FACL : A Form-based Agent Communication Language for Enduser-Initiative Agent-Based Application Development

Takeshi CHUSHO and Katsuya FUJIWARA
Department of Computer Science,
School of Science and Technology, Meiji University
Kawasaki, 214-8571, Japan
chusho@cs.meiji.ac.jp

Abstract

The number of endusers using the Internet increases on the inside and outside of offices. Enduser-initiative development of applications has become important for automation of their own tasks. As the solution based on the philosophy : "All routine work both at office and at home should be carried out by computers," this paper describes a multi-agent framework and an agent communication language(ACL) for the MOON(multiagent-oriented office network) systems which are distributed systems including window work in B-to-C and B-to-B electronic commerce. The multi-agent framework is a Java application framework and includes a form-based ACL(FACL) as a common protocol for passing application forms and the three kinds of agents working at client terminals, server-at-windows and the MOON servers respectively. FACL has very simple message structure of (who, what, how, which) because FACL was designed based on the simple concept that "one service = one form." FACL is used primarily for window task sending or receiving written forms between a client agent and a domain expert agent. In addition, broker agents and mobile agents of MOON servers participate in these communications for directory services and form delivery services. FACL brings high interoperability among distributed application systems based on agent technologies, and promotes that endusers themselves develop their agents by teaching agents what to do.
Key words : multi-agent, agent communication language, application framework, object-oriented technology, electronic commerce

1. Introduction

The number of endusers using the Internet increases on the inside and outside of offices. This trend promotes the following our philosophy: "All routine work both at office and at home should be carried out by computers." Our policy for this purpose is enduser-initiative development of distributed information systems by implementing agents of the endusers, by the endusers, for the endusers because package software may not substitute for various work of various endusers [3, 4].

As the solution based on agent technologies, this paper describes a form-based agent communication language(FACL) for enduser computing under distributed systems. As a typical distributed information system, we direct our attention to an application system for windows or counters in banks, city offices, travel agents, mail-order companies, etc. Some kind of window work such as mail-order business has already been put to practical use in current computer networks including the Internet as online shopping. However, both friendliness of enduser interfaces for clients and automation of routine work for domain experts are still insufficient. In addition, the work to be automated may be limited to particular ones such as electronic commerce related to B-to-B and B-to-C, which make a profit over the development cost.

In the near future, the information society will require such new technologies that domain experts can automate their own work by themselves and that almost all clients can operate computers at home or at office without extra training or without the help of others. Furthermore, a com-

mon protocol between the windows and their clients must be developed for avoiding appearance of a number of incompatible interfaces corresponding to the explosive number of combination of the windows and their clients.

Multi-agent systems must be the solution for these problems because endusers may teach their operations to agents without programming and because these agents may cooperate each other by using a common agent communication language. The multi-agent systems will bridge a semantic gap between applications and endusers while application frameworks [6] and patterns [10] bridge a granularity gap between components and applications as shown in Figure 1.

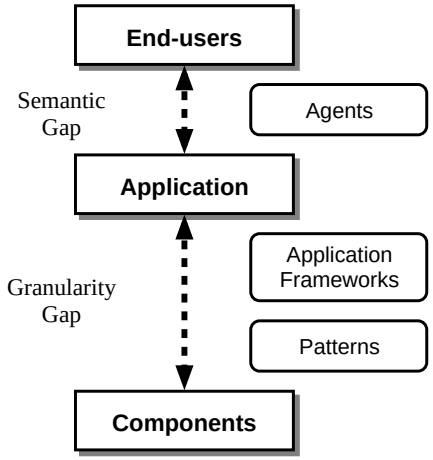

Figure 1. Technologies for bridging a semantic gap and a granularity gap between endusers and components.

Agent technologies came out in 80's from two research areas of artificial intelligence and software technologies. A variety of agents have already proliferated without a consensus of opinion on the meaning of the term [1]. The application area is a wide range from email support [15, 17] to negotiations in electronic commerce [16]. Recently it includes mobile agents. For example, a tool for nonprogrammers to build a mobile agent by using visual modeling and a small set of generic icons [5].

In particular, multi-agent systems are important for advanced applications based on distributed systems and the Internet such as electronic commerce support systems. An agent communication language(ACL) is one of the key technologies for interactions among independently-developed applications with agents [11, 20]. Roughly speaking, in the history of ACLs, the Knowledge Query and Manipulation Language(KQML) was developed in late 80's by the ARPA Knowledge Sharing Effort [7] and many ACLs followed it. Then the standardization is being tried by FIPA(Foundation for Intelligent Physical Agents) and OMG(Object Management Group) Agent WG. FIPA has proposed the draft of FIPA ACL [8] which provided more than twenty Communicative Acts(CAs) for communication among independently developed agents. OMG Agent WG [18] intends to recommend common agent technology representing reusable, interoperable, portable application components which enable developers to better understand how to develop applications using agent technology.

In this paper, a customizable multi-agent system is considered as a kind of application framework including domain specific patterns, where agents and their ACL are considered as business objects and a communication protocol for cooperation among the objects respectively. This paper proposes the form-based ACL(FACL) which will be used for applications with window task sending or receiving written forms and which has very simple message structure of (who, what, how, which).

The objective is that endusers themselves develop their agents by teaching agents what to do. The first version of a multi-agent framework, the wwHww system, was designed as a distributed cooperative system based on a message-driven model of object-oriented technology and was implemented as a Java-base application framework. An application framework implies a reusable semi-complete application or a skeleton of an application that can be specialized to produce custom application, and that is represented by both a set of abstract classes and the way their instances interact [6, 14]. In the wwHww system, use of the application framework including FACL brings high interoperability among distributed application systems, that is, among agents in the applications. The customization of the hot spots in the application framework implies the agent development by endusers.

This paper presents the requirements for the agent-based application in Section 2, the design of FACL in Section 3, the multi-agent framework in Section 4, and finally discussions about ACL, design patterns and heterogeneity.

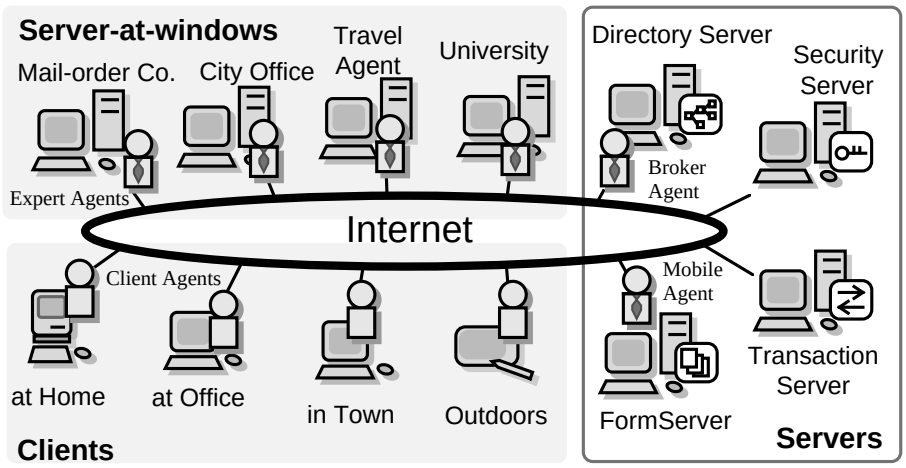

Figure 2. A MOON(multiagent-oriented office network) system.

2. Requirements for Agent-based Application

2.1. An Example of Application

Recently, business process re-engineering [13] has attracted notice since information technologies may drastically improve efficiency and effectiveness of company activities. Now it extends to virtual companies and electronic commerce. Furthermore, information technologies represented by the words "Internet" and "multimedia," will give the power of process re-engineering of human society as well as business fields.

Let's suppose that you move from some city to any city. How many windows in various organizations do you contact for an address change? How many application forms do you fill in? How many questions do you ask to domain experts? If these processes can be executed by agents in our computer at home or at office, a lot of time will be saved.

As for domain experts at windows, they are asked about the same question of how to fill in the form again and again every day and repeat the same explanation and the same check of written applications. If these routine work can be processed by agents, a lot of cost will be reduced.

Of course many kinds of window work have already been put to practical use in the Internet and intranets. However, these systems must have been developed by IT professionals, not by endusers and are expensive. Furthermore, although the domain experts require frequent modification of specifications in these systems for service upgrade, it is difficult to modify software timely because the domain experts do not maintain the systems by themselves and need to ask the IT professionals instead. Our goal is development and maintenance of agent-based applications by endusers.

As a typical distributed information system, we direct our attention to application systems for window work. Such window work is not limited to the actual window work in the real world. For example, in a supply chain management system, exchanges of data among related applications can be considered as the virtual window work. Therefore, in electronic commerce, the window work is indispensable.

2.2. Application architecture

Agent-based applications are constructed on a multiagent-oriented office network (MOON) for window work. The MOON system is based on a client/server model and is partitioned into the following three parts as shown in Figure 2:

1. Client terminals with client agents for sending written applications to windows, such as personal computers and workstations both at home and at office, public telephones with terminals in town, portable computers for mobile computing outdoors, etc.

2. Server-at-windows with expert agents for receiving written applications, such as windows in mail-order companies, city offices, travel agents, universities, etc.

3. MOON servers for managing the network system.

The MOON servers imply the following four servers and some of them may be located physically in server-at-windows:

1. A directory server with a broker agent manages network addresses of server-at-windows to receive written applications as service directories of windows.
2. A form server with a mobile agent manages various application forms for services at these windows, which forms are defined with help messages and selection menus by domain experts.
3. A transaction server stores written applications received by server-at-windows with the identification numbers, manages the states of the process and replies to inquiries about the states. It may be connected with a workflow system in the organization including the server-at-window.
4. A security server controls access rights to server-at-windows and the MOON servers, and manages authentication of clients.

2.3. Features of agent-based applications

2.3.1. Automatic form processing

The first feature of the MOON system is electronic form processing which is navigated by agents both in client terminals and in server-at-windows. The following requirements are essential for enduser-initiative agent-based application development by using the application framework of the MOON system:

1. Clients can teach the fixed operations of filling in a form about such plain words as their names, addresses and phone numbers to their agents. Then their agents do so instead.
2. Domain experts can teach their expertise to their agents. Then the agents guide clients in filling in the form and check the written form.

These facilities bring freedom from routines to both clients and domain experts. We named these agents "Intelligent Clones" since they perform the routine work with adaptation and learning facilities as if their owners would do so [2].

2.3.2. A common ACL

The second feature of the MOON system is standardization of ACL for communication between client agents and expert agents. Design of ACL depends on features of multi-agent systems. Roughly speaking, multi-agent systems are classified into two types. That is, agents are cooperative or competitive each other. This paper describes cooperative multi-agent systems because clients and domain experts are cooperative in the most cases of the form-based application domain.

Furthermore, these cooperative multi-agent systems are classified into two subtypes. That is, agents are homogeneous or heterogeneous each other. In the homogeneous environment, every agent has the same internal architecture. In many cases, the agent has a shared goal, and solves the subgoal cooperatively.

In the heterogeneous environment, every agent is developed independently and has an individual goals. These agents cooperate each other for constructing the compliant society. This paper describes this type because each of client agents and expert agents can perform each task alone in the form-based application domain.

3. Design of FACL

3.1. Design policies

The form-based ACL(FACL) was designed for communication between client agents and expert agents. Because the communication is performed by sending or receiving written forms, FACL is different from FIPA ACL and is very simple as follows:

1. FACL does not limit the services of each agent although FIPA ACL limits them to a set of the given Communicative Acts(CAs) or embeds them into content of each message. The services in FACL are provided as forms by the enduser individually.
2. The concept of FACL is that one service implies one form which is registered to the directory server although FIPA ACL gives examples that client agents

inquire about services of a particular agent or an unknown agent by using the CA of “query-ref” or “propagate” respectively. This difference affects to a mechanism of directory services.

3.2. The basic form of FACL and the semantics

Messages of requests to windows in FACL include the following elements:

- Who receives your request?
- What do you request to the window?
- How do you request it?

The name of the multi-agent framework, wwHww, is derived from ‘who-what-how with WWW’ and is pronounced as ‘who’ for convenience. The following element is added to these three elements.

- Which is your request?

That is, the basic form of FACL is shown as follows:

(who, what, how, which)

The sender’s name is usually included in the how-parameter at the form-based application level although a sender name is included as one of message parameters in FIPA ACL.

The semantics of FACL is based on a message passing concept of conventional object-oriented programming languages. The four parameters correspond to elements of a message between objects respectively as follows:

who : A message receiver object

what : A method name

how : Parameters for a method invocation

which : A message number

In the MOON system, the who-parameter implies a window where a written application is sent to. The what-parameter implies the title of the application form. The how-parameter implies contents of the application form. The which-parameter implies a receipt number stamped on the received written application.

As KQML can be thought of as consisting of three layers: the content layer, the message layer and the communication layer [7], FACL can be done so. The who-parameter is positioned on the communication layer as the identity of the message receiver. The what-parameter is positioned on the message layer as determinant of the kinds of interactions since this parameter corresponds to the performative name in KQML. The how-parameter is positioned on the content layer as the actual content of the message. The which-parameter may be positioned on the communication layer if a receipt number is considered to be attached to a message as a unique identifier associated with the communication such as the ‘:reply-with’ keyword of KQML. In FACL, however, the which-parameter is considered to be positioned on the message layer because the receipt number is stamped on the received written application and is referred to later by messages for inquiries about the received application.

The states of values of these parameters in FACL affect semantics of the message. If a value of a message parameter is unknown, the message implies an inquiry about the parameter. This semantics is quite different from conventional object-oriented programming languages because it is illegal not to determine the message receiver, the method name and the actual parameters for conventional message sending. This extension in this paper, however, produced attractive effect. Examples are given in the next subsection.

3.3. Enduser interface

The actual enduser interface for filling in the form is different from the basic form which is the internal representation in the system. Examples of requests by using FACL are given in the basic form for convenience, where the following notations are used:

a, b, ... : Parameters with known values.

?a, ?b, ... : Inquiries about the parameters with known values, which request help messages.

x, y, ... : Parameters without values.

?x, ?y, ... : Inquiries about the parameters themselves, which request all possibles for selection.

In the basic form of FACL, a word in which the first character is ’?’, is used for two different kinds of inquiries although it means a variable in FIPA ACL and

KIF(Knowledge Interchange Format) [9]. The second use such as ?x and ?y for requesting all possibles in FACL is similar to the use of a variable in Prolog.

1. Examples of sending written applications:

   (a) (a, b, c, x)

   The written application, b, with the contents, c, is sent to the window, a. A message number will be assigned to the variable, x, by the window receiving this message.

   (b) (a, b, , ?d)

   The state in the process of the written application, b, of the message number, d, is inquired of the window, a.

   (c) (a, b, , -d)

   The written application, b, of the message number, d, which was already sent to the window, a, is canceled. That is, the first character, '-', of the which-parameter with a known value, implies the cancellation.

   In FIPA ACL, the CAs of "request" or "inform" may be used for (a). That is, "request" is used for requesting the receiver to perform some action and "inform" is used for informing some proposition. For (b), the id which the receiver defined as the parameter value of ":reply-with" in the previous message, may be used as the parameter value of ":in-reply-to" in the sender's message. For (c), the CA of "cancel" is used. In comparison with FIPA ACL, FACL provides the simpler and easier-to-understand message expressions because the what-parameter specifies the service name explicitly.

2. Examples of inquiries about application forms:

   (a) (a, b, ?x,)

   The application form, b, to be sent to the window, a, is displayed. How to fill in the form is navigated by the expert agent. Some typical items are filled automatically by the client agent.

   (b) (a, ?x, ,)

   The title list of all application forms which the window, a, receives, is displayed.

   (c) (?x, ?y = (a list of keywords), ,)

   The list of titles of all application forms which relate to the list of keywords, is displayed with the names of windows receiving them. The system retrieves forms whose titles include the keywords or in which help messages include the keywords.

   (d) (?x, ?y, ,)

   All windows and all titles of application forms which are received by those windows, are listed.

   (e) (?a, , ,)

   The explanation on the work of the window, a, is displayed.

   (f) (a, ?b, ,)

   The explanation on the application form, b, to be sent to the window, a, is displayed.

   These inquiries must be simpler than messages in FIPA ACL in which "request" for (a), (e) and (f), "query-ref" for (b) and "propagate" for (c) and (d) may be used because FACL is based on the form concept. In addition, qualified names for the who-parameter may be used such as a1.a2 and a1.?x2 in a hierarchy of an organization.

3.4. An example of operations

Let's consider a citizen who wants to get permission for parking at the city hall and suppose that he or she does not know where and how it can be gotten. Figure 3 shows operations to be done by using a personal computer as a client terminal of the MOON system at home. The operations and the basic form of FACL to be sent at each step are described as follows:

(a) "City.*" is input to the who-column and the keywords of "parking" and "Hall" are input to the what-column in the initial screen, and then the basic form of (City.?x, ?y = ("parking" "Hall"), ,) is sent. That is, the character, '*', implies a wild card and is transformed into '?x' of the basic form.

(b) The system displays "CityHall Sec." in the who-column and "Parking form" in the what-column. By clicking the value area in the what-column, the basic

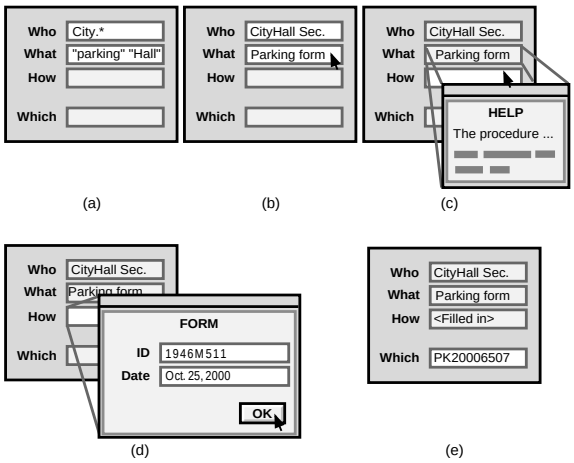

Figure 3. An example of operations of the MOON system at home.

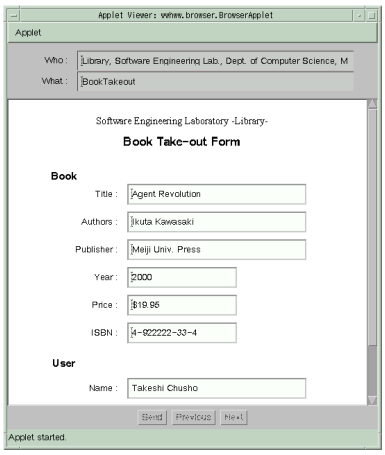

Figure 4. An example of the wwHww browser at a client terminal.

form of (“CityHall Sec.”, ? “Parking form”, ,) is sent. The first character, ’?’, of the what-parameter with a known value implies the help message request.

(c) The system displays the help message on the parking form. After clicking the value area of a blank in the how-column, the basic form of (“CityHall Sec.”, “Parking form”, ?z,) is sent.

(d) The system displays the application form. After filling in the two blanks for the ID no. and the date, the basic form of (“CityHall Sec.”, “Parking form”, (“ID no.” = “1946M511”, “date” = “Oct. 25, 2000”), w) is sent. In equations of the how-parameter, the left sides are column names in the form and the right sides are input values of the columns.

(e) The system displays the message number in the which-column, while the value is assigned to the variable, w. Then the system terminates by clicking the close button.

4. Multi-agent Framework

4.1. Software architecture

The first version of the multi-agent framework, wwHww, has been developed with a library system, which is used in our laboratory. Such a form-base system is helpful to us since there are no librarians in our laboratory, For example, we can know who borrowed some book because everyone fills in an electronic application form when taking out a book from our laboratory. We can know whether some book has been already registered or not because everyone fills in an electronic application form after he or she bought the book for our laboratory.

An example of the wwHww browser for taking out books is shown in Figure 4. The head part indicates the name of the server-at-window in the who-parameter and the name of the service in the what-parameter. The white part implies the how-parameter, that is, the electronic application form itself requested.

The software architecture is shown in Figure 5. The wwHww browser of the client side is composed of two subsystems, that is, the form browser and the directory browser. The wwHww server of the server side is composed of three subsystems, that is, the directory server, the form server and the transaction server. This system was implemented in Java and there are Java applet versions and Java application versions for two browsers. The wwHww protocol for messages in FACL was implemented by Java RMI. Consequently the application framework is constructed in three layers with 19 classes in Java. The upper layer is composed of the form browser of 6 classes, the directory browser of 1 class, the directory server of 1 class, the form server of 1 class and the

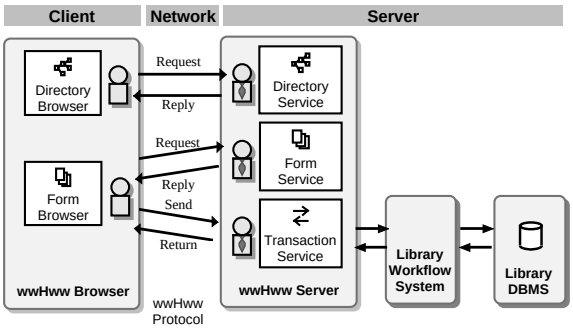

Figure 5. An application and the multi-agent framework.

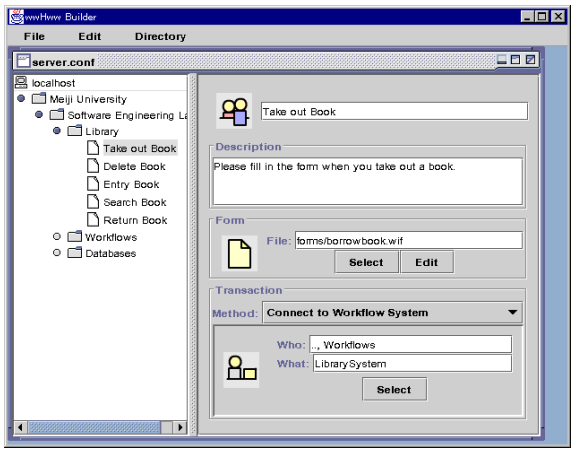

Figure 6. An example of the browser for system definitions by domain experts.

transaction server of 1 class. The middle layer is composed of the core of wwHww browser of 4 classes and the core of wwHww server of 2 classes. The lower layer is composed of the wwHww protocol of 3 classes.

4.2. An application building procedure

Domain experts build expert agents by using the framework as follows:

1. Service definitions : Services at the window, are defined.
2. Form definitions : Electronic forms for these services are defined while embedding navigation information into these forms.
3. Transaction processing definition : How to process written forms is selected among three typical processing methods of printing out, storing in a database or passing to a workflow system.
4. Registration : These definitions are registered into the corresponding servers.

An example of a browser for the library system definitions is shown in Figure 6. The left-hand part implies a hierarchical directory. The right-hand part implies definitions about the service for taking out books.

4.3. The four kinds of agents

Basically domain experts build expert agents by form definitions while teaching their expertise as mentioned above. These expert agents are mobile agents also since the form is sent to a client terminal from the form server and return to the transaction server. On the other hand, the directory server is a broker agent since a client agent asks about suitable expert agents.

The client agent is independent of the expert agent, the mobile agent and the broker agent. The client agent has facilities for automatically filling in the form. The knowledge is classified into two categories. One is knowledge on the owner itself such as a name, an address, a phone number and a birth day, which is independent of each form. The other is knowledge on each form such as a member number and a grade of membership, which is dependent on the server-at-window of the form.

As for automatically filling in the form of the first category, some intelligence is required for different expressions of the same meaning, such as ”Phone” and ”TEL.” This problem was solved by introducing concept names such as @NAME, @ADDRESS, @PHONE and @BIRTHDAY. The label name of an item in a form is mapped into the concept name by using mapping rules. Then the concept name is mapped into the individual value by using mapping rules also. For example, both ”Phone” and ”TEL” are mapped into @PHONE. Then the actual phone number of the owner is filled in the form by the client agent.

As for the second category, some intelligence is required for the same expressions of the different meaning, such

as a ”member number” of IEEE or a ”member number” of ACM. When there are two mapping rules for the concept name of @MNO, selection depends on the context in the form including the label name of ”member number.” For this reason, the mapping rule has constraints which are taught by the owner.

Actually, these two kinds of problems may happen in both categories although the solutions are same. For example, @ADDRESS may have two values of a home address and an office address. A ”member No.” may be used in a form instead of a ”member number.”

5. Discussions

5.1. The number of primitives in ACL

FIPA ACL provides 22 CAs(communicative acts) and 15 pre-defined message parameters based on the speech act theory. These CAs are classified into five groups, that is, five for information passing, three for requesting information, four for negotiation, eight for action performing and two for error handling. It looks electronic commerce oriented although the goal of FIPA ACL is wide range application.

Generally, the first messages for triggering interactions are classified into two types, that is, whether a message requests the receiver's reply or not. For example, ”request” and ”inform” in FIPA ACL correspond to each type respectively. An example of FIPA ACL messages is given as follows:

```
(inform
    :sender       agent1
    :receiver     hp1-auction-server
    :content      (price (bid good02) 150)
    :in-reply-to  round-4
    :reply-with   bid04
    :language     sl
    :ontology     hp1-auction
)
```

The meaning of this message is actually understood by the combination of three parameters of “:content,” “:language” and “:ontology.” That is, the message content expression can be understood by specifying the description language of the expression and by specifying the domain in which the symbols in the expression are used.

In FACL, since the goal is enduser-initiative agent development, the other policy is taken as

one service = one form = one CA.

Because the primitive communication method is a form, the number of kinds of forms is not limited. It must be easy for endusers to understand these concepts.

5.2. ACL as design pattern

The framework of the wwHww system is constructed of Java classes. Messages of FACL imply messages among classes. Therefore this framework is considered as a design pattern for form-based applications. Generally the design patterns are descriptions of communicating objects and classes that are customized to solve a general design problem in a particular context [10].

In FACL, there are three abstract classes of a client agent, an expert agent and a broker agent. The wwHww protocol with the definite who-parameter and definite what-parameter, implies a message between a client agent and an expert agent. The wwHww protocol with the indefinite who-parameter or indefinite what-parameter, implies a message between a client agent and a broker agent.

The Agent Working Group of OMG describes software agents in the green paper [18] as “basically, software agents are design patterns for software,” although the agent-based pattern is not defined yet. The green paper indicates one of issues for modeling an agent as an object. That is, expressive limitations arise in practice if an agent is provided with a method for each message which the agent can accept, instead of providing the agent with a single method, “AcceptCommunicativeString,” which would permit the agent to accept arbitrary messages. The FACL provides one solution that domain experts can register arbitrary forms with the directory server.

5.3. Heterogeneous environment

In the heterogeneous environment, each agent may be developed independently. This type of a multi-agent system is widely applied in various domains such as electronic commerce. In the heterogeneous environment, the external interface of an agent for communication with other agents should be strictly separated from the internal architecture.

In FIPA ACL, although the heterogeneous environment is supposed, independently developed agents may not communicate each other in the case that a different language is specified for each CA. For this reason, FIPA is making the content language library [9].

FACL supports cooperative multi-agent systems on the heterogeneous environment for compliant society of independent agents. Because of the form-base concept, the separation of the internal architecture and the external interface is strict. However, automatically understanding meaning of messages among agents is more difficult. For this reason, it is necessary for the owners to teach their agents what to do once. That is, a domain expert teaches an expert agent how to navigate the client to fill in the form, and a client teaches the client agent what to fill in the form about typical items. This problem is reduced by using XML [12] for implementing an electronic form because the items in the form are given the particular meaning in advance. For example, it is useful to declare the concept names such as @NAME and @PHONE for automatically filling in forms as element types. The wwHww intelligent form was implemented by using XML and BML(Bean markup Language) which is an XML-based component configuration or wiring language customized for the JavaBeans component model [21].

6. Conclusions

The Form-based Agent Communication Language, FACL, and the multi-agent framework based on this language were developed. A MOON(multiagent-oriented office network) system is easily developed by using this framework by domain experts themselves. These benefits were ascertained by feasibility study.

Acknowledgment

The authors express their gratitude to Prof. Robert Kowalski and Dr. Francesca Toni of Imperial College of Science, Technology and Medicine for invaluable discussions about the concepts of agents and multi-agents [19]. They are also indebted to Shinnosuke Chinda and Kei Shimada for their invaluable technical assistance, who implemented the client agent for automatically filling in the form and the library system of our laboratory respectively.

References

[1] Bradshaw, J. M., "An Introduction to Software Agent," Software Agent, MIT Press, pp.3-46, 1997.

[2] Chusho,T., Software Crisis and programming paradigms, (in Japanese), Keigaku-Shuppan, 1992.

[3] Chusho,T., Kashiwagi,K. and Kasama, Y., "wwHww : An Application Framework for End-User Computing in Multi-organizational Office Network Systems," The 12th annual conf. of Japan Society for Software Science and Technology, pp.281-285, Sep. 1995.

[4] Fujiwara, K. and Chusho,T., "Development and Evaluation of An Application Framework of Window Work for End-users," Trans. Information Processing Society of Japan, Vol. 41, No. 4, pp.1202-1211, April 2000.

[5] Falchuk, B. and Karmouch, A., "Visual Modeling for Agent-Based Applications," IEEE Computer, Vol. 31, No. 12, pp.31-38, Dec. 1998.

[6] Fayad, M. and Schmidt, D. C. (Ed.), "Object-Oriented Application Frameworks," Commun. ACM, Vol. 39, No. 10, pp. 32-87, Oct. 1997.

[7] Finin, T., Labrou, Y. and Mayfield, J., "KQML as an Agent Communication Language," Software Agent, MIT Press, pp.291-316, 1997.

[8] FIPA, "Agent Communication Language," FIPA Spec 2-1999, Draft ver.0.1, Apr. 1999.

[9] FIPA, "Content Language library," FIPA Spec 18-1999, Draft ver.0.2, 1999.

[10] Gamma, G., Helm, R., Johnson, R. and Vlissides, J., Design Patterns, Addison-Wesley, 1995.

[11] Genesereth, M. R., "An Agent-Based Framework for Interoperability," Software Agent, MIT Press, pp.317-345, 1997.

[12] Glushko, R. J., Tenenbaum, J. M. and Meltzer, B., "An XML Framework for Agent-based E-commerce," Commun. ACM, Vol. 42, No. 3 March, pp.106-114, 1999.

[13] Hammer, M. and Champy, J., Reengineering the Corporation, Harper Collins, 1993.

[14] Johnson, R. E., "Frameworks = (Components + Patterns)," Commun. ACM, Vol.40, No.10, pp.39-42, 1997.

[15] Maes, P., "Agents That Reduce Work and Information Overload," Commun. ACM, vol.37, no.7, pp.30-40, Jul. 1994.

[16] Maes, P., Guttman, R. H. and Moukas, A. G., "Agents That Buy and Sell," Commun. ACM, vol.42, no.3, pp.81-91, Mar. 1999.

[17] Malone, T., Lai, K. and Fry, C., "Experiments with Oval : a Radically Tailorable Tool for Cooperative Work," CSCW92, pp.289-297, 1992.

[18] OMG Agent Working Group, "Agent Technology, Green Paper," OMG Document no. ec/99-12-02, Dec. 1999.

[19] Sadri, F. and Toni, F, "Computational Logic and Multi-Agent Systems : a Roadmap," Technical Report(Preliminary version), Dept. Computing, Imperial College, July 1999.

[20] Singh, Munindar P., "Agent Communication Languages: Rethinking the Principles," IEEE Computer Vol. 31, No. 12, PP.40-47, Dec. 1998.

[21] Weerawarana, S. and Duftler, M. J., "Bean Markup Language," http://www.alphaworks.ibm.com/tech/bml, 1999.