.

5.9.1 Scenario 1: With No User Model

In this scenario, a user submits a query to the User Agent UA_1 in the form 'weather at waterloo ontario', and selects the desired response time and the search engine to be used, as shown in Figure 5.4. In this implementation, the text of the query does not have a specific syntax; however, during query processing the first word of the text is considered as the user's topic of interest, whereas the remained words are considered as constraints. The UA_1 then assigns this query, translated in KIF, as a goal to be achieved by the capable Resource Agent. In this case, RA_1 is selected which utilizes AltaVista search engine. RA_1 is constrained to provide the required information within the specified time. Now, RA_1 translates the query expressed in KIF into the language appropriate for the underlying system, i.e., AltaVista search engine. Then, it submits this query to AltaVista search engine to access and retrieve the top ranked addresses and their associated summaries by the search engine. The number of addresses and their summaries are returned by RA_1 depending on the time constraint. Finally, those addresses and summaries are returned to UA_1 and displayed to the user. The Agent Status section shows the agent activities while achieving a goal(s). Table 5.5 shows a complete list of the results returned by AltaVista search engine that includes the top ten ranked addresses. The returned addresses are only pointers to both relevant and non-relevant information to the user's needs. For example, the address weather.ec.gc.ca/forecast/ykf.html points to a document that contains information about 'weather' that might be of the user's interest. Although, the returned addresses might point to relevant and non-relevant information, they have been included as relevant based on the techniques used by the search engine. This scenario demonstrates that the User Agent returns results, which includes only the addresses and the summaries of the information ranked by the search engine within the specified time, however to provide relevant information

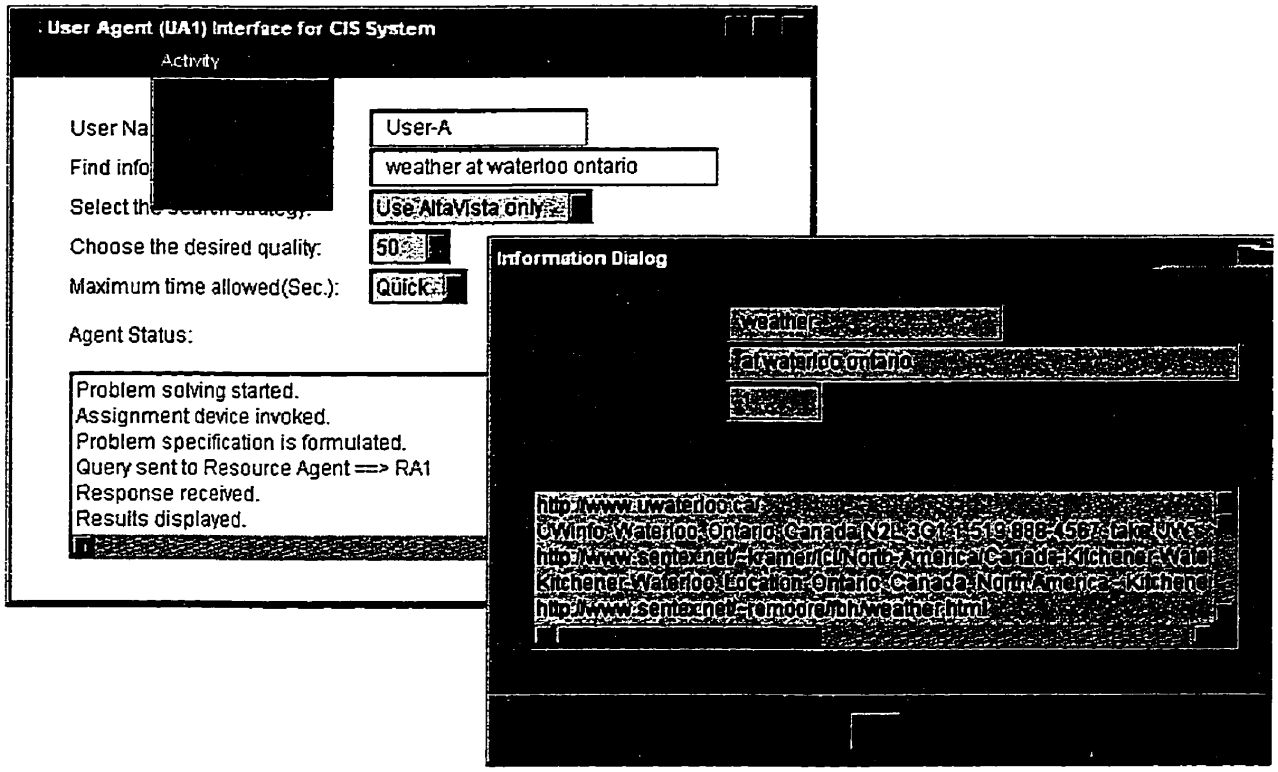

Figure 5.4: An example of the main and results interaction windows with the user.

should be based on the user's interest.

5.9.2 Scenario 2: Specializing to User's Interest

This scenario is used to demonstrate the ability of the system to adapt the user's topic of interest and preferences and explore more relevant information. In this scenario, a collection consists of one hundred documents are saved in the local database. These documents are downloaded from the Web utilizing AltaVista and Lycos search engines. The user submitted a query in the form 'weather at waterloo ontario' to UA_1. Initially the user described his/her view of weather as 'temperature, wind and humidity'. The user provides consistent feedback so that the system

1.	Uwinfo -- University of Waterloo Uwinfo. Waterloo, Ontario, Canada N2L3 G1. +1 519 888-4567. General. * Daily Bulletin * UW overview * Virtual campus tour * Directions and map *... URL: www.uwaterloo.ca/
2.	Mirror Website -- Weather Information This Mirror Website page provides access to weather information services. The MIRROR Website is located in Middlesex County, Ontario, Canada. We... URL: www.mirror.org/weather.html
3.	Environment Canada Weather Forecast: Waterloo, Ontario Canadians can vote for the top weather stories of the 20th century. Waterloo, Ontario. Current Conditions. Shallow Fog Patches. Temperature: -15.1... URL: weather.ec.gc.ca/forecast/xhf.html
4.	Kitchener-Waterloo, Ontario Kitchener-Waterloo, Ontario WWW Sites. General. Agriculture. Arts & Entertainment. Classifieds. Clubs & Associations. Education. Health. Media. News... URL: zeus.peacenet.com.pe/atgfor/kw.htm
5.	Kitchener.Com - Your Community Portal Kitchener Waterloo resource site featuring free classifieds and local information. URL: www.kitchener.com/
6.	Kevin's Web Page Make Sure you visit Santa's Christmas Page - Great for all Seasons! SANTA'S CHRISTMAS PAGE. MY FAVOURITE SITES. SEND KEVIN DORSCHT E-MAIL. ---... URL: www.sentex.net/~kdor/
7.	Mount Shasta City Northern California and Mt. Shasta Mount Shasta City, Mt. Shasta and the community. Shopping Online... URL: www.mountshasta.com/
8.	University of Waterloo Weather Station - Station Information Station Information. In mid 1997, the equipment for the Climatological Station was provided to the University of Waterloo by Meteorological Service... URL: weather.uwaterloo.ca/info.htm
9.	TUG Libraries: Electronic Reference Shelf – Weather Weather. Table of Contents. Current Conditions & Forecasts. Local. Canada. US. A. & the Rest of the World. Historical Meteorological Data. Current... URL: www.tug-libraries.on.ca/reference/tools/weather.html
10.	Karl's Chain of Links v1.00 Home | Email | Site Map | Sailor Moon | Links] Love them. Links. Places I go daily. CTRL-A. Slashdot. Dilbert. Kevin & Kell. Userfriendly. Shuggy... URL: www.cschib.uwaterloo.ca/u/kmazurys/links.html

Figure 5.5: Results returned utilizing AltaVista.

response should converge to the set of concepts that define the user's topic of interest. The system is iteratively used for a number of user sessions. Each session involves reading the documents retrieved by the model and providing positive feedback for documents about 'weather'. After a feedback is provided to the relevant documents, the user model is modified and the search is performed again using the modified model. After the user provides a feedback, the user model is modified using Equation 4.2 with a learning rate of 0.5.When the search is performed, the documents are sorted by their qualities (i.e. similarity scores) using Equation 4.5. Each quality of these documents is multiplied by 100 to produce values from 0 to 100. The concepts and qualities representing the user model at different sessions are shown in Table 5.6. This table shows the initial view on the left-hand side and the final view, after ten sessions, which consists of twenty concepts with their qualities on the right-hand side. When the user of UA_1 submits a query requesting information about 'weather' with a quality of 60%, based on the previous interaction between UA_1 and the user, UA_1 has built a model of the user toward his/her interest about 'weather'. Figure 5.7 shows the quality and addresses of two documents that are retrieved as relevant (i.e. documents with quality equal or higher than 60%) by UA_1, utilizing the user model. To evaluate the performance of the system, 'recall' and 'precision' measures are used. These measures are well known and commonly used to evaluate the performance of information retrieval systems. The results of this experiment are shown in Figure 5.8. Recall measures the proportion of relevant information actually retrieved in response to a search (that is, the number of relevant documents actually obtained divided by the total number of relevant documents in the collection). Precision measures the proportion of retrieved documents actually relevant (that is, the number of relevant documents actually obtained divided by the total number of retrieved documents). The precision and

user model	User model session 1			User model session 6			User model session 10	
	Concept	Quality		Concept	Quality		Concept	Qualit
TEMPERATURE	TEMPERATURE	0.286		TEMPERATURE	0.399		TEMPERATURE	0.488
WIND	WIND	0.269		WIND	0.329		WIND	0.463
HUMIDITY	HUMIDITY	0.241		HUMIDITY	0.327		HUMIDITY	0.431
	FORCAST	0.132		RAIN	0.206		RAIN	0.255
	CONDITION	0.118		VISIBILITY	0.186		VISIBILITY	0.243
	RAIN	0.106		PRESSURE	0.113		PRESSURE	0.183
	CLOUD	0.094		WATERLOO	0.104		WATERLOO	0.143
	LOCAL	0.091		ONTARIO	0.101		ONTARIO	0.143
	PRESSURE	0.089		PRECIPITATION	0.100		PRECIPITATION	0.143
	VISIBILITY	0.076		FORCAST	0.097		FORCAST	0.143
	RELATIVE	0.075		SPEED	0.091		SPEED	0.132
	PERIOD	0.073		CLOUD	0.088		CLOUD	0.124
	OCTOBER	0.068		LOW	0.078		LOW	0.114
	SPEED	0.066		HIGH	0.078		HIGH	0.114
	INFORMATION	0.064		UNIVERSITY	0.073		UNIVERSITY	0.112
	PRECIPITATION	0.064		SUN	0.071		SUN	0.104
	CALM	0.062		RELATIVE	0.069		RELATIVE	0.094
	STATION	0.056		ENVIRONMENT	0.067		ENVIRONMENT	0.092
	LOW	0.054		CONDITION	0.062		CONDITION	0.091
	HIGH	0.052		LOCAL	0.061		LOCAL	0.084

Figure 5.6: Building user model.

Document quality	Address
73.65	http://weather.ec.gc.ca/forecast/ykf.html
68.41	http://weather.uwaterloo.ca/info.htm#formulas

Figure 5.7: Results returned by UA_1.

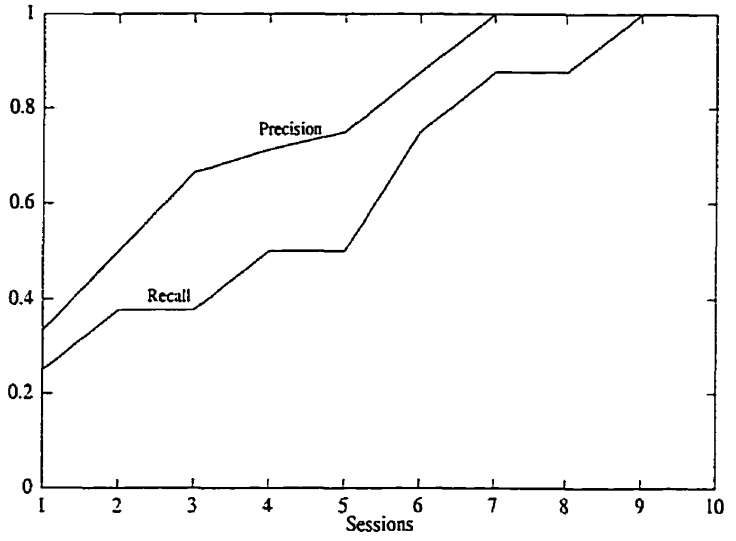

Figure 5.8: Recall-precision graph.

recall are evaluated over each time the user provides a feedback. The scope of this experiment is restricted to a predefined collection of documents. This means that, the number of documents are static to the local database over the duration of this experiment. Recall is calculated by manually going through each document in the collection to identify the documents that are relevant. A threshold is set for document scores. To achieve the desired goal of precision and recall the similarity scores of relevant documents should lie above the threshold and the scores of irrelevant documents lie below the threshold.

Figure 5.8 shows an improvement of the behavior of the system precision and recall over time. This is primarily because the learnt concepts that define the topics during interactions affect the system performance. The initial view that is provided by the user to define the topic helps the search for relevant information. After the user provides a feedback, the user's model is modified and the search is performed again with the updated model. The user's topic of interest is quite

successful at converging on the concepts occurring in the documents with higher qualities to the desired concepts relating to 'weather'. It can be seen that with an initial view and with consistent feedback, the system has succeeded in specializing to the topics of the user interest. The feedback provided by the user also helps the system to differentiate between relevant and irrelevant documents. This means lower scores are assigned to irrelevant documents, thereby improving precision. Thus, the results of this experiment suggest that the user's model and relevance feedback help improve recall and precision of the system.

5.9.3 Scenario 3: Assigning Goals

In the following scenario, a user decided to go on vacation and submitted a query to the User Agent, UA_1, to provide information about some recommended vacations. The query in the form of 'vacation at waterloo ontario' with a quality of 60% and a response time within 30 seconds. It is assumed that there are two aspects representing the vacation namely, weather situation and places to visit. Thus, UA_1 requires information about vacation (a goal G denoted by $Topic(AB_{UA_1}) = vacation$). The UA_1 problem solver starts to reason about G by using the local information resources. From $W_t^{UA_1}$ there is no resource that carries information about vacation. Then, UA_1 decomposes the goal by applying the *Decompose* action to G that generates a set of sub-goals namely G_1 and G_2. Where, $G_1 = Topic(AB_{UA_1}) = weather$ and $G_2 = Topic(AB_{UA_1}) = places$. Clearly, satisfying G_2 is beyond UA_1 capability. It is assumed that each topic produced as a result of the decomposition is assigned the same quality given by the user. The User Agent displays each sub-topic in a separate window to get a feedback for each of them. It should be noted that, UA_1 provides the user with an interaction window to define the concept explicitly, as shown in Figure 5.9. The UA_1 also monitors the number of negative feedback

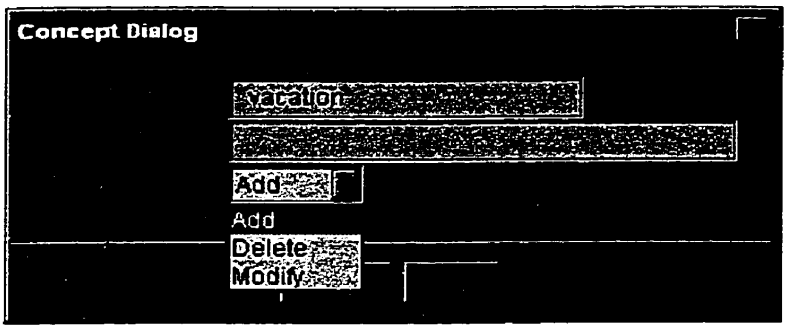

Figure 5.9: The User Agent queries the user for concept definition.

received from the user for the same retrieved topic. When the number of negative feedback exceeds a predefined threshold, it sends an email to interest users for topic definition.

To deal with this capability interdependency, the assignment device is invoked. Then, the problem specification is formulated as a query, q_{01}, that includes the topic and the constraints that include the location, the desired-satisfying-time t_d and the expiration-time t_e. This is represented as $q_{01} = (places, at\ waterloo\ ontario, t_d, t_e)$. Where, t_d is determined based on the specified time-stamp of G and t_e is calculated based on t_d and the current-time. Since UA_1 has no models of other agents, in its knowledge, yet as related to this goal, it sends a request to the BA for recommendation. The role of BA is to provide UA_1 with the existing resource agents that are able to provide information about 'places'. With the role of BA and the capability of UA_1 to assign goals to other agents, UA_1 is able to provide the user with the desired information from the available resources during query processing.

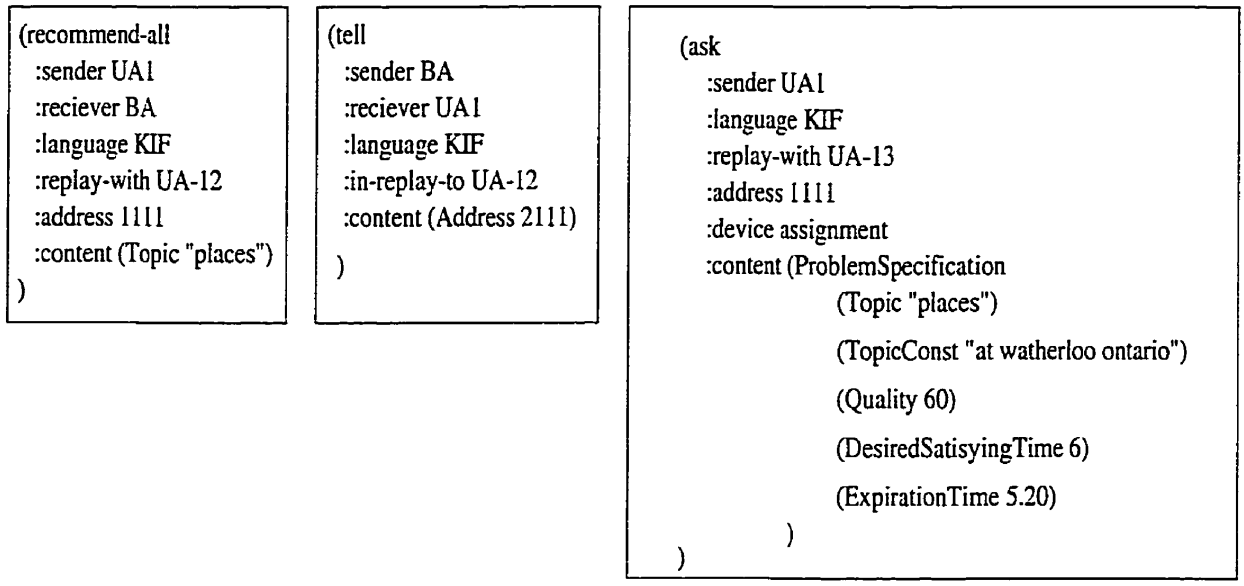

Figure 5.10: Examples of messages sent and received by UA_1 during assignment.

Thus, CIGS copes well with open environments, for which BA recommends both the old and the new (i.e. just joined the environment) information resources that are represented or modeled by Resource Agents and able to provide the desired information. In this scenario, RA_1 existed and advertised its interest in 'places' to BA. However, RA_3 just advertised its interest in 'places' to BA. Then, RA_1 and RA_3 can be considered as potential contractors. As soon as UA_1 receives the response from BA it sends out q_{01} to the address recommended, as shown in Figure 5.10. At the same time an alarm is set and activated either by the expiration-time or the arrival of the responses.

When RA_1 and RA_3 receive q_{01}, each enables its respective schedulers to assign a time frame $t_{lsc_{RA_1}}$ and $t_{lsc_{RA_3}}$, respectively, for q_{01} goal. $W_t^{RA_1}$ and $W_t^{RA_3}$ show that the information is available at their local resources with no further coordination required with other agents. Then, the pre-interaction determines the estimated cost for achieving the goal of q_{01}, say C_1 and C_2 by each agent respectively with possible-

starting-times t_{pst_1} and t_{pst_2}. In this implementation, every one-percent of quality is set to one unit of cost. Then, RA_1 formulates a bid as $B_{RA_1} = (T, Q, t_{pst_1}, t_{lsc_1}, C_1)$. Whereas, RA_3 formulates a bid as $B_{RA_2} = (T, Q, t_{pst_2}, t_{lsc_2}, C_2)$ and they send these bids back to UA_1 as responses to q_{01}, provided that the desired-satisfying-time and the expiration-time of the goal can be met. Also, the mental state of this goal is set to desire by RA_1 and RA_3.

As soon as UA_1 receives the responses (bids) it selects the best bid, using the evaluation parameters and the selection heuristics discussed in section 5.5.1. It is assumed that both bids met the desired time. Thus, for this scenario, RA_1 is selected because it provides higher quality than the requested with less cost. Accordingly, a dismiss message is sent to RA_3 and an award message to RA_1. When RA_3 receives a dismiss message it frees the reserved time from its local scheduler because the contracted query is no longer valid. Whereas, when RA_1 receives the award message it transforms the status of the mental state of the goal from desire to commitment. Hence, RA_1 will achieve the goal and sends the results to UA_1 and then invoke the knowledge update. Finally, UA_1 receives the required information and then invokes the knowledge update device to update, for example, models of the other agents in terms of information quality, response time and cost; the information is then displayed to the user and becomes available locally.

5.9.4 Scenario 4: Avoiding Redundant Goals

In this scenario, it is considered that UA_2 is also trying to get some information related to 'weather', or $Topic(AB_{UA_2}) = weather$. This goal is equivalent to G_1 of UA_1. Since the goal G of UA_1 is decomposable into G_1 and G_2, and UA_1 is able to avoid redundant effort by invoking the redundancy avoidance device, the sys-

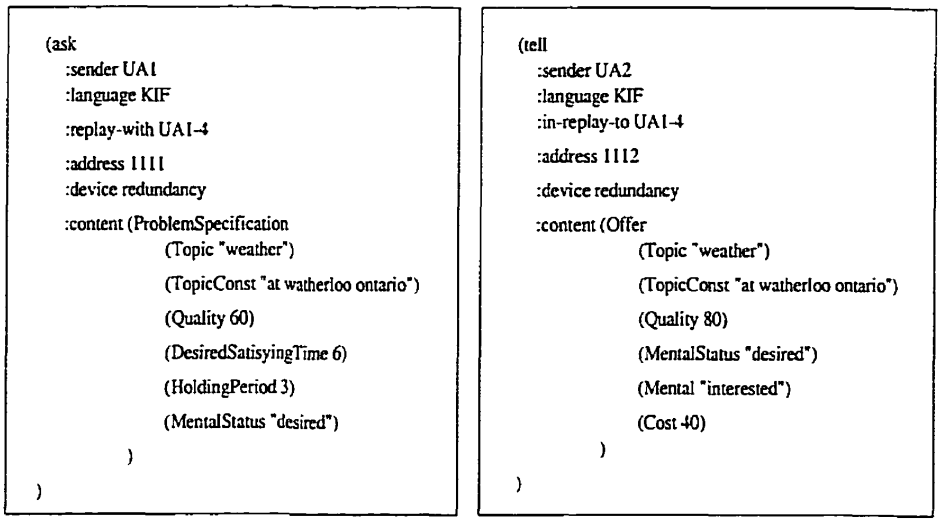

Figure 5.11: Examples of messages sent by UA_1 and UA_2 during redundancy avoidance.

tem performance can be enhanced by achieving G_1 and G_2 simultaneously. When UA_1 starts to achieve this goal, reasoning about $M_{UA_2}^{UA_1}$ indicates that G_1 might need to be achieved by UA_2 at the same time. To deal with this common interdependency, the redundancy avoidance is invoked. Then, the problem specification is formulated as a query, q_{02}, that includes the topic and the constraints that include the location, the desired-satisfying-time t_d, the holding period t_h and the mental state of G_1 as 'desired'. This is represented as $q_{02} = (weather, at\ waterloo\ ontario, t_d, t_h, desired)$. Using models of the others, q_{02} is sent out to UA_2 as a potential partner, as shown in Figure 5.11-(a). Also an alarm is set to be activated either when the response from UA_2 is received or the expiration-time is reached.

When UA_2 receives q_{02} from UA_1 and evaluates its part of the evaluation parameters including topic, quality, interested or not interested to achieve the goal and the mental state of the goal as *desired* or *committed*, then, UA_2 formulates

an answer as $R_{UA_2} = (T, Q, interested, desired, C)$ and sends it back to UA_1 as a response to q_{02}, as shown in Figure 5.11-(b).

As soon as UA_1 receives the response from UA_2, it determines whether UA_2 is an appropriate partner or not by using the evaluation parameters and heuristics discussed in section 5.5.2. It is assumed that UA_2 requires information with these characteristics, $Topic = weather$ and $Quality = 80$. It is also assumed that UA_2 is an appropriate partner, then UA_1 and UA_2 engage in the negotiation process for who will achieve G_1. Both agents are cooperative agents and they have the right to opt out. In this implementation, it is assumed that the cost proposed when formulating the problem specification is equal to 50% of its local cost. Also, a predefined threshold is used for the discounting strategy to calculate the cost for the next counter-offer. Once the negotiation process starts, each agent tries to benefit out of this process, by accepting the offer that has cost less than the cost of pursuing it locally utilizing the time set for negotiation. In this scenario, UA_1 accepts the offer because it provides a utility of 20 to itself and a utility of 40 to UA_2. Moreover, UA_1 receives 15% higher quality. Both agents, after UA_1 receives the information from UA_2, invoke their respective knowledge update to update models of each other for future interactions.

5.10 Summary

This chapter described in detail the implementation of the agents' knowledge and capabilities used in CIGS. It explained how each of the following is implemented: (1) Information resources representations, (2) The problem solvers, (3) The pre-interaction, (4) Interaction that includes users and agents, (5) Communication between the different elements of the environment, (6) The local scheduler, and (7)

Execution. It demonstrated how the agents interact with each other to handle a user query through a set of scenarios. These scenarios have shown how the agents transparently gather information from distributed information resources.

Chapter 6

Conclusions and Future Research

The main goal of this dissertation has been to develop a cooperative information gathering system in open domain. This goal has been achieved with a design of multi-tier agent-based architecture in which agents cooperatively carry out distributed, coordinated, intelligent information gathering. The system architecture has been implemented and demonstrated using a variety of scenarios. This chapter reviews the main contributions of this work and presents some directions for future research.

6.1 Summary of Contributions

This thesis has tackled many issues to accomplish its desired goal. One appropriate way to examine the contributions of this research is to frame them within the context of each issue.

- Reasoning and Manipulating Information.

  This has been accomplished through providing a distinguished view and def-

initions for the fundamental aspects of the domain: namely, data and information. Information has also been viewed as the characteristic of information resource that includes topic and quality. This enables the agents to locate and retrieve the desired information effectively with the appropriate and necessary operations.

- Design Issues.

  Unlike some previous approaches, the agents of the proposed system provide a complete solution for the main design issues: namely, autonomy, heterogeneity and transparency. This has been achieved using the CIR-Agent model to design the agents. Modeling each agent as an independent pro-active entity incorporates autonomy. Heterogeneity is dealt with by modeling an agent as goal-driven entity. Modeling each agent as cooperative, coordinated agent to provide transparency.

- Agents' Behavior.

  In multi-agent environment, agents may exhibit different types of behaviors. This thesis has focused on cooperative behavior. The objective in analyzing cooperation has been to enable agents to exhibit different types of behaviors. To achieve this, an agent is equipped with different interaction strategies, based on the help and benefaction aspects, including selfish, benevolent and cooperative.

- Dynamic and Open Environments.

  Because different users usually have different views and interest in the same information, users are constrained by time limits, information resources contents are changing constantly, and information resources might appear and disappear at any time, the agents are classified into different types to deal

with these issues. This thesis detailed the functionality of a number of agents including User Agents, Broker Agents and Resource agents.

- System Performance.

  The system performance is enhanced in terms of computational time by achieving decomposable goals simultaneously. The system also focuses on retrieving the most relevant information by utilizing user model.

- System Architecture.

  Of the existing cooperative information gathering, the proposed system architecture is based on viewing information gathering as problem independent of its structure, for which a single or a group of agents can participate in an open environment. The agent determines the degree of its participation in terms of cooperation during the runtime, rather than problem-structured centered. This view is supported, in CIR-Agent model, through the agent's architecture that enables the agent to be goal-driven, autonomous, rational as well as able to determine the interaction setting as cooperative or otherwise.

Finally, the feasibility of the proposed architecture has been demonstrated by implementing a prototype on distributed information systems. It has been shown how the agents can transparently cooperate to locate and retrieve information from distributed, dynamic and heterogeneous information resources to different users having different views and interest.

6.2 Future Research

The work presented in this dissertation addresses a number of basic issues. This section discusses some of the ideas that can be pursued to improve the proposed

solutions.

- A natural extension of the work presented in this thesis is to widen the accessibility to information resources of different types of structures and formats. This may require utilizing the appropriate converters and developing different heuristics to identify and extract information. This raises the issue that is worth investigation: how to merge the information.

- Another issue to consider is the development of a concise, uniform and declarative description of semantic information, independent of the underlying syntactic representation or the conceptual models of information resources. One way that may be considered is to allow the agents to specify the ontology for certain domains.

- The problem solving technique must also be considered. Using search techniques, in which a set of heuristics and evaluation functions, could be developed, to make the decomposition more dynamic.

- Improving the topic and quality of information is also an objective that is worth pursuing. The ability to process information resources for extracting information is essential both to refine the search activities and to provide relevant information. This may require developing different techniques and algorithms to extract information. It is also worth investigation to consider and introduce a set of learning techniques that might improve the information.

Bibliography

[1] Arens, Y., Chee, C. Hsu, C. and Craig A. Knoblock, " Retrieving and Integrating Data from Multiple Information Sources", *International Journal of Intelligent and Cooperative Information Systems*, Vol. 2, No. 2., pp. 127-158, 1993.

[2] Argyle, M. "Cooperation: The Basis of Sociability", Routledge, England, 1991.

[3] Aronfreed, J., "The Socialization of Altruistic and Sympathetic behaviour: Some Theoretical and Experimental Analysis", *Altruism and Helping Behavior*, Macaulay, J. and Berkowitz, L. (Eds.), New York: Academic Press, 1970.

[4] Aronfreed, J., "Moral Development from the Standpoint of a General Psychological Theory", *Moral Development and Behavior*, Lickona, T. (Ed.), New York: Holt, Rinehart and Winston, 1976.

[5] Arthur M. Keller and Michael R. Genesereth, "Using Infomaster to Create a Housewares Virtual Catalog," in *Int.Journal of Electronic Markets*, Institute for Media and Communication Management, University of St. Gallen, Switzerland, Vol. 7, No. 4, 1997.

[6] Atkin, E., Birmingham, W., Durfee, E., Glover, E., Mullen, T., Rundensteiner, E., Soloway, E., Vidal, J., Wallace, R., and Wellman M. "Toward Inquiry-Base Education Through Interacting Software Agents", Computer, pp. 69, May 1996.

[7] Barbuceanu, M. and Fox, M. S.,"Cool: A Language for Describing Coordination in Multi-Agent Systems", In *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95)*, San Francisco, CA, pp. 17-24, 1995.

[8] Bar-Tal, D., Raviv, A. and Leiser, T., "The Development of Altruistic Behavior: further evidence", *Development Psychology*, vol. 16, pp. 516-524, 1980.

[9] Bayardo, R., Bohrer, W., Brice, R., Cichocki, A., Fowler, G., Helal, A., Kashyap, V., Ksiezyk, T., Martin, G., Nodine, M., Rashid, M., Rusinkiewicz, M., Shea, R., Unnikrishnan, C., Unruh, A. and Woelk, D., "Agent-Based Semantic Integration of Information in Open and Dynamic Environments", In *In the Proceedings of SIGMOD-Record*, vol. 26, no. 2, pp. 195-206, 1997.

[10] Berndtsson, M., Active Capability Support for Cooperative Strategies in Cooperative Information Systems, *Ph.D. Thesis*, University of Exeter, Department of Computer Science, 1998.

[11] Bertram, R., "Kin Selection in Lions and in Evolution', *Growing Points in Ethnology*, Bateson, P. and Hinde, R. (Eds.), Cambridge University Press (UK), pp. 281-301, 1976.

[12] Berkowitz, L., "Social Norms, Feelings, and Other Factors Affecting Helping and Altruism", *Advances in Experimental Social Psychology*, Berkowitz, L. (Ed.), Academic Press, Vol. 6, 1972.

[13] Joseph P. Bigus and Jennifer Bigus, *IBM Agent Building Environment Developer's Toolkit*, Components and Adapter References, Level 6, June 1997.

[14] Joseph P. Bigus and Jennifer Bigus, *IBM Agent Building Environment Developer's Toolkit*, User's Guide, Level 6, June 1997.

[15] Birmingham, W. "An Agent-Based Architecture for Digital Libraries", D-Lib Magazine, July, 1995.

[16] Birmingham, W., Durfee E., Mullen, T., and Wellman M., "The Distributed Agent Architecture of the University of Michigan Digital Library", In AAAI Spring Symposium on Information Gathering in Heterogeneous, Distributed Environments, Stanford, CA, AAAI Press, 1995.

[17] Collet, C., M. Huhns, and W. Shen, "Resource Integration Using a Large Knowledge Base in Carnot", IEEE Computer, Vol. 24, No. 12, pp. 55-62, 1991.

[18] Cost, R. S., Finin, T., Labrou, Y., Luan, X., Peng, Y., I., Mayfield, J. and Boughannam, A., "Jackal: a java-based tool for agent development", In *AAAI Workshop on software Tools for Developing Agents*, 1998.

[19] Curiel, I., Cooperative Game Theory And Applications, Kluwer Academic Publishers, Netherlands, 1997.

[20] Dawes, M., "Social Dilemas", *Annual Review of Psychology*, vol. 31, pp. 169-193, 1980.

[21] Davis, R. and Smith, R., "Negotiation as a Metaphor for Distributed Problem Solving", *Artificial Intelligence*, vol. 20, pp. 63-109, 1983.

[22] Deen, S., "An Architectural Framework for CKBS Applications", IEEE Transaction on Knowledge and Data Engineering, pp. 663-671,1996.

[23] Deen, S., "A Database Perspective to a Cooperation Environment", *In the Proceedings of the First International Workshop on Cooperative Information Agents, CIA '97*, Kiel, Germany, pp. 19-41, 1997.

[24] Decker, K., "Task Environment Centered Simulation", In *Simulating Organizations: Computational Models of Institutions and Groups*, Prietula, M. Carley, K. and Gasser, L. (Eds.), AAAI Press/MIT Press, 1997.

[25] Decker, K., Lesser, V., Prasad, M V. and Wagner, T., "MACRON: An Architecture for Multi-agent Cooperative Information Gathering", *In the Proceedings of the CIKM Workshop on Intelligent Information Agents*, Baltimore, Maryland, December, 1995.

[26] Decker, K., Lesser, V., "Designing a Family of Coordination Algorithms", *In the Proceedings of the First International Conference on Multi-Agent Systems*, San Francisco, AAAI, pp. 73-80, 1995.

[27] Deutsch, M. "A Theory of Cooperation and Competition", *Human Relations*, vol. 2, pp. 129-139, 1949.

[28] Deutsch, M. "An Experimental Study of the Effects of Cooperation and Competition", *Human Relations*, vol. 2, pp. 199-231, 1949.

[29] Donald F. Geddis, Michael R. Genesereth, Arthur M. Keller, and Narinder P. Singh, "Infomaster: A Virtual Information System", In *Intelligent Information Agents Workshop at CIKM '95*, 1995.

[30] Durfee, E., Lesser, V. and Corkill, D., "Cooperation Through Communication in a Distributed Problem Solving Network", *Distributed Artificial Intelligence*, Huhns, M. (Ed.), Pitman, London, pp. 29-58, 1987.

[31] Durfee, E. H., Kiskis, D. L., and Birmingham, W.P., "The Agent Architecture of the University of Michigan Digital Library", *IEEE/British Computer Society Proceedings on Software Engineering*, Special Issue on Intelligent Agents 144(1), February 1997.

[32] Eager, D. J., Lazowska, E. D. and Zahorjan, J.,"A Comparison of Receiver-initiated and Sender-initiated Adaptive Load Sharing", In *Proceedings of the ACM SIGMETRICS Conference on Measurement and Modelling of Computer Systems*, pp. 1-3, Austin, Texas, USA, 1985.

[33] Elmagarmid, A., Rusinkiewicz, M. and Sheth, A., Management of Heterogeneous and Autonomous Database Systems, Morgan Kaufmann Publisher, USA, 1999.

[34] Farrington, D., "Naturalistic Experiments on Helping Behaviour", *Cooperation and Competition in humans and animals*, Colman, A. (Ed.), Van Nostrand Reinhold (UK), pp. 173-217, 1982.

[35] Fowler, J., Perry, B., Nodine, M. and Bargmeyer, B., "Agent-Based Semantic Interoperability in InfoSleuth", *In the Proceedings of SIGMOD-Record*, vol.28, no.1, pp.60-70, March 1999.

[36] Finin, T. and Wiederhold, G., *An Overview of KQML: A Knowledge Query and Manipulation Language*, Department of Computer Science, Stanford university, 1993.

[37] Finin, T., Labrou, Y. and Mayfield, J.,"KQML as an Agent Communication Language", In Bradshaw J.M. (Ed.) *Software Agents*, Cambridge, MA: AAA/MIT Press, pp. 291-316, 1997.

[38] FIPA, http://www.fipa.org, 1998.

[39] Fraser, N. and Hiple, K., Conflict Analysis: Models and resolutions. North Holland, NY, 1984.

[40] Galan, A., JAFMAS: A Java-based Agent Framework for Multi-Agent Systems, *Ph.D. Thesis*, University of Cincinnati, Department of Electrical and Computer Engineering and Computer Science, 1997.

[41] Genesereth, M. R., Fikes, R.E., et al. "Knowledge Interchange Format, Version 3 Reference Manual", Logic-92-1, Stanford University Logic Group, 1992.

[42] Genesereth, M. R., Keller, A. M., Duschka, O.,"Infomaster: An Information Integration System", *Proceedings of 1997 ACM SIGMOD Conference*, May 1997.

[43] Ghenniwa, H., Coordination in Cooperative Distributed Systems, *Ph.D. Thesis,* University of Waterloo, Systems Design Department, 1996.

[44] Ghenniwa, H and Kamel, M., "Coordination in Cooperative Distributed Systems: A Rational, Intelligent Agent Model", submitted to Autonomous Agents and Multi-Agent Systems, 1999.

[45] Ghenniwa, H and Kamel, M., "Interaction Devices for Coordinating Cooperative Distributed Systems", Intelligent Automation and Soft Computing, 2000.

[46] Graham, J. and Decker, K., "Towards a Distributed, Environment-Centered Agent Framework", *In Proceedings of the 1999 Intl. Workshop on Agent Theories, Architectures, and Languages (ATAL-99)*, Orlando, July 1999.

[47] Grzelak, J. and Derlega, V. (Eds.), Cooperation and Helping Behavior: Theories and Research, Academic Press, New York, USA, 1982.

[48] Harget, A. J. and Johnson, I.D., "Load Balancing Algorithms in Loosely-coupled Distributed Systems: A Survey", *Distributed Computer Systems*, Chapter 3, pp. 85-108, 1990.

[49] Hatfield, E., Walster, W. and Piliavin, A., "Equity theory and Helping Relationships", *Altruism, Sympathy, and Helping*, Wispe, L. (Ed.), Academic Press, Inc., pp. 115-139, 1978.

[50] Hornstein, A., Masor, N., Sole, K. and Heilman, M., "Effects of Sentiment and Completion of a Helping Act on Observer Helping: a case for socially mediated zeigarnik effects", *Journal of Personality and Social Psychology*, vol. 17, pp. 107-112, 1971.

[51] Huntingford, F., "The Evolution of Cooperation and Altruism", *Cooperation and Competition in humans and animals*, Colman, A. (Ed.), Van Nostrand Reinhold (UK), pp. 3-25, 1982.

[52] Huhns, M., N. Jacobs, T. Ksiezyk, W. Shen, M. Singh and P. Cannata., "Enterprise Information Modeling and Model Integration in Carnot", in Charles

J. Petrie Jr., ed.,. Enterprise Integration Modeling: Proceedings of the First International Conference, MIT Press, Cambridge, MA, 1992.

[53] Huhns, M., N. Jacobs, T. Ksiezyk, W. Shen, M. Singh and P. Cannata., "Integrating Enterprise Information Models in Carnot", *Proceedings of International Conference on Intelligent and Cooperative Information Systems*, M. Huhns, M. Papazogloou, and G. Shlageter (Eds.), Society Press, Los Alamitocs, CA, pp. 32-42, 1993.

[54] Jacobs, N. and Shea, R., "The Role of Java in InfoSleuth: Agent-based Exploitation of Heterogeneous Information Resources", MCC Technical Report MCC-INSL-018-96, March, 1996. Presented at the IntraNet96 Java Developers Conference.

[55] Jennings, N. R. and Campos J. R., "Towards a Social Level Characterisation of Socially Responsible Agents", In *IEE Proceedings on Software Engineering*, 144 (1), pp. 11-25, 1997.

[56] Kalenka, S. and Jennings, N. R., "Socially Responsible Decision Making by Autonomous Agents" in Cognition, Agency and Rationality (eds K. Korta, E. Sosa, X. Arrazola) Kluwer pp. 135-149, 1999.

[57] Knoblock, C., "Planning, Executing, Sensing, and Replanning for Information Gathering", In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, Montreal, Canada, 1995.

[58] Knoblock, C., "Generating Parallel Execution Plans with a Partial-Order Planner", In *Proceedings of the Second International Conference on Artificial Intelligence Planning Systems*, Chicago, IL, 1994.

[59] Knoblock, C. and Ambite, J., "Agents for Information Gathering", In Bradshaw J.M. (Ed.) *Software Agents*, Cambridge, MA: AAA/MIT Press, 1997.

[60] Knoblock, C., Arens Y., and Hsu, C., "Cooperating Agents for Information Retrieval", In *Proceedings of the Second International Conference on Cooperative Information Systems*, University of Toronto Press, Toronto, Ontario, Canada, 1994.

[61] Lander, S. E. and Lesser, V. R., "Understanding the Role of Negotiation in Distributed Search Among Heterogeneous Agents", in *Proceedings of the International Joint Conference on Artificial Intelligent*, Chambery, France, pp. 438-444, 1993.

[62] Latane, B. and Darley, M., "The Unresponsive Bystander: Why Doesn't He Help?", New York: Appleton-Century-Crofts, 1970.

[63] Lawrence, S. and Giles, L., "Searching the World Wide Web", Science, Volume 280, Number 5360, pp. 98, April 3, 1998.

[64] Lenat, B., Guha, R., *Building Large Knowledge-Based Systems: Representation and Inference in the Cyc Project*, Addison-Wesley Publishing Company, Inc., Reading, MA, 1990.

[65] Lerner, J., "The Desire for Justice and Reactions to Victims", *Altruism and Helping Behavior*, Macaulay, J. and Berkowitz, L. (Eds.), New York: Academic Press, 1970.

[66] Lesser, V., Horling, B., Klassner, F., Raja, A., Wagner, T. and Zhang, S., "BIG: An Agent for Resource-Bounded Information Gathering and Decision Making", *Artificial Intelligence Journal, Special Issue on Internet Information Agents*, vol. 118, no. 1-2, pp. 197-244, 2000.

[67] Litewin, W., Mark, L., and Roussopoulos, N., "Interoperability of Multiple Autonomous Databases", *ACM Computing Surveys*, Vol. 22, No. 3, pp. 267-296, 1990.

[68] Lux, A., Greef, P., Bomarius, F. and Steiner, D., "A Generic Framework for Human Computer Cooperation", *Proceedings of International Conference on Intelligent and Cooperative Information Systems*, M. Huhns, M. Papazogloou, and G. Shlageter (Eds.), Society Press, Los Alamitocs, CA, pp. 89-97, 1993.

[69] Mahendra Sekaran and Sandip Sen, "To help or not to help," in the Proc. of the Seventeenth Annual Conference of the Cognitive Science Society (pages 736-741), Pittsburgh, Pennsylvania, July 1995.

[70] Marwell, G. and Schmidt, R., "Cooperation: An Experimental Analysis", New York, Academic Press, 1975.

[71] McClintock, C. and Van Avermaet, E., "Social Values and Rules of Fairness: A Theoretical Perspective", *Cooperation and Helping Behavior: Theories and Research*, Derlega, V. and Grzelak, J. (Eds.), Academic Press, New York, USA, pp. 43-71, 1982.

[72] Mohamed, A. and Huhns, M., "Benevolent Agents", *The Third International Conference on Autonomous Agents (Agents99)*, 1999.

[73] Mullen, T. and Wellman M., "A Simple Computational Market for Network Information Services", *Proc. First Int'l Conf. Multiagent Systems*, Amer. Assn., Artificial Intelligence Press, Menlo Park, Calif., pp. 283-289, 1995.

[74] Newell, A., "Reflections on the Knowledge Level", Artificial Intelligence, No. 53, pp. 31-38, 1992.

[75] Newell, A., "The Knowledge Level", Artificial Intelligence, No. 18, pp. 87-127, 1982.

[76] Nodine, M., Fowler, J. and Perry, B., "Active Information Gathering in Infosleuth", *In the Proceedings of the International Symposium on Cooperative Database Systems for Advanced Applications*, 1999.

[77] Nodine, M., Perry, B. and Unruh, A., "Experience with the InfoSleuth Agent Architecture", *In the Proceedings of AAAI-98 Workshop on Software Tools for Developing Agents*, 1998.

[78] Nodine, M. and Chandrasekhara, D., "Agent Communication Languages for Information-Centric Agent Communities", In *Proceedings of the Hawaii International Conference on System Sciences*, 1999.

[79] Oates, T., NagendraPrasad, M. V. and Lesser, V., "Cooperative Information Gathering: A Distributed Problem Solving Approach," In AAAI Spring Symposium on Information Gathering in Heterogeneous, Distributed Environments, Stanford, CA, AAAI Press, 1995.

[80] Pastor, A., McKay, P. and Finin, W., "View Concepts: Knowledge-based Access to Databases", In *Proceedings of the First International Conference on Information and Knowledge Management*, Baltimore, MD, pp. 84-91, 1992.

[81] Perry, B., Taylor, M. and Unruh, A., "Information Aggregation and Agent Interaction Patterns in InfoSleuth", *In Proceedings Fourth IFCIS International Conference on Cooperative Information Systems*, CoopIS 99, IEEE Comput. Soc, Los Alamitos, CA, USA, pp. 314-24,1999.

[82] Piliavin, M., Rodin, J. and Piliavin, J., "Costs, Diffusion, and the Stigmatized Victim", *Journal of Personality and Social Psychology*, vol. 32, pp. 429-438, 1975.

[83] Salton, G. and Buckley, C., "A note on Term Weighting and Text Matching", TR90-1166, Department of Computer Science, Cornell University, 1990.

[84] Salton, G. and McGill, M., "Introduction to Modern Information Retrieval", McGraw-Hill, c1983.

[85] Saltor, F., Castellanos, M. and Garcia-Solaco, M., "Suitability of data models as canonical models for federated databases", SIGMOD RECORD, vol. 20, no. 4, pp. 45-49, Dec. 1991.

[86] Sandip, S., "Reciprocity: a foundational principle for promoting cooperative behavior among self-interested agents", In *Proc. of the Second International Conference on Multiagent Systems*, AAAI Press, Menlo Park, CA, pp. 322-329, 1996.

[87] Schwartz, S. and Howard, J., "Helping and Cooperation: A self-Based Motivational Model", *Cooperation and Helping Behavior: Theories and Research*, Derlega, V. and Grzelak, J. (Eds.), Academic Press, New York, USA, pp. 327-353, 1982.

[88] Simon, E., Distributed Information Systems: from Client/Server to Distributed Multimedia, McGraw-Hill, U.K., 1996.

[89] Shakshuki, E., Ghenniwa, H., Kamel, M., "An Architecture for Cooperative Information Systems", submitted to Journal of Knowledge-Based Systems, 2000.

[90] Shakshuki, E., Ghenniwa, H., Kamel, M., "Cooperative Agents for Information Gathering", submitted to Journal of Applied Artificial Intelligence, 1999.

[91] Shakshuki, E., Ghenniwa, H., Kamel, M., "A Multi-Agent System Architecture for Information Gathering", *In the Proceedings of the First International Workshop on Web Agent Systems and Applications (WASA-2000)*, London, UK, pp. 732-736, September, 2000.

[92] Shakshuki, E., Ghenniwa, H., Kamel, M., "Information Gathering System: Internet Navigation", *In the Proceedings of the IJCAI Workshop on Intelligent Information Integration*, Stockholm, Sweden, Sun SITE Central Europe (CEUR)-WS, vol. 23, July, 1999.

[93] Sheth, A. and Larson, J., "Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases", *ACM Computing Surveys*, Vol. 22, No. 3, pp. 183-236, Sept. 1990.

[94] Sheth, B., A Learning Approach to Personalized Information Filtering, *Masters Thesis*, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 1994.

[95] Smith, R. G., "The Contract Net Protocol: High Level Communication and Control in a Distributed Problem Solver", *IEEE Transactions on Computers*, Vol. 29, No. 12, pp. 1104-1113, 1980.

[96] Sycara, K and Pannu, A., "The RETSINA: Multiagent System Towards Integrating Planning, Execution and Information Gathering", In *Proceedings of the Second International Conference on Autonomous Agents*, ACM, New York, NY, USA; pp. 350-1,1998.

[97] Thomas, G., Thompson, G., Chung, C., Barkmeyer, E., Carter, F., Templeton, M., Fox, S. and Hartman, B., "Heterogeneous Distributed Database Systems for Production Use", *ACM Computing Surveys*, vol.22, No. 3, pp. 237-266, 1990.

[98] Wagner, T., Garvey, A. and Lesser, V., "Complex Goal Criteria and its Application in Design-to-Criteria Scheduling. In *Proceedings of AAA-97*, Providence, Rhode Island, pp. 294-301, August, 1997.

[99] Wellman, M. P., "A Market-Oriented Programming Environment and its Application to Distributed Multicommodity Flow Problems", *Journal of Artificial Intelligence Research*, vol. 1, pp. 1-22, 1993.

[100] Wellman, M.P., Durfee, E.H., Birmingham, W.P., "The Digital Library as Community of Information Agents", IEEE Expert 11(3), pp. 10-13, June 1996.

[101] Woelk, D. and C. Tomlinson, "InfoSleuth: Networked Exploitation of Information Using Semantic Agents", COMPCON Conference, March 1995.

[102] Woelk, D. and C. Tomlinson, "The InfoSleuth Project: Intelligent Search Management via Semantic Agents", Second International World Wide Web

[103] Woelk, D., P. Cannata, M. Huhns, W. Shen and C. Tomlinson., "Using Carnot for Enterprise Information Integration", Second International Conference on Parallel and Distributed Information Systems, pp. 133-136, January, 1993.

[104] Woelk, D., W. Shen, M. Huhns and P. Cannata, "Model Driven Enterprise Information Management in Carnot", in Charles J. Petrie Jr., ed.,. Enter-

prise Integration Modeling: Proceedings of the First International Conference, MIT Press, Cambridge, MA, 1992.

[105] Vidal, J., Durfee, E., "Task Planning Agents in the UMDL", In *Proceedings CIKM'95*, Workshop on Intelligent Information Agents, December 1-2, 1995.