An application of multi-agent technology to electronic commerce - SAGE:Francis -

Tamami Sugasaka, Ryusuke Masuoka, Akira Sato,
Hironobu Kitajima and Fumihiro Maruyama
Fujitsu Laboratories Ltd.
2-2-1 Momochihama, Sawara-ku, Fukuoka 814, Japan
{tamami, masuoka, satoyan, kitajima, superb}@flab.fujitsu.co.jp

Abstract

We are making researches into seamless integration of information distributed over networks under the SAGE (Smart AGent Environment) project. In the SAGE, conversational agents speak in ACL (Agent Communication Language) and cooperate in solving problems focused on retrieval and integration of heterogeneous information. The SAGE consists of user agents which agentify users, database agents which agentify legacy software or database management systems (DBMSs) and mediator agents, called facilitators, which help the interoperation between other agents.

To put the SAGE into practical uses, we apply the SAGE to EC (Electronic Commerce). We have already constructed its prototype, called SAGE:Francis, which provides search services for information on commodities. As the result of an experiment with two legacy database applications, Oracle and Access, whose field names and category structures are different, the SAGE:Francis succeeded in integrating information on commodities and the response time was on average 3 seconds within real-world requirements.

We report, in this paper, the SAGE:Francis with emphases on services, architecture, facilitation, and user interface.

1 Introduction

In recent years, as an astounding growth of World Wide Web, an information-oriented society has been progressing rapidly. In the years to come, the amount of information stocked on the Internet will continue to increase and it will be natural to get information through the Internet. However it is still a very hard endeavor to find what we really want from a vast amount of information.

One way to deal with the situation is to use search engines which give us the locations of our target information found by related keywords or guide us through the categorized path to the locations. However using search engines is not easy for two reasons.

One reason is the difficulty of flushing upon adequate keywords. This goes for everyone, not only who is unaccustomed to using search engines but also who is experienced in searching on the Internet, because of differences in vocabularies and concepts between searchers and providers.

Another reason is the dependence on an experience or a flair to find our way to the location of our target information through the categorized path. If we could not find out the location along one path, we might be able to find out it along another path. Only with an experience or a flair, a good path is selected.

We are making researches into seamless integration of heterogeneous information distributed over networks under the SAGE (Smart AGent Environment) project. This project utilizes software agent technology, especially of those conversational agents which communicate by ACL (Agent Communication Language) [1]. ACL consists of KQML (Knowledge Query and Manipulation Language) [5] and KIF (Knoqledge Interchange Format) [4]. Main research areas for SAGE project include:

- Agentification of users and legacy applications
- Facilitation of interoperation between agents by facilitators, a kind of mediator agents
- Message formats and transactions
- Libraries and tools for the above
- Real-world applications

To put the SAGE into practical uses, we are trying to apply the SAGE to EC (Electronic Commerce). EC is growing remarkably as one of the most important applications on the Internet. EC is divided into inter-company EC and consumer EC.

In most cases of inter-company EC, a centralized server built on the Extranet offers the total support for business processes, such as search for commodities, price negotiation, procurement and payment. In consumer EC, electronic shopping malls built on the Internet introduce commodities to customers, help procedures of commodity procurement and so on. When the whole of EC becomes active in the future, many useful servers will be built in inter-company EC and the shopping malls become larger in consumer EC. Besides inter-company EC and consumer EC will not have so clearly defined boundary. Therefore to integrate such commodity information, it is necessary to combine distributed servers and legacy software and offer easy, simple and useful interfaces.

We believe SAGE can provide solutions for such situation by the following three merits.

The first merit is an integration of disperse data sources and reuse of legacy information sources. Especially with translation service provided by facilitator, it should be much easier to make data sources of different origins interoperate.

The second merit is a suggestion of adequate keywords. The SAGE is customized for each domain bounded reasonably, so that it is able to set limits to keywords, indicate a combination of adequate keywords, and candidate for their value.

The third merit is a flexibility of a change in information sources. The SAGE provides an advertise/unadvertise service, which is the provider sides to inform facilitators of their capabilities, so the SAGE can dynamically facilitate of interoperation between agents.

In this paper, we describes services and architecture of SAGE:Francis in the section 2, functions and architecture of facilitators in the section 3, user interface and user agents in the section 4, and the results of the experiments in the section 5.

2 Services and Architecture

In the first phase of the application to EC, the SAGE:Francis provides users with services concerning search information on commodities.

One service is the input interface which is easy to use and simple and make users input relevant conditions. In addition users need not to care about the agent technology which is used behind the scene.

Another one is the output interface which is easy for users to compare found commodities by listing them in a table and displaying details of each item in the same format.

Figure 1 shows the arrangement of agents in the SAGE:Francis. We have reported details of its configuration and implementation in [12].

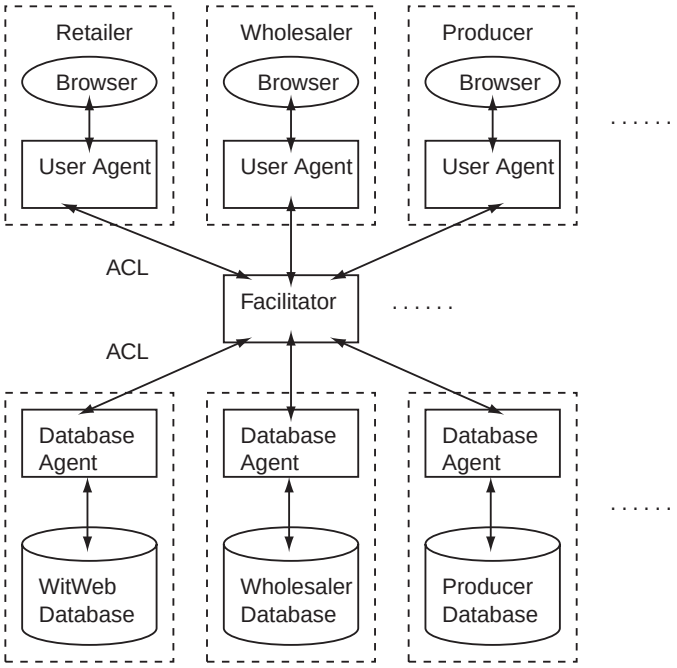

Figure 1: Architecture of SAGE:Francis: The SAGE:Francis has a 3-tier structure where facilitators are placed between user agents and database angents. User agents represent users as agents to SAGE and database agents represent databases as agents to SAGE. Facilitators stand between them to provide brokerage, translation, merge and other services to help those agents interoperate.

When a database agent starts, it advertises categories, message formats and ontology it can handle to the facilitator. The user agent turns a query form user into an ACL message and sends it to the facilitator. The facilitator chooses the appropriate database agents based on advertise/unadvertise messages from the database agent and translates it for each database agent if necessary. [1] Then the facilitator sends out the messages to the chosen database agents and wait for the reply messages. The database agent changes the ACL message from the facilitator into a SQL query and consult the database. The result from the database is composed into an ACL message and the ACL message is sent

[1] We allow database agents to have different ontologies for KIF terms in KIF content of the messages.

back to the facilitator. The facilitator merges the messages from the database agents into one and sends back to the user agent. The user agent displays the result.

3 Facilitators

3.1 Functions

We identified the useful functions of facilitators and implemented them. Facilitators in our agent system play a very important role brokering messages with those useful functions. Such functions of facilitators include the followings:

- Delegation based on:
  - (Un) Advertise messages
    - Categories
    - Agent capabilities
  - User Authentication
- Ontology translation
- Message handling
  - Waiting for asynchronous messages
  - Merging multiple messages
  - Sorting the contents of messages

Facilitators delegate messages to database agents based on advertise messages from database agents. In those messages, database agents can advertise categories of their information and capabilities such as message patterns which they can handle.

Since we allow agents have different ontologies for the terms in KIF contents of the messages, facilitators also provide ontology translation services. The knowledge needed for translation is stored in the Knowledge Bases of facilitators. (see Chapter 3.2) But the translations are done by functions not by the inference engines because of consideration of speed.

Waiting for asynchronous messages is actually a necessity for any agent in our agent system.

Merging the messages can be useful for agents which are not designed to wait and handle several messages together. In our system, facilitators provide this service so that all transactions a user agent has to handle is one-to-one single message transaction with the facilitator.

Sorting of merged contents of multiple messages is sometimes a requirement for the system. Facilitators can provide such a service.

These functions are realized through the use of the inference engine (c.f. 3.2) in the facilitator during the processing of the messages.

3.2 Architecture

Our facilitator is entirely written in Allegro Common LISP. The architecture of facilitator and how its components work are described in figure 2. A facilitator is a multi-thread application to deal with multiple messages simultaneously.

We will give more details of the inference engine and its associated knowledge base used in tha facilitator. These two components contribute to realize dynamic integration of heterogeneous information sources through management of ontologies and their translation information and contents of advertise/unadvertise messages.

The Knowledge Base of the facilitator keeps the following knowledge:

- Ontologies
- Translation knowledge (between ontologies)
- Advertisement
- Advertisement meta-data

Different ontologies for agents and translation knowledge between them are loaded from files at the startup of the facilitator and kept in knowledge base. Advertisement and its meta-data are inserted and deleted as the facilitator receives advertise messages and unadvertise messages from database agents.

Using the knowledge in the knwoledge base, the inference engine is used for the support of realization functions of facilitators.

The inference engine used in our facilitator is written in Allegro Common LISP as other parts of the facilitator. The codes for the inference engine is based on Frolic (c.f. [11]). Frolic has basic inference capability and support for LISP expression evaluation.

Then we modified and extended the codes to accomodate the functions of the facilitator. The modifications and the extensions include the followings:

- Multi-module capability for efficiency and categorization of the knowledge
- Multi-thread safeness
- Handling of KIF expression
- Retraction of rules for realization of processing unadvertise messages
- Switchable cache of inferred results for the efficiency

The processors in the processor pool of the facilitator mainly use the inference engine.

The advertise-processor of the facilitator uses the inference engine to assert the contents and meta-data of the advertise messages. The unadvertise-processor uses the inference engine to retract the contents and meta-data of the advertise messages. The ask-processor consults the inference engine to which database agents to delegate message to, retrieve ontologies and translation information.

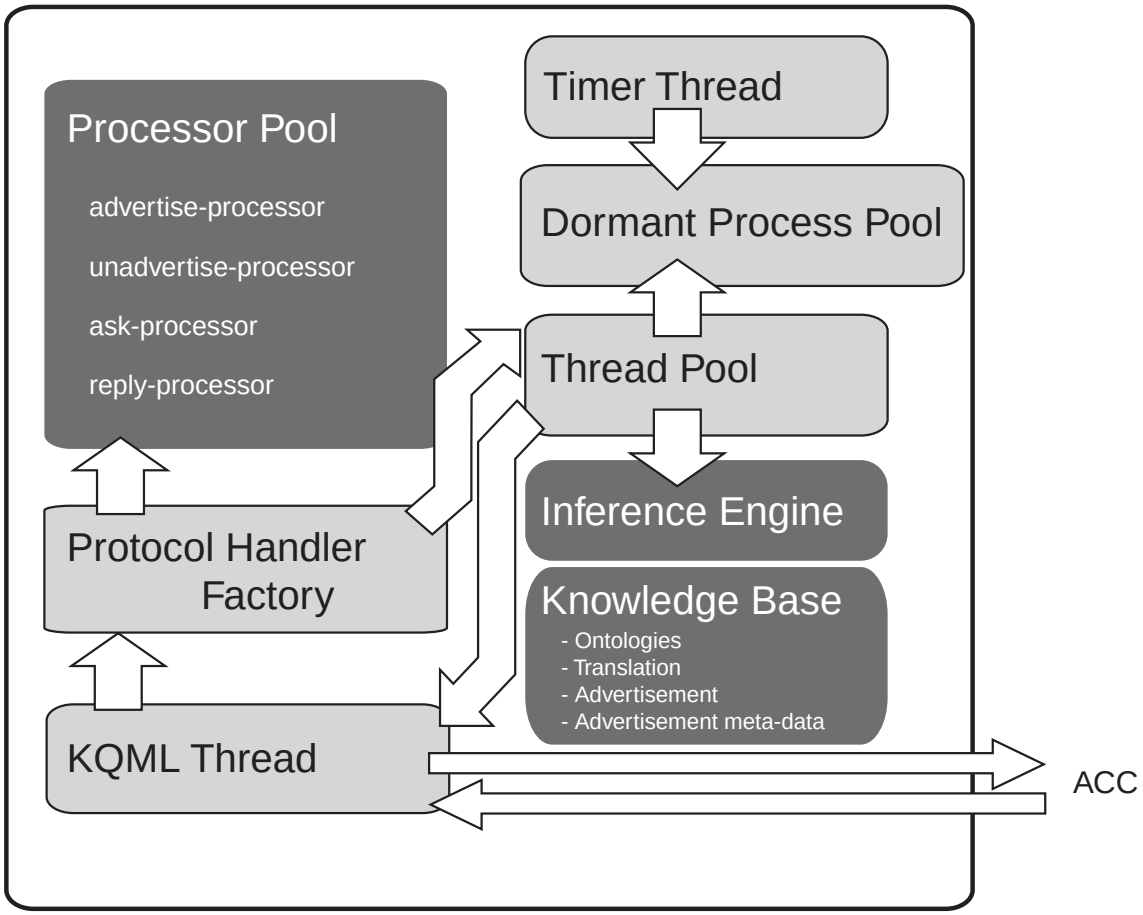

Figure 2: Architecture of facilitator: A facilitator receives and sends messages from and to other agents through the KQML threads and the ACC (Agent Communication Channel). A thread (called a protocol handler) to handle a received messages is created by the Protocol Handler Factory using processes in the Processor Pool. The thread is executed in the Thread Pool to process the received message, using the knowledge stored in the Knowledge Base through the Inference Engine. When the thread needs to be persistent (for example, to wait for the replies), the thread is kept in the Dormant Process Pool. Such threads can be restarted and executed by the events of corresponding messages or the events by created by the Timer Thread.

4 User Agents and Interfaces

Figure 3 shows the architecture of user agent.

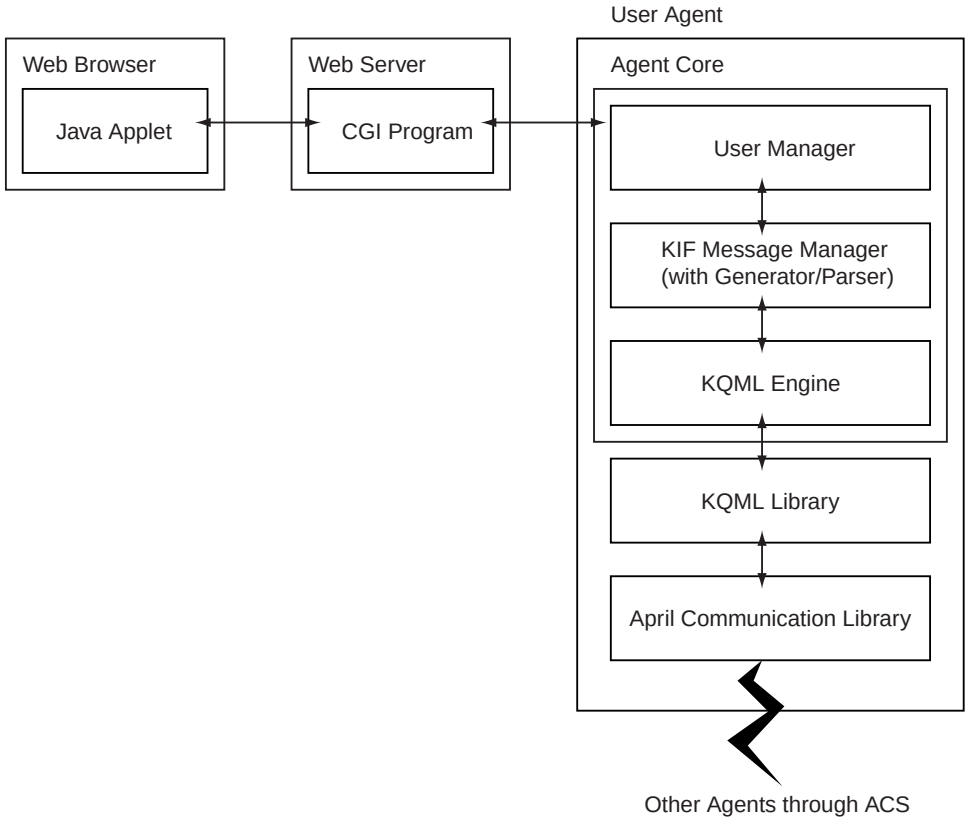

Figure 3: Architecture of user agent

A Java applet is downloaded into a browser and provides an user interface. This applet sends the HTTP/POST message to the CGI program at a web server, which in turn relays the message to the agent process to the applet. In the user agent process, there are several internal components. They are user management, KIF message manager (including KIF generator/parser), KQML engine, KQML library and April communication library. The user agent process can handle messages from multiple applets simultaneously using threads.

In order to make user agents general-purpose and domain-independent, a Java applet and KIF generator/parser are made to be configured with a configuration file. We provide useful resources for the configuration file. These resources make developers of the SAGE customize easily.

Resources for ACL messages: are utilized, when user agents turn an user's query through a Java applet into an ACL message in the KIF generator and relay an ACL message from facilitator to the Java applet in the KIF parser. These resources are **title** for title name, **heading** for heading name, **heading.width** for a width of heading, **input.cols** for the number of column of input box, **list.rows** for the number of value displayed in a box, **popup.caption** for caption of popup window, **popup.rows** for the

number of candidates displayed in popup window and **popup.cols** for the number of column of popup window.

Resource for a layout: are utilized in a browser. These resources are **title** for title name, **heading** for heading name, **heading.width** for a width of heading, **input.cols** for the number of column of input box, **list.rows** for the number of value displayed in a box, **popup.caption** for caption of popup window, **popup.rows** for the number of candidates displayed in popup window and **popup.cols** for the number of column of popup window.

5 Results

We have set up an agent system for EC and carried out proof-of-concept experiments. For the experiments, we wanted to use real data from real databases in use, but unfortunately it was not available for several reasons. Therefore we produced two databases, "Fukuoka Market" and "Kawasaki Market." They use different terminologies for field names and category structure for their commodities.

In the experiment, SAGE:Francis consists of one Web server, one user agent, one facilitator and two database agents. Being set up this way, search requirements were put into a Java applet in a browser and a query was made. (see figure 4). Then the result from the two databases was returned as a list (see figure 5) and the details of each item were displayed by clicking the button in the list. (see figure 6). The system worked out successfully.

Figure 4: User Interface (1): Users enter search requirements into a Java applet.

High performance of the system is a very important requirement, as we are going to apply the system to real-world problems. Even a system with useful functions is worthless if its performance is poor.

With the same setup, performances are measured for the response time, which is the time between when the user clicks the "O.K." button to send a query message and when the Java applet starts to display the result list. The response time averaged for 25 trials was 3 seconds. This performance is within real-world requirements.

Figure 5: User Interface (2): The result of the search is returned as a list.

Figure 6: User Interface (3): The details of each item is displayed by clicking the button in the list.

6 Conclusion

We have reported SAGE:Francis, which is an application of multi-agent technology to electronic commerce, with emphases on services, facilitators and user interface. The system is actually built and proof-of-concept experiments have been carried out. Performance measurement showed us that the system is within real-world requirements.

Other application areas in which we are currently working on, include knowledge management in enterprises and integration of online databases. These applications utilize the same facilitator as SAGE:Francis.

The SAGE with facilitator technology enables users to search heterogeneous information on commodities distributed over networks through one interface. Besides, the SAGE is a flexibility of a change in information sources with advertise/unadvertise services, so the SAGE enables users to get immediately the latest information after its enter.

In the future, we are trying to apply the SAGE to many domains. Accordingly, our work includes realizing faster implementations of agents, implementing other useful functions of facilitators, distributed implementation of facilitators, learning preference of user and so on.

References

[1] M.R.Genesereth and S.P.Ketchpel: "Software Agents," Comm.ACM Vol.37 No.7, 1994.

[2] "Conversational Agent," IEEE Internet Computing Vol.1 No.2 pp.73-75.

[3] Arthur M. Keller and Michael R. Genesereth, "Multivendor Catalogs: Smart Catalogs and Virtual Catalogs," in EDI Forum, The Journal of Electronic Commerce, Vol. 9, No. 3, September 1996.

[4] M.R.Genesereth and R.E.Fikes, "Knowledge Interchange Format Version 3.0 Reference Manual," Technical Report Logic-92-1, Computer Science Department, Stanford Univ., 1992/6, URL: http://logic.stanford.edu/papers/kif.ps

[5] The DARPA Knowledge Sharing Initiative External Interfaces Working Group, "Specification of the KQML Agent Communication Language," 1994/2/9, URL: http://logic.stanford.edu/papers/kqml.ps

[6] McCabe, F. G. and Clark, K. L.: "April - Agent PRocess Interaction Language," Lecture Notes in Artificial Intelligence 890, pp. 324-340, Springer-Verlag, 1995

[7] URL: http://infomaster.stanford.edu/

[8] URL: http://www.fujitsu.co.jp/hypertext/Publish/journal/237e/e37tk1_7.html

[9] URL: http://www.hitachi.co.jp/Prod/comp/ec/twx21/index.html

[10] URL: http://www.tradeex.com/

[11] Jed Krohnfeldt and Craig Steury: "Frolic: Logic Programming with Frobs", Utah PASS Project OpNote 96-08, 1993, `ftp://ftp.cs.utah.edu/pub/frolic.tar.Z`

[12] R.Masuoka, T.Sugasaka, A.Sato, H.Kitajima and F.Maruyama: "SAGE and Its Application to Inter-company EC", Proceedings of PAAM98, pp.123 - 135.