

An Integrated Reasoning and Learning Environment for WWW Based Software Agents for Electronic Commerce

Behrouz Homayoun FAR[†], Member, Sidi O. SOUEINA[†], Hassan HAJJI[†], Shadan SANIEPOUR[†], and Anete Hiromi HASHIMOTO[†], Nonmembers

SUMMARY A major topic in the field of network and telecommunications is doing business on the World Wide Web (WWW), which is called Electronic Commerce (EC). Another major topic is blending Artificial Intelligence (AI) techniques with the WWW. In the Ex-W-Pert Project we have proposed an agent model for EC components that blends the traditional expert systems' reasoning engine with a multi-layer knowledge base, communication and documentation engines. In this project, EC is viewed as a society of software agents, such as customer, search, catalog, manufacturer, dealer, delivery and banker agents, interacting and negotiating with each other. Each agent has a knowledge-base and a reasoning engine, a communication engine and a documentation engine. The knowledge-base is organized in three layers: skill layer, rule layer and knowledge layer (S-R-K layers). In this project, for each EC agent, we identify the class of problems to be solved and build the knowledge base gradually for each layer. We believe that using this multi-layer knowledge base system will speed up the reasoning and ultimately reduce the operation costs.

Key words: *software agent, WWW, expert system, case-base classifier, CORBA, uncertainty management, integer programming*

1. Introduction

The World Wide Web (WWW) allows people at remotely located sites to communicate and share their ideas using a common communication protocol that can handle text files, images, sounds, forms, etc. WWW embodies a tremendous amount of information and this amount is increasing rapidly [7]. Using WWW, one can address a file by simply calling the protocol (*HTTP*, *FTP*, etc.), host, access port and path, respectively. Presently, a common use of WWW system is running a client application, using a browsing tool, pointing at a local or proxy server to browse data written in the hypertext format. The hypertext files contain anchors addressing other URLs and making connection to other servers possible.

A new challenge in AI is extending the functionality of traditional expert systems to be able to work on the WWW. Electronic Commerce (EC) is a potential application for such expert systems. In this project, EC is viewed as a society of software agents, such as customer, search, catalog, manufacturer, dealer, delivery

and banker agents, distributed over the Internet and interact and negotiate with each other. Each EC agents is an expert in it's own field. It has a knowledge-base and a reasoning engine, a communication engine and a documentation engine. The knowledge-base is organized in three layers: skill layer, rule layer and knowledge layer (S-R-K layers). In this project, for each EC agent, we identify the class of problems to be solved and build the knowledge base gradually for each layer.

The structure of this paper is as follows. In Sect. 2 an agent oriented model of EC is introduced. In Sect. 3 the internal structure of the EC agents is introduced and a viewpoint on knowledge representation in such agents is presented. In Sect. 4 protocols for cooperation and knowledge sharing are discussed. Implementation issues are discussed in subsequent Sects. 5–9. Section 10 is devoted to a discussion on agent and agency, knowledge representation, EC and developing tools. Finally, a conclusion is given in Sect. 11.

2. Agent Model of EC

Conventional EC models are built upon functional components, such as *commodity information, ordering, payment and electronic brokerage* [25]. *Commodity information* includes a showcase of goods and services offered, mainly in the form of a simple *home page* or a complicated electronic shopping mall. This is the heart of EC and must include interactive catalogs and directories. Electronic catalog system offers more flexibility as compared to the conventional catalog repository and can answer to search requests. *WWW based ordering system* includes a mechanism for placing and collecting orders, processing and/or distributing them. *Payment system* features a fail-safe mechanism for exchanging goods for electronic money. Finally, *electronic brokerage* involves special mechanism for offering a more efficient or safer service, and various add-on tasks such as bidding and bargaining.

In this project, EC is viewed as a society of software agents that interact and negotiate with each other. We have devised 7 types of EC agents, namely: *customer, search, catalog, manufacturer, dealer, banker and delivery* agents. Figure 1 depicts an artificial society of such agents. Each agent is called an *Ex-W-Pert Agent*. The main functions of the EC system, i.e., commodity in-

Manuscript received May 5, 1998.

Manuscript revised July 15, 1998.

[†]The authors are with the Department of Information and Computer Sciences, Saitama University, Urawa-shi, 338-8570 Japan.

Table 1 Agent components of EC and their implemented functions.

Type	Commodity Information	Ordering System	Payment System	Brokerage System	Implemented Functions
Customer	Share	Share	Share	Share	1. customer identity management 2. WWW search system 3. catalog search system 4. ordering management 5. payment management
Search	Implement	—	Share	Share	1. WWW search system 2. customer identity management 3. reporting system 4. payment management
Catalog	Implement	Share	Share	—	1. catalog search system 2. catalog recording system 3. customer identity management 4. reporting system 5. payment management
Manufacturer	Implement	Share	Share	Share	1. ordering management 2. dealer management 3. stock management 4. catalog recording system 5. payment management
Dealer	Implement	Share	Share	Implement	1. customer management 2. stock management 3. ordering management 4. brokerage and sale 5. delivery management 6. payment management
Banker	Share	Implement	Implement	—	1. secure transaction system 2. customer management 3. payment management
Delivery	Share	Share	Share	Implement	1. routing management 2. customer management 3. dealer management 4. payment management

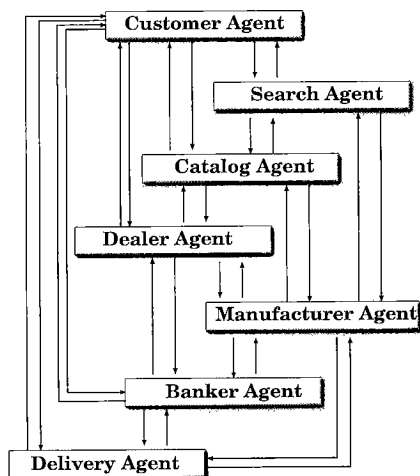


Fig. 1 Agent components of EC.

formation, ordering, payment and brokerage are either fully implemented on an agent or shared among a number of agents. These functions are further detailed to the list summarized in Table 1.

Each EC agent is an expert in its own field and

may interact with its human counterpart or behave autonomously. For instance, a *customer agent* receives instruction from a user to search the WWW space, to find a catalog and dealer for a desired product and to sign a contract. A *dealer agent*, on the other hand, identifies a customer's needs and takes parts in bidding to win the contract. A *delivery agent* competes to get a shipment contract.

The functions mentioned in Table 1 are almost self-explanatory. Some distinguishing features of this project are summarized below.

- Generating the menu items of the window based user interface using users' utility model, in *Customer agent* (Sect. 5).
- The ability of off line query processing, and using case-based knowledge to direct the search process, in *Search agent*. Together with a custom reporting facility that allows a user select the form and structure of the search report (Sect. 6) [15].
- Knowledge based guidance through the use of a case-based classifier, in *Catalog agent* (Sect. 7).

- Integer programming formulation of the Pickup and Delivery Problem (PDP), and devising an algorithm to handle PDP, in *delivery* agent (Sect. 8).
- Dealing with uncertain and non-deterministic business practice, such as bidding and competing to sell a product, in *Dealer* and *Manufacturer* agents, using game theoretic techniques (Sect. 9).

Concerning portability and modular design of the agents, we have used Java programming language to implement the functions and make use of HTTP and IIOP protocols for message based connections.

Presently, the Customer, Search, Catalog and Delivery agents are fully implemented. The Manufacturer and Dealer agents are implemented partially and the Banker agent is at the system design stage. Specific functions of the EC agents are described in Sects. 5–9.

3. Internal Structure of EC Agents

Figure 2 shows an overview of the internal structure of Ex-W-Pert agents. Similar to conventional expert systems, each agent has a *local knowledge-base* and a *reasoning engine*. Compared to the conventional expert systems, a main difference is that all agents have an additional *communication engine* and a *documentation engine*. The communication and documentation engines facilitate communication and navigation on the internet.

3.1 Reasoning Engine and Knowledge Base

Human experts when engaged with a goal-oriented task, try to achieve the goal within the constraints imposed by the task and avoid the *cognitive overload* [13] through selective utilization of their knowledge.

It is believed that human experts possess a conceptual (mental reference) model of how the objects in the external world interact based on standard operating procedures. Conceptual models have a hierarchical

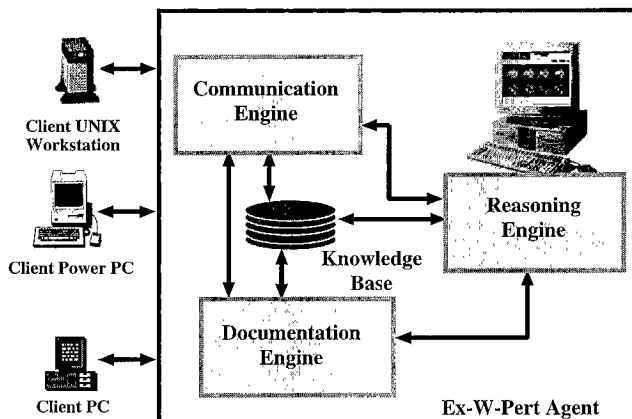


Fig. 2 Internal structure of a general purpose EC agent.

structure defined best by the Skill-Rule-Knowledge (S-R-K) levels [26] concerning with the routine, innovative and creative problem solving tasks, respectively. The multi-layer structure of the knowledge base, based on the S-R-K model is shown in Fig. 3.

level-1: Skill-based level:

This level deals with the *routine* tasks. It denotes the kind of task for which problem solving knowledge and strategies are well defined. At this level, reasoning is governed by stored patterns of predefined rules. Such context specific patterns are called *rules-of-thumb* (or symptomatic rules), that map directly from an observation to a ready-made solution.

At this level a query of an agent is accepted and by searching the knowledge-base, proper immediate action is selected. For instance, in case of a search agent, the query comes in the form of a list of keywords, submitted by the customer agent. Then search agent finds related keywords and conducts search using the new set of keywords.

level-2: Rule-based level:

This level deals with the *innovative* tasks when dealing with familiar or similar problems. It denotes the kind of task for which problem solving knowledge is well defined. Rule based behavior is conventionally described by case bases, decision tables, digraphs, fuzzy sets and natural language models [22].

At this level a query of an agent is accepted and a case data base is consulted to determine the action. For example, in case of a search agent,

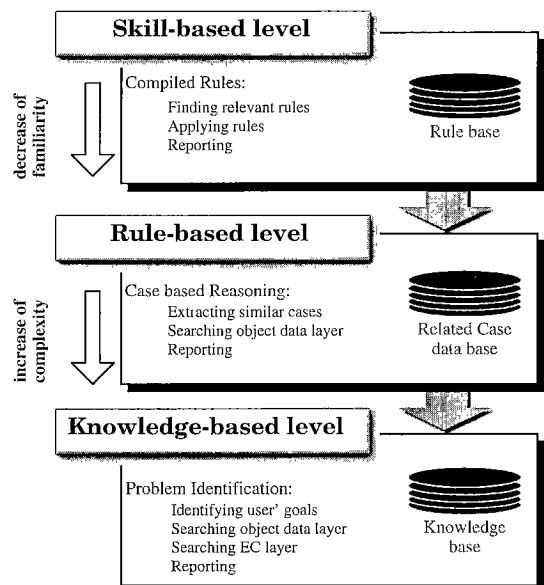


Fig. 3 Hierarchical knowledge base of the EC agents.

the initial query comes in the form of a sentence with reduced and restricted grammar [15]. This is adopted to avoid unnecessary overload of natural language processing. Then a set of similar cases are searched and cases matching the needs of the user are retrieved. Further search is conducted based on the instructions recorded on the matched cases.

level-3: Knowledge-based level:

This level deals with the *creative* tasks for which common patterns in stored knowledge form do not exist and reasoning should start from the so called *first principles*, starting from problem identification. In other words, neither problem solving knowledge nor the strategy is well defined.

At this level a query is accepted and the agent uses a KQML-CORBA based communication protocol (see Sect. 4) to consult with the other agents to determine the proper action. For example, a search is conducted by consulting a catalog agent, which in turn, contacts the dealer and/or manufacturer agents for proper information.

We believe that using this multi-layer knowledge base system will speed up the reasoning and ultimately reduce the operation costs.

3.2 Communication and Documentation Engines

EC agents must communicate with other agents in order to work flexibly and autonomously. In this project, we have considered building Java-based, KQML (Knowledge Query and Manipulation Language [23]) speaking and CORBA (Common Object Request Broker Architecture [11]) enabled agents that communicate over the Internet. The communication engine is mainly responsible for maintaining connection to the network, communicating with other agents and managing messages. The communication protocol is discussed in Sect. 4.2.

Documentation engine performs three main tasks:

- Acquiring data from the other agents, as requested by the reasoning and communication engines.
- Preparing and reformatting data items to be appropriate for transferring over the internet.
- Maintaining index of data items transferred and/or retrieved.

Using documentation and communication engines, the local knowledge can be shared with the other agents.

4. Cooperation and Knowledge Sharing

4.1 Formal Ontologies

Knowledge sharing, data integration, knowledge inter-

operation, and reuse requires a formal description of a representational vocabulary, including classes, relations and functions, which is called *ontology* [16]. Ontologies are considered as ways of specifying content-specific agreements for the sharing and reuse of knowledge [19].

In this project we have developed ontologies for both knowledge bases (e.g., *agency* and *common-task* ontologies) and commodities (e.g., *hardware-products* and *software-products*) that we experiment with. The *agency* and *common-task* ontologies describe inter-agent and intra-agent concepts, respectively. (See Sect. 4.3 for an example.)

4.2 Communication Protocol

The EC agents interact based on a 3-step protocol composed of *agent identification*, *query processing* and *payment processing* which is novel to this project. In the agent identification step, necessary data for identifying an agent is submitted and acknowledged. In query processing step the kind and contents of the required service is specified and in the payment processing step the price and payment method is negotiated.

The above mentioned protocol is handled by a message management mechanism depicted in Fig. 4. An incoming message is added to the end of the queue and the message handler decides upon a proper process by consulting the reasoning engine and appropriate layer of the knowledge base.

Concerning portability, modularity, security and transaction handling issues, the HTTP and IIOP protocols are selected for message based connections. HTTP is used for Web based connections and IIOP establishes the client-server relationships, remote method invoca-

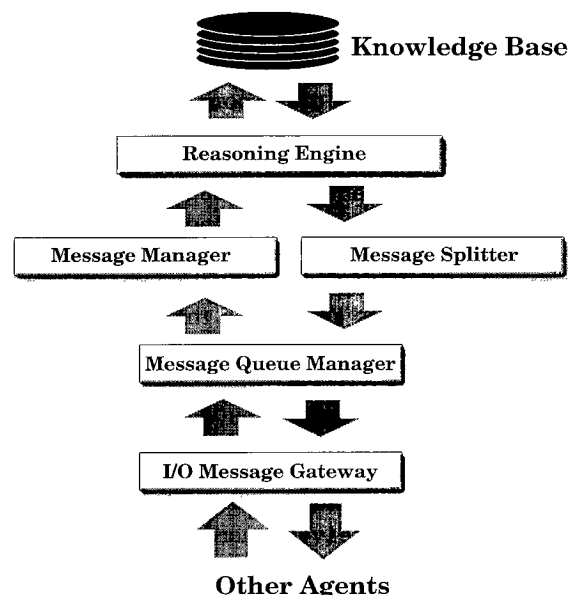


Fig. 4 Communication engine.

tion and integration of a wide variety of objects across a network that EC agents may require their services.

4.3 Knowledge Sharing Protocol

Knowledge sharing over the Internet is done via using KQML [23]. KQML is a semantics level communication language that is widely used for knowledge sharing and agent communication projects [6].

We have developed a socket based and an object based communication API for KQML messages. The socket based one, similar to other projects, allowing passing KQML messages encoded in ASCII form. The object based API for Java objects allows, first, simplifying the programming using command objects that take care of the communication issues in an abstract way, and second, as the API is installed on each machine, communication proceeds in the same manner whether or not an agent is on the same machine or on a different machine.

Here is a simple example illustrating the communication and knowledge sharing procedure. In this example, the *agency* and *hardware-products* ontologies (described in Sect. 4.1) are used.

Suppose that a *Customer* agent, say *Agent-a*, needs some additional information for a given product. In the first step, *Agent-a* asks for the information from an agent that most likely has such information, in this case, a *Catalog* agent *Agent-b* that *Agent-a* has already recorded in the friendly agents list. (See Sect. 4.4 for security zones.)

```
(ask-one
  :language KQML
  :ontology hardware-products
  :content (price(?x ?y))
  :zone friendly
  :sender Agent-a
  :receiver Agent-b)
```

If *Agent-b* does not have that data and this request fails, it first informs *Agent-a* of the consequences.

```
(sorry
  :in-reply-to S1
  :content (msg(check, report))
  :transaction Pending
  :sender Agent-b
  :receiver Agent-a)
```

Then *Agent-b* passes a query to the naming server to find a proper agent that has such data.

```
(ask-one
  :language KQML
  :ontology agency
  :content (catalog(?list))
  :sender Agent-b
  :receiver naming)
```

The name server, returns a list of the agents that may have the requested data and the security zone they belong to.

```
(tell
  :language KQML
  :ontology agency
  :content ((Agent-c, friendly),
            (Agent-d, trustable),
            (Agent-e, trustable))
  :sender naming
  :receiver Agent-b)
```

Based on the answer of the naming server, *Agent-b* decides to ask *Agent-c* to acquire the data.

```
(ask-one
  :language KQML
  :ontology hardware-products
  :content (price(?x ?y))
  :zone friendly
  :sender Agent-b
  :receiver Agent-c)
```

Upon a proper answer from *Agent-c*, such as

```
(tell
  :language KQML
  :ontology hardware-products
  :content (=price(PC, 184000))
  :sender Agent-c
  :receiver Agent-b)
```

now *Agent-b* can fulfill the original request of *Agent-a* by delivering this info to *Agent-a* and terminating the transaction.

```
(tell
  :language KQML
  :ontology hardware-products
  :content (=price(PC, 184000))
  :in-reply-to S1
  :transaction Terminated
  :sender Agent-c
  :receiver Agent-b)
```

Upon accepting the request, at the first place, *Agent-b* continues to search and check until the goal of the *Agent-a*, and consequently the user, is satisfied. Note that, considering user friendliness, only part of the process is visible to *Agent-a* and the user. They need not be aware of all the details of the communications and negotiations that *Agent-b* has gone through.

4.4 Security

Basically, the security service of CORBA's common services is adopted to build fine grained locally secure transaction and business processes.

Furthermore, we have devised three security zones for the agents to operate, i.e., *friendly*, *trustable* and *hostile* [29]. Each agents can operate in any of the security zones when dealing with the other agents. In the *friendly* zone the information is exchanged freely. In the *trustable* zone check and verification steps are added using public coding and decoding keys [12].

Agents competing for scarce resources may apply individual strategies to increase their own utilities, there-

fore they may operate in the *hostile* zone. In this project we have used game theoretic and machine learning techniques to design a multi-agent learning language (MALL) [30] to handle this case (see also Sect. 9).

5. Implementation: Customer Agent

In customer agent, we have considered building a window-based user interface to serve both the beginner and advanced users. A novel feature is generating the menu items using user utility model. This is the first step towards migrating from current direct manipulation user interface to indirect management interface (i.e., digital secretary). Figure 5 shows examples of the user interface windows for the search agent [15]. The interface unit is composed of a set of hierarchically related windows. It starts with the most simple functions and becomes more complicated as going down the hierarchy. Usually, the first interface window depicts the main functions of the system together with troubleshooting and help menus. By clicking on a button, another window is popped up that offers more detailed functions. The way of generating dynamic menus is described here-with.

Building an accurate user model is essential to decision making and decision-support tasks. A model of the user’s preferences is required to make proper decisions or to suggest goods alternatives. While much work on reasoning under uncertainty in AI has focused on providing representations and tools for reasoning about belief, relatively little work has addressed elicitation of utility methods. Reasoning with influence diagrams do focus on the problem of selecting the best action assuming that the utility function is given. The models

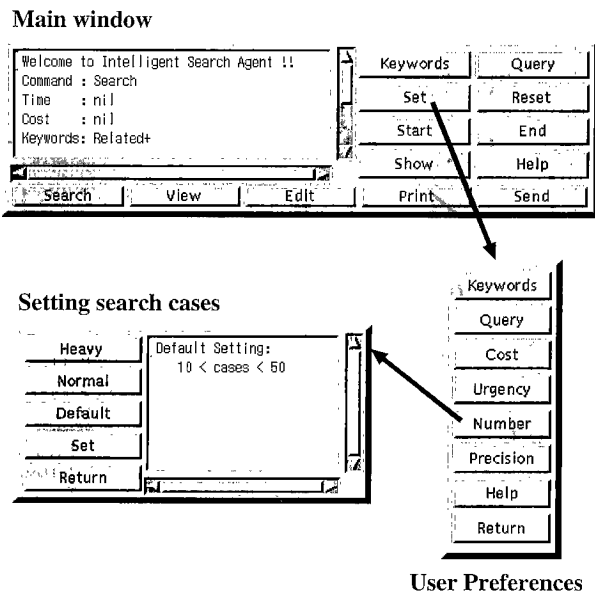


Fig. 5 Example of user interface.

for reasoning about utility in AI is pointed out in [33] which attempted to define a Bayesian network-like models for reasoning about utilities. In the same direction, it was shown that utility functions are decomposable as Markov networks [5].

In this work, we are particularly interested in automatic elicitation of user utility model. The example is of a user interacting with the customer agent. If the agent keep a model of the user’s utility model, then the user critiques will help to gain a better understanding of the user preferences and, thus, present the user quickly items that are most likely to interest to the user. Figure 6 depicts the scenario.

Basically, the customer agent presents the user with a menu displaying a set of candidates. The user selects one candidate. Based on the user’s choice the agent updates its belief about the user’s preferences and upon this new belief, passes a query to the other agents to retrieve the items that may interest the user. The agent is required to identify the most relevant feature to the problem. This information is used in the search of new candidates or in the formulation of questions to the user. This scenario continues until identification of solution to the user’s utility maximization problem.

While this problem has been extensively studied in the field of decision analysis, eliciting utility for interactive decision making raises new issues and offers new ways to facilitate the process of acquisition of the utility model. We cite in particular:

- The system has access only to a partial imperfect knowledge of the user model and using this local

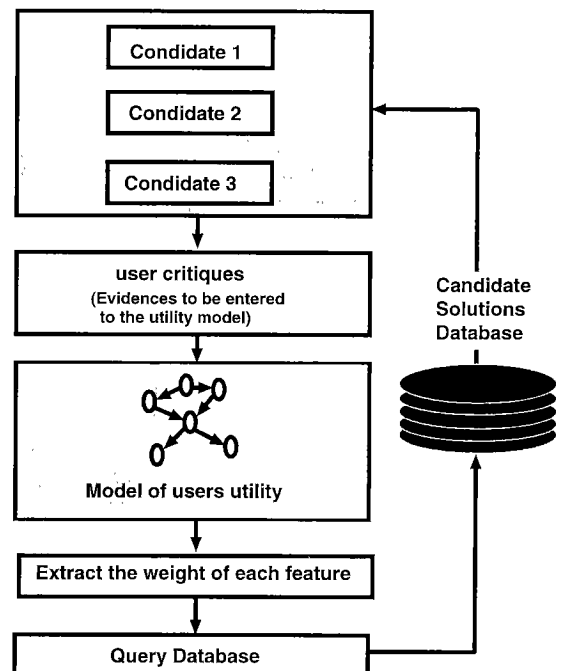


Fig. 6 Incremental elicitation of user utility model.

information, the system should reason about global dominance among alternatives.

- The elicitation process and the inference should take only a reasonable time. This implicitly imply that the system has to reason about information that matter most for the user choice, while dropping less influencing information.
- Effective display of the information. This includes facilitation of the user feedback and generating explanation.

6. Implementation: Search Agent [15]

Figure 7 shows how the customer agent receives an instruction and transfers it to a search agent, and how the search agent retrieves and summarizes the acquired data by attending to the user's questions, clarifying the needs, conducting search, fetching data, selecting relevant data items, arranging and ordering the data, and making a report of the results.

A user interacts with the customer agent by either specifying a number of keywords or an input query. The system considers activating the relevant level of skill-rule-knowledge based search. In each case a proper set of keywords and search attributes are identified. The "WHERE" package is used to communicate with the other index servers on the internet. The outcome of inquiry comes in the form of lists of candidate URLs that possibly contain the data. These lists may have superfluous and partially unrelated data items that are trimmed and based on the scoring algorithm a proper

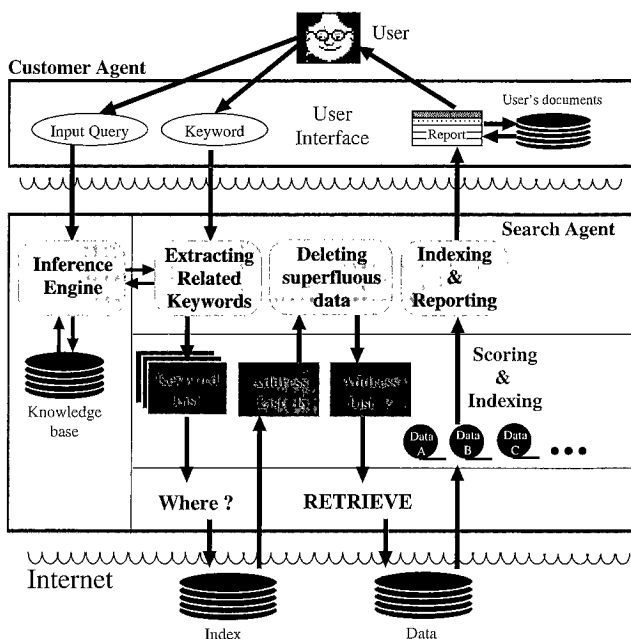


Fig. 7 Search agent.

list of URLs is devised. Then the "RETRIEVE" package is used to actually fetch the data and the "REPORT" package is used to index and make a report of the thus acquired data. This report is displayed to the user on demand.

7. Implementation: Catalog Agent

In catalog agent, the main idea is to have an expert to help the customers know more about products, so that they can select the one better matched to their request. This has advantage for manufacturers in the way that will let them advertise their products, and for customers, because will help them find their request.

Figure 8 shows how a catalog agent receives and processes requests from the other agents (see example in Sect. 4.3) and interacts with a database of goods in order to retrieve the kind of information which is not available through conventional search engines. The arrows depict the passage of messages between catalog agent and the other agents. On the right side, the mechanism of generating active html files using Oracle Web Request Broker (http://www.oracle.com/products/asd/was/collateral/was30_twp.html) is depicted.

Manufacturers and dealers first record their goods and services in the data base and the catalog agent retrieves the relevant data items from the data base and produces a hypertext catalog document that fits the customers' needs. This document is produced on the fly, based on the instructions given by the customer agent.

The catalog agent will let the customer know about the specification that each product can have, through knowledge navigation [10], [20].

Each product in the database is recorded with its features. The features are mostly given in numerical values. Each feature may have a range of variation, which

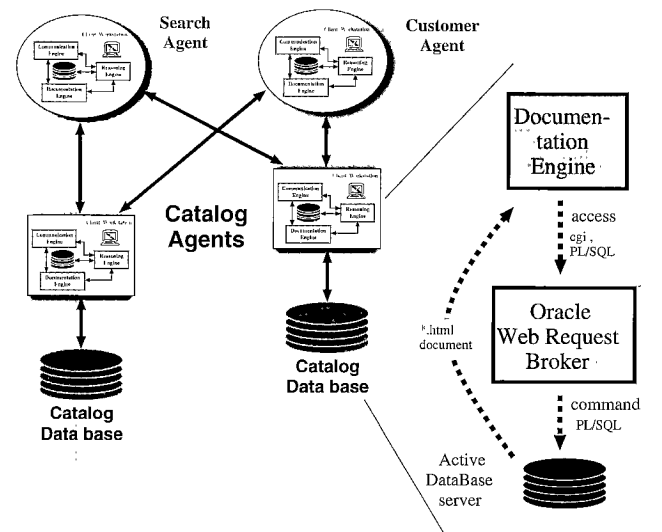


Fig. 8 Catalog agent.

is divided into limited number of qualitative clusters (e.g., low, medium, high and very high). Clustering the features in these groups, will let the user not only know about the values of each feature, but also know about the position of that product in the typical products. Besides, when the user indexes some products with a feature, this kind of clustering, will enable the system show the variation in that feature. So each feature in the database will not only be shown with its exact value, but also with the cluster which it belongs to.

A new feature is knowledge based guidance through the use of a case-based classifier. The case-based classifier has a set of data previously clustered. For each new product it will just look for how similar is the current product to each group. The most appropriate group is the most similar one. For most of the products besides the specification of each product that is described by the features, it is necessary to have a general type (e.g., Computer for personal use, Computer for official use, etc.). The type of a product might also be one of its features (e.g., the number of pins for ICs). The type of product is the main category for each product, and is the most important issue, which the system will introduce to the customer in the first place. When a new product arrives, case-based classifier finds the similarity and clusters each feature. It will also put the new product in the appropriate type.

When a user requests for some products, the system will show one sample for each type including the features which are important in that product (in the example of computer, the features can be speed, memory capacity, etc.). If the user is aware of the exact specification, the only thing to do is choosing one type and define the value for each attribute and then submit this request to the catalog agent. The catalog agent searches for that request and finds the exact match, or if doesn't exist, the most similar one, in its database. For finding similarity we consider weight for the features and associate the more important features with higher weight. If the user doesn't know the exact variation in each product and doesn't have a clear idea about how the features vary, he/she can choose one type (in our example let's consider computer for personal use which is type A), and say "*I want a ... (computer) type ... (A), but please show me the variation in feature, let's say speed.*" Now the user can choose one group and say "*I want speed Medium, but please show me the variation in feature ... (e.g., memory capacity).*" The catalog agent indexes the computers which are in type A and the speed is medium, by the feature memory capacity. This routine can go on until the user gets aware of his request and order it.

Apparently in each step the user is free to ask the system to show the variation in one feature, or directly define the value for each feature. When the user directly defines values, the system will ask him which features are not important to him (don't care features and while

searching will not assume those don't care features), as constraint. The system also has the ability of warning the user of possible trade-offs. When it completes the database search, it will notice if there is a direct or inverse relation between any two feature in each product, which is repeated among all items. Then will consider it as a rule. For each new product if the rule is true, it will continue its belief about that rule, but if the new product is conflicting, it will cancel the rule (learning). Then whenever it finds some failure in its search (the user asked for an item but it can not find the exact or even similar match) then it will try to explain the failure with its rules. If any rule matched to the current situation, it will accept it as a reason and warn the user about that trade-off. The final result of the Catalog Agent is that the user knows about his exact request and explicitly can explain what he wants.

8. Implementation: Delivery Agent

The delivery agent, among the other functions depicted in Table 1, is mainly responsible for delivering goods to customers as agreed during the negotiation between the dealer agent and customer agent. The delivery agent compiles the requests into a plan that assigns the requests to vehicles, and for each vehicle, constructs a route through the locations assigned to that vehicle so that all side constraints are satisfied, and the total cost involved executing the routes is minimal.

We recognized that the delivery agents' planning task falls into the class of Pickup and Delivery Problems class (PDP), and devised an algorithm to handle this, using integer programming.

The PDP problem in its most basic form consists of a fleet of vehicles and a set of customers' requests. Each vehicle has a given capacity, a start location and an end location. Each transportation request specifies the size of the load to be transported, the locations to pick up (the origins) and the locations to deliver (the destinations). Each load has to be transported by one vehicle from its set of destinations without any trans-shipment. Each request specifies, at least, an origin location and a destination location. The vehicle travels through the locations so that each origin is visited before the corresponding destination.

PDP problem has a number of variations. The problem characteristics are quite important in designing a fast algorithm. For example, the model depends highly on whether the requests are serviced by a single agent or multiple agents. Also a static domain requires customers to specify their requests before any solution is searched. A dynamic model on the other hand, allows customers to request services after the vehicles are dispatched. The objective functions applicable to routing problems include minimizing the number of vehicles and route length and also maximize the customers' satisfaction.

Optimal methods typically involve one of the three approaches: dynamic programming, mathematical programming and branch-and-bound methods [31]. Here we use the integer programming method with focus on formulating a single vehicle PDP problem, since for multiple vehicles a satisfactory general solution has yet to be developed.

Let \mathcal{N} be the set of transportation requests. For each transportation request $i \in \mathcal{N}$, a load size $\bar{q}_i \in \mathcal{N}$ has to be transported from the origins N_i^+ to the set of destinations N_i^- . Each load is subdivided as follows:

$$\bar{q}_i = \sum_{j \in N_i^+} q_j = - \sum_{j \in N_i^-} q_j$$

Where q_j is positive for pickups requests and negatives for delivery requests. We define also the set of origins as the union of all origin points and the set of destinations as the union of all destination points, that is $N^+ = \cup_{i \in \mathcal{N}} N_i^+$ and $N^- = \cup_{i \in \mathcal{N}} N_i^-$. Furthermore, the vehicle has a capacity Q , a start location $+o$ and an end location $-o$. Let $G = (V, E)$ the graph representing possible locations the agent can visit, and E the set of edges connecting the locations.

Definition 1: A pickup delivery problem route R for a vehicle is a directed route through a subset S of V such that:

1. R starts from a node in the start location $+o$ and ends in an end location $-o$.
2. All locations in N_i^+ are visited before all locations in N_i^- .
3. The vehicle visits each location in $N^+ \cup N^-$ only once.
4. The vehicle load never exceeds Q .
5. The route R ends in $+o$.

The definition states that a feasible pickup delivery route starts in a start node, and ends in an end node. It also visits each location only once making sure that goods are previously loaded in pickup nodes before visiting delivery nodes. The vehicle load should not exceed the vehicle capacity.

To model then our pickup delivery problem we have to translate this conditions into constraints of an Integer Programming Problem (IPP). If we let $x(e)$ the variable defined as:

$$x(e) = \begin{cases} 1 & \text{if } e \text{ is chosen in the optimal route} \\ 0 & \text{otherwise} \end{cases}$$

And $c(e)$ the cost associated with the edge e then we can express formally the constraints as follows:

Condition 1: Node balance

Each node should be visited only once. It follows that from all edges incident to a any node, exactly 2 edges are chosen in the optimal route. With each node $v \in V$ we associate the set:

$$\delta(v) = \{(u, v) : (u, v) \in E\}$$

The node balance condition can, then, be written as follows:

$$x(\delta(v)) = 2$$

Condition 2: Reference direction

This condition is added only to make the formulation easier. Namely, The agent is supposed to start at $+o$ and end at $-o$. We add the fact that the agent has to come back to the start location $+o$. We write

$$x((-o, +o)) = 2$$

Since all routes end in $-o$, adding the cost of coming back to $+o$ the objective functions does not influence the solution. This condition is added to make the subsequent formulation easier.

Condition 3: No Subtours

The agent should visit all the cities making a unique big cycle (Hamiltonian cycle). For each subset of nodes U , let $[U : \bar{U}]$ the set defined as follows (called the cutset of U):

$$[U : \bar{U}] = \{(u, v) \in E : u \in U; v \notin U\}$$

The no Subtours conditions is, then, written as:

$$x([U : \bar{U}]) \geq 2$$

Condition 4: Precedence

Each pickup node, denoted by $+i$, should be visited first before its corresponding delivery node, denoted by $-i$. We first construct the set:

$$U'_p = \{U \subseteq V \mid -o \notin U, \exists i \in V; \\ -i \in U \quad \text{and} \quad +i \notin U\}$$

The precedence condition can then be written as:

$$\forall U \in U'_p, x([U : \bar{U}]) \geq 4$$

The collection of feasible solutions to the PDP are the solutions to the set of constraints:

$$\begin{aligned} x((+o, -o)) &= 1 \\ x(\delta(v)) &= 2 \quad \forall v \in V \\ x([U : \bar{U}]) &\geq 2 \quad \forall U \subseteq V \\ x([U : \bar{U}]) &\geq 4 \quad \forall U \in U'_p \end{aligned}$$

Condition 5: Capacity

At each point, the capacity of the vehicle should not be exceeded. If we let y_i specify the load of the vehicle arriving at node i then the capacity condition is captured by the following equations:

$$\begin{aligned} y_{o+} &= 0 \\ y_l &\leq Q \quad (l \in N_i^+ \cup N_i^-) \\ x(e) = 1 &\Rightarrow y_i + q_i = y_j \quad (e = (i, j) \in E) \\ y_i &\geq 0 \end{aligned}$$

At this point, we wrote all the equations that a feasible route has to satisfy, but some of the constraints may be redundant (implied by other strong constraints) or implicit (i.e., satisfied with equality by all feasible solutions). For example, if $U = \{+o, -i\}$ then $x([U : \bar{U}]) = 4$. We construct the set

$$\mathcal{U}_p = \{U \subseteq V \mid 3 \leq |U| \leq |V| - 3, \\ \exists i \in V, +i \notin U, \text{ and } -i \in U\}$$

For the set of Subtours constraints, some constraints are implied by precedence constraints. For example, if U is a subset of nodes such that $-o \in U$ and $+i \notin U$, then the Subtours constraint for U is implied by the precedence constraint for $U - \{-o\}$. We construct then the set \mathcal{U}'_s that contains only irredundant subtour constraints.

$$\mathcal{U}_s = \{U \subseteq V \mid 3 \leq |U| \leq |V| - 2, \\ \forall i(+i \in U \Leftrightarrow -i \in U)\}$$

The reduced IPP containing only necessary constraints defining all feasible routes is then given by:

$$\begin{aligned} \min \sum_{e \in E} c(e)x(e) \\ x(\delta(v)) = 2 \quad (v \in V) \\ x([U : \bar{U}]) \geq 2 \quad (U \in \mathcal{U}_s) \\ x([U : \bar{U}]) \geq 4 \quad (U \in \mathcal{U}_p) \\ y_{o+} = 0 \\ y_l \leq Q \quad (l \in N_i^+ \cup N_i^-) \\ x(e) = 1 \Rightarrow y_i + q_i = y_j \quad (e = (i, j) \in E) \\ 0 \leq x(e) \leq 1 \quad (e \in E) \\ 0 \leq y_i \leq 1 \quad (i \in V) \end{aligned}$$

Solving this problem would be easy if the variables are not required to be integer (i.e., polynomial-time using Simplex algorithm or Interior Point Method). Forcing the solution to be integer makes the problem computationally expensive (NP-Hard [31]). Note however that it is possible to use the structure properties of the graph $G = (V, E)$ in a way to characterize the facets of the convex hull of the set of all feasible solutions. If such knowledge is available, the facets are exactly the constraint to be added to the initial IPP such that solving the IPP with standard Linear Programming approaches would return an integer optimal solution. Unfortunately, for a large class of combinatorial problems, only a partial linear description of their convex hull is known [1].

Given the complexity of PDP and routing problems in general, many algorithms do not seek global optimal solutions, such as, a branch-and-cut algorithm that solved a small size PDP [28] and a column generation approach to the PDP [35]. Note however that these techniques are more suitable for off-line computation due to the large time it takes to find the solution. Developing approximate algorithms that supplies

in real-time the delivery agent, during interaction with the dealer agent, with the *near-optimal* action to carry out is an incentive for more research.

9. Implementation: Dealer and Manufacturer Agents

We have developed a Multi-Agent Learning Language (MALL), aimed to make agents learn from their environment about the strategies of their opponents and later make their own plans depending on the other players' moves. The MALL language is currently bundled with the *dealer* and *manufacturer* agents when competing against each other by applying individual strategies attempting to increase their own *utilities*. Details of the language and implementation are given in [29], [30].

For instance, a dealer applying a strategy with lower cost and better services to the customers is more likely to increase its own income. For the purpose of this example, let us say that given the basic knowledge of the environment (the market), plus, the signals of dealer agent A is sending, the dealer agent B tries to learn A 's strategy. Obviously, agent A is omniscient about his own strategy and we can say that the game is zero-sum (since the benefit of A is the loss of B). Then B by analyzing the signals that A gives will get some hints. Agent A can be said to *play partially revealing* to make its opponents get the least indicating signals [39]. On the other hand, agent B in this case, would construct its own plan through a process of explanation and analysis of the information revealed by agent A .

10. Other Works and Discussion

10.1 Agent and Agency Concerns

A variety of methods and techniques have been suggested to deal with multi-agent environment [8], [21], such as cooperation [27], coordination [2], interoperability [18], open agent architecture (KAoS) [9], and Agent Oriented Programming (AOP) [32].

Among the well defined and technically implementable features specified as agents' attributes, the followings are implemented in this project:

- Reactivity (i.e., selectively sense and act, e.g., dealer agent).
- Autonomy (i.e., goal directness, proactive and self-started behavior, e.g., search agent).
- Collaboration (i.e., work with other agents and entities to achieve a common goal, e.g., customer and catalog agents).
- Knowledge-level communication ability (i.e., communicate with other entities in a language higher level than symbol level program to program protocols, e.g., dealer agent).

- Inferential capability (i.e., act on abstract task specification, using models of self, situation, and/or other agents, e.g., customer and dealer agents).
- Temporal continuity (i.e., persistence of state and personality, e.g., customer agent).
- Adaptation (i.e., learn and improve with experience, e.g., customer, search and dealer agents).

However, other features, such as mobility (i.e., migrate from one host to another in a self-directed way) are considered unnecessary, therefore not implemented.

10.2 Knowledge Representation and Reasoning Concerns

In a multi-agent system, dealing with novel situations in an innovative or creative way is a necessity. There are already some representations and models suggested to deal with innovation and creativity. For example, the model by Kolodner and Wills[38] includes re-interpretation, evaluation, assimilation, strategic control. The initial formulation of the problem to be solved may be incomplete, contradictory and under-constrained. The process of understanding and formulating the problem is re-interpretation. Evaluation is the process of weighting the alternative solutions. Assimilation is a means for calling relevant solution alternatives, and strategic control is defined as breaking typically rigid control structure and allow more interaction among processes.

In re-interpretation and assimilation, adaptation is a necessity[17]. It allows the agent to consider the knowledge missed from the other parts and derive a better solution. The idea of using adaptation knowledge in the other procedures of reasoning has been previously used in many works[24],[36],[37].

Based on the idea of interacting the adaptation knowledge with the other knowledge sources, we have made a kind of network in which the nodes are the constraints which represent a new problem to be solved. We make the link between two nodes if there is any adaptation knowledge which may inform us having one of the constraint in the problem description allows us to adapt the solution in the way that satisfy the second constraint. The direction of each link shows that adaptation in which direction is possible. This allows us to use all adaptation knowledge available, combine them, and depict them in the network. Therefore, having a new constraint we can move through the network regardless of which knowledge we are using and find possible interpretation for the new problem[34].

10.3 Electronic Commerce Concerns

Some current issues of EC and representative projects are mentioned in [3],[4], and [25]. The generic EC

model may be viewed as a set of *actors* using *media* performing business *actions*. The *actors* are providers (dealers, merchants, etc.), and consumers (users, information seekers). The *media* is composed of multimedia objects (documents, images, software, etc.) which represent goods or services. The *actions* are business processes, such as stock management and payment processing.

Almost all of the projects mentioned in [3] and [25], concentrate on either of the *actors*, *media* or *actions*. They mainly focus on identification and implementation of EC functions and business processes. At this moment, there is no proposed model other than ours to cover the whole set of *actors*, *media* and *actions*. Furthermore, in the other projects, the consumers are assumed to have a wide variety of domain expertise[4]. We, on the contrary, assume that the customers are ordinary and casual users and provide them with tools and techniques, borrowed from AI, to share the expertise with them. We think that implementing and applying AI techniques to EC is a new and challenging task for both fields. In this project we propose application of the Ex-W-Pert system architecture[14],[15] for such implementation, that is, adding the communication and documentation engines to the conventional expert systems to help the *actors* and put the burden of knowledge processing on the system rather than the *actor* itself.

The idea of EC, by means of Electronic Data Interchange (EDI) and private or local area networks (LAN) has been around for some years. What internet based EC adds to this is changing from private networks to a global network together with interactive data interchange, with a reasonable cost, but at the expense of lower security. Moving to a global network implies less control over the kind and amount of data available.

10.4 Developing Tools

There are already some agent developing tools available, mostly implemented in Java programming language, such as Agent Building Environment (ABE), Java Agent Template (JAT), Java Expert System Shell (JESS), and enhanced with Object Request Broker (ORB), suitable for distributed object platforms, such as Voyager and Open Agent Architecture (OAA), JKQML (<http://www.alphaWorks.ibm.com/formula/jkqml>) and AgentBuilder (<http://www.agentbuilder.com>). Although such tools and environments can facilitate the communication (for example, using KQML language in JAT as the knowledge level communication) and message passing (for example, using HTTP and IIOP protocols in ABE and OAA, respectively), but they fail to provide the appropriate aids to build extensive knowledge bases in the sense we use in this project.

One exception is Open Knowledge Base Connec-

tivity (OKBC) (<http://www.ai.sri.com/~okbc/>) for accessing knowledge bases stored in knowledge representation systems using a model of knowledge representation systems based on a common conceptualization of classes, individuals, slots, facets, and inheritance. The protocol supports both networked and direct access to knowledge bases. However, OKBC is mostly an interface to the knowledge based system and does not help building the knowledge base itself in the multi-layer fashion described in Sect. 3.1.

11. Conclusion

In this paper, a general model for Electronic Commerce (EC) components was introduced and the ways of implementing intelligence into a WWW based agents was discussed. An architecture for WWW based multi-layer intelligent EC agents was demonstrated. Present issues and implementation status of such agents of the EC family was discussed in detail.

Acknowledgments

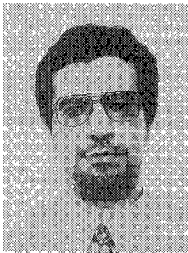
The authors are grateful to Oracle Japan for donating the data-base and WWW servers that were used to build the experimental agents of this project.

References

- [1] K. Aardal and C.P.M. Van Hoesel, "Polyhedral techniques in combinatorial optimization I: Theory," *Statistica Neerlandica*, vol.50, pp.4–26, 1996.
- [2] M.F. Abraham, "Modern Sociological Theory," Oxford University Press, 1982.
- [3] N. Adam and Y. Yesha, eds., "Electronic Commerce: Current Research Issues and Application," Springer Verlag, p.155, 1996.
- [4] N. Adam and Y. Yesha, "Strategic directions in electronic commerce and digital libraries: Towards a digital agora," *ACM Computing Surveys*, vol.28, no.4, pp.818–835, 1996.
- [5] F. Bacchus and A. Grove, "Graphical models for preference and utility," 11th Conference on Uncertainty in Artificial Intelligence (UAI-95), 1995.
- [6] D. Benech and T. Desprats, "A KQML-CORBA based architecture for intelligent agents communication in co-operative service and network management," *IFIP/IEEE MMNS '97*, 1997.
- [7] T. Berners-Lee, R. Cailliau, H.F. Nielsen, and A. Secret, "The world-wide web," *Communications of The ACM*, vol.37, no.8, pp.76–82, 1994.
- [8] J.M. Bradshaw, ed., "Software Agents," MIT Press, p.480, 1997.
- [9] J.M. Bradshaw, S. Dutfield, P. Benoit, and J.D. Wooley, "KAoS: Toward an industrial-strength open agent architecture," *Software Agents*, J.M. Bradshaw, ed., pp.375–418, MIT Press, 1997.
- [10] R.D. Bruke, K.J. Hammond, and B.C. Young, "Knowledge-based navigation of complex information spaces," 13th National Conference on artificial Intelligence, AAAI'96, pp.462–468, 1996.
- [11] "The common object request broker: Architecture and specification," Object Management Group (OMG), Revision 2.2, 1998.
- [12] D. Denning, "Protecting public keys and signature keys," *IEEE Comput.*, 1983.
- [13] B.H. Far and M. Nakamichi, "Qualitative fault diagnosis in systems with non-intermittent concurrent faults: A subjective approach," *IEEE Trans. Syst., Man. & Cybern.*, vol.23, no.1, pp.14–30, Jan./Feb. 1993.
- [14] B.H. Far and Z. Koono, "Ex-W-Pert system: A web-based distributed expert system for groupware design," *World Congress on Expert Systems' 96*, pp.545–552, 1996.
- [15] B.H. Far and Z. Koono, "Ex-W-Pert system: A WWW based intelligent search engine for electronic commerce," *Joint Conference on Knowledge-Based Software Engineering, JCKBSE' 96*, pp.174–181, 1996.
- [16] A. Farquhar, R. Fikes, and J. Rice, "The ontolingua server: A tool for collaborative ontology construction," *Knowledge Systems Laboratory, KSL-96-26*, 1996.
- [17] F. Gebhart, A. Voss, W. Grather, and B. Scmidth-Belz, "Reasoning with Complex Cases," Kluwer Academic Publishers, 1997.
- [18] M.R. Genesereth, "An agent based framework for interoperability," *Software Agents*, J.M. Bradshaw, ed., pp.317–345, MIT Press, 1997.
- [19] T.R. Gruber, "Toward principles for the design of ontologies used for knowledge sharing," *Formal Ontology in Conceptual Analysis and Knowledge Representation*, N. Guarino and R. Poli, eds., Kluwer Academic Publishers, 1993.
- [20] K.J. Hammond, R. Bruke, and K. Schmitt, "A case base approach to knowledge navigation," *Case-Based Reasoning, Experiences, Lessons, and Future Directions*, MIT Press, 1997.
- [21] M.N. Huhns, and M.P. Singh, "Readings in Agents," Morgan Kaufmann Publishers, 1997.
- [22] G. Johannsen, "Categories of human operator behavior in fault management situations," *Tasks, Errors and Mental Models*, L.P. Goodstein, H.B. Andersen, and S.E. Olsen, eds., Taylor and Francis, ch.16, pp.251–258, 1988.
- [23] "DRAFT Specification of the KQML Agent- Communication Language," The DARPA Knowledge Sharing Initiative, 1993.
- [24] D.B. Leake, A. Kinely, and D. Wilson, "Case-Based Similarity Assessment: Estimating Adaptability from Experience," 14th National Conference on Artificial Intelligence, AAAI Press, Menlo Park, CA, 1997.
- [25] R. Pyle, ed., "Special issue on electronic commerce on the Internet," *Communications of The ACM*, vol.39, no.6, 1996.
- [26] J. Rasmussen, "The role of hierarchical knowledge representation in decision making and system management," *IEEE Trans. Syst., Man & Cybern.*, vol.SMC-15, no.2, pp.234–243, March/April 1985.
- [27] J.S. Rosenschien, "Rational interaction: Cooperation among intelligent agents," Ph.D. Dissertation, Computer Sciences Dept., Stanford University, Stanford CA, 1985.
- [28] K.S. Ruland, "Polyhedral solution to the pickup and delivery problem," Ph.D. Dissertation, Washington University, 1995.
- [29] S.O. Soueina, B.H. Far, T. Katsube, and Z. Koono, "MALL: A Multi Agents Learning Language for Competitive and Uncertain Environment," *IEICE TRANS. Inf. & Syst.*, vol.E81-D, no.12, pp.1339–1349, 1998.
- [30] S.O. Soueina, H. Hashimoto, S. Saniepour, H. Hajji, and B.H. Far, "A learning language for intelligent agents," 1998 JSAI Annual Conference, pp.578–579, 1998.
- [31] A. Schrijver, "Theory of Linear and Integer Program-

ming," Wiley-Interscience, 1986.

- [32] Y. Shoham, "Agent-oriented programming," *Artificial Intelligence*, vol.60, pp.51-94, 1993.
- [33] Y. Shoham, "A symmetric view of utilities and probabilities," 15th International Joint Conference on Artificial Intelligence, IJCAI 97, pp.1324-1329, 1997.
- [34] S. Saniepour, H. Hajji, S.O. Soueina, H. Hashimoto, and B.H. Far, "CBR in design: Towards more creative system using adaptation knowledge," 1998 JSAI Annual Conference, pp.154-155, 1998.
- [35] M. Sol, "Column generation techniques for pickup and delivery problems," Ph.D. Dissertation, Eindhoven University of Technology, 1994.
- [36] B. Smith and M.T. Keane, "Retrieving adaptable cases: The role of adaptation knowledge in case retrieval," 1st European Workshop, EWCBR-93, 1993.
- [37] W. Wilke, I. Vollrath, K. Althoff, and R. Bergmann, "A framework for learning adaptation knowledge based on knowledge light approaches," *Adaptation in Case Based Reasoning Workshop, ECAI 1996*, 1996.
- [38] L.M. Wills and J.L. Kolodner, "Towards more creative case based design systems," *Case Based Reasoning Experiences, Lessons, and Future Directions*, D. Leake, ed., MIT Press, 1996.
- [39] S. Zamir, "Repeated games of incomplete information," *Handbook of Game Theory*, Volume I, Chapter IV, 1992.

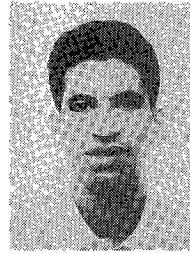


Behrouz Homayoun Far received BSc. and MSc. degrees in Electronic Engineering in 1983 and 1986, respectively, from Tehran University. He has received his Ph.D. degree from Chiba University - Japan, in 1990. He is currently an Associate Professor at the Department of Information and Computer Sciences, Saitama University - Japan. The research fields of his interest are qualitative and temporal reasoning, Knowledge acquisition

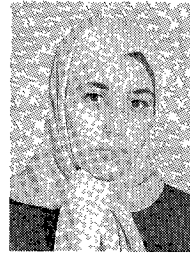
and distributed AI. Dr. Far is a member of the Association for Computing Machinery, IEEE Computer society, Japanese Society for Artificial Intelligence, and Information Processing Society of Japan.



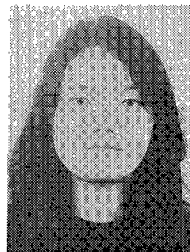
Sidi O. Soueina received B.Sc. and Msc. in 1993 and 1997 from Almustansiriyah University - Iraq, and Saitama University - Japan, respectively. He is currently a Ph.D. candidate at the Department of Information and Computer Sciences, Saitama University. His research interests are Distributed Artificial Systems, Machine Learning and Reasoning.



Hassan Hajji received BSc. in Applied Mathematics in 1995, from Mohamed Premier University - Oujda, Morocco. He is currently a Msc. student at the Department of Information and Computer Sciences, Saitama University - Japan. His research interests are in inference and applications of Bayesian Networks.



Shadan Saniepour received BSc. in Electronics in 1995, from Science and Technology University - Iran. She is currently a Msc. student at the Department of Information and Computer Sciences, Saitama University - Japan. Her research interests are in Case based Reasoning and Distributed AI.



Anete Hiromi Hashimoto received BSc. in Computer Science in 1997 from University of Para - Brazil. She is currently a Msc. student at the Department of Information and Computer Sciences, Saitama University - Japan. Her research interest is in Intelligent Tutoring Systems.