

# Implementation of FIPA Ontology Service

Hiroki Suguri\*  
suguri@comtec.co.jp

Eiichiro Kodama\*\*  
kodama@iwate-pu.ac.jp

Masatoshi Miyazaki\*\*  
miyazaki@iwate-pu.ac.jp

Hiroshi Nunokawa\*\*\*  
nunokawa@sfais.or.jp

Shoichi Noguchi\*\*\*  
noguchi@sfais.or.jp

## ABSTRACT

The authors have implemented FIPA 98 Spec 12: Ontology Service. In addition, we developed a sample application of on-line shopping service that integrated multiple database schemata using the ontology service to verify and demonstrate the specification and the implementation. In this paper, we briefly introduce FIPA and FIPA Specifications at first. Next, we describe Comtec Agent Platform, on which the ontology service is deployed. Then, after reviewing the ontology service specification and its relationship with OKBC, we discuss details of the implementation of the ontology service and demonstration application. We conclude with a prospect of real-world applications in the Agentcities project and future works that we are currently preparing for.

## Keywords

Practical experience, Standardisation issues, Implications on agent internal structure, Component architectures for agents, FIPA, Ontology Service, OKBC, Agentcities.

## 1. INTRODUCTION

FIPA (The Foundation for Intelligent Physical Agents, <http://www.fipa.org/>) is a non-profit organization aimed at producing standards for the interoperation of heterogeneous software agents. The purpose is realized by publishing a set of specifications so that the agents developed by different vendors can talk to each other based on the standard specifications.

In 1997, FIPA introduced its first set of specifications called FIPA 97. FIPA 97 specifications include Spec 1: Agent Management, Spec 2: Agent Communication Language, Spec 3: Agent/Software Integration and four other application specifications. Specs 1-3 are called normative specifications that prescribe technical aspects of multi-agent systems like how agents are managed, what language and interaction protocols agents use to communicate with each other, and how agents and non-agent software interact in the Internet. Four application specifications are called informative, which are

intended to verify the normative technical specifications and to promote the development and deployment of agent-based applications.

In 1998, FIPA produced second set of specifications: FIPA 98. FIPA 98 includes normative specifications of Spec 8: Human-Agent Interaction, Spec 10: Agent Security Management, Spec 11: Agent Management Support for Mobility and Spec 12: Ontology Service [3]. Spec 1: Agent Management and Spec 2: Agent Communication Language were also revised and updated.

Comtec Corporation (in October 2000, Communication Technologies was established by the management buyout of Comtec Corporation) implemented FIPA 97 Specs 1-3 and released the source code of world's first free implementation of FIPA in September 1998. In October 1999, Comtec added the implementation of Spec 12: Ontology Service to its agent platform and also made the source code open to public. A live service of Comtec Agent Platform had been offered in order to facilitate the global interoperability trials of FIPA-based agent platforms using the Internet since February 9, 1999, following the first interoperability trials at FIPA Seoul meeting in January 1999. Unfortunately, the live service has been stopped during the chaos of the transition of the company from Comtec Corporation to Communication Technologies [6].

In October 1999, FIPA decided that it no longer abide by a yearly cycle (such as FIPA 97 and FIPA 98) and release the specifications asynchronously. FIPA also undertook major restructuring of the organization and introduced a new document policy and lifecycle management of the specifications. According to the new lifecycle schema of the specifications, the documents follow the steps below:

- (1) Preliminary, where the technical committee decides that the document is worth public review;
- (2) Experimental, where FIPA Architecture Board, the top technical authority within FIPA, confirms that experimental implementation and interoperability trials of the implementations is encouraged;
- (3) Standard, where the interoperability of the multiple implementations by different developers has been achieved; and
- (4) Deprecated and Obsolete where documents are no longer valid and archived for historical reference purpose only.

As of April 2001, most of the latest specifications are at the Experimental stage. Such specifications include abstract architecture, agent management, agent communication

---

\* Communication Technologies. 2-15-28-6F Omachi Aobaku, Sendai 980-0804 Japan. Tel +81-22-222-2591

\*\* Iwate Prefectural University. 152-52 Takizawa-aza-sugo, Takizawa, Iwate 020-0173 Japan. Tel +81-19-694-2000

\*\*\* Sendai Foundation for Applied Information Sciences. 5-12-55 Tsutsujigaoka Miyaginoku, Sendai 983-0852 Japan. Tel +81-22-298-9653

language and content languages, interaction protocols, message transports and message encodings, agent/software integration, and ontology service [4]. Previous FIPA 97 and FIPA 98 specifications are now Obsolete status. Unfortunately, Comtec agent platform is based on FIPA 97 and FIPA 98. It is currently not yet fully compatible with the latest Experimental specifications. Since FIPA 97 and FIPA 98, there are some major changes in agent management and message transport in the latest specifications. However, the differences between FIPA 98 Spec 12: Ontology Service and the latest experimental ontology service specification are just cosmetic and there is no functional change.

## 2. COMTEC AGENT PLATFORM

Comtec Agent Platform is a straightforward implementation of agent management, agent communication language and agent/non-agent software integration functions of the FIPA 97 specifications. Figure 1 shows the basic building blocks of the platform.

Application Agents (Non-agent Software Integration and Ontology Service)	
Agent Platform Agents (ACC, AMS and DF)	
Agent Communication Language Library	
Kawa Scheme Interpreter	Java IDL
Java 2 Standard Edition (JDK 1.2)	

Figure 1. Basic building blocks of Comtec Agent Platform

Java 2 is the foundation for the platform. Hardware and operating system independency is achieved by using Java. Kawa [1] is a Scheme (a dialect of LISP) interpreter written in Java. Kawa allows Scheme programmers to access Java classes and vice versa. Java IDL is adopted to implement IOP-based communication between agents as specified by the agent management. Agent communication language library is a collection of primitives of ACL communicative acts, content language interpreters SL0 and SL2, basic design patterns of an agent, and interaction protocol handlers. Agent Platform agents are ACC (Agent Communication Channel), AMS (Agent Management System) and DF (Directory Facilitator). (N.B., ACC is no longer an agent in the latest Experimental specification.) On top of the platform agents are the application agents such as ARB (Agent Resource Broker) and software wrapper agents of agent/software integration, and ontology server and ontology client agents of the Ontology Service specification.

Figure 2 depicts the difference between Comtec's implementation and FIPA's reference model of the configuration of the agent platform. With FIPA 97 reference model, agents communicate via Internal Platform Message

Transport mechanism, or IPMT. The communications protocol used in IPMT is not specified by FIPA so that agents can use platform-specific or programming language dependent message transport mechanisms such as Unix IPC, Java RMI, HTTP, SMTP etc. To ensure the communications between agents on different agent platforms that use different IPMT protocols, the ACC must speak common 'baseline protocol', which is CORBA IIOP to forward the message to ACCs on other platforms.

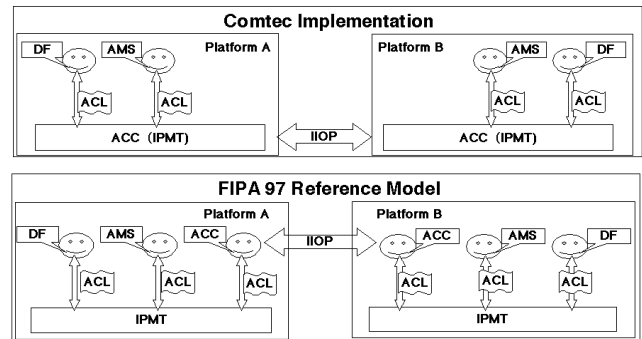


Figure 2. Agent platform configuration models

Comtec's implementation simplifies the FIPA reference model by using IIOP as the internal messaging protocol, too. Therefore, ACC is responsible for both inter- and intra-platform message exchange. Figure 3 below details out how an agent talks to the ACC.

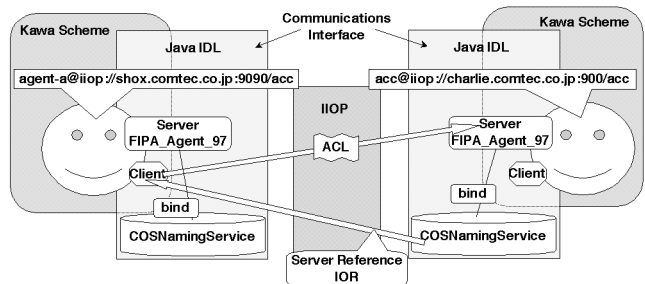


Figure 3. Agent communications

In this picture, agents (agent-a and ACC) are composed of two parts. The main part is written in Kawa Scheme, which controls agent's behavior. The communications interface utilizes Java IDL to use IIOP transport protocol. To establish the communications between the agents, COS naming service (tnameserv) is employed for name resolving. The ACC registers the server (receiver) reference with the COS naming service and agent-a (sender) retrieves the object reference from the naming server. Agent-a sends a message to ACC using the IOR and ACC then forwards the message to the recipient agent (the recipient address is specified in the :receiver slot of the

ACL message) with the same manner (which is not described in Figure 3).

### 3. FIPA ONTOLOGY SERVICE SPECIFICATION

After the initial release of the Agent Communication Language specification, FIPA Technical Committee C, which was responsible for the agent communications, began the work of ontology service in January 1998. Since the basic framework of the agent communications was established by specifying the ACL, content language SL and interaction protocols such as contract net and auctions, it seemed natural to proceed to deal with the problem of how agents can share a common ontology in order to make a meaningful conversation. The task finished in October 1998 and the document was published as FIPA 98 Specification Part 12: Ontology Service. The original document itself has been in obsolete status since revised version was edited according to the new document management and lifecycle policy described above. However, the differences between the old and new documents are mostly cosmetic and no significant change has been made. A Japanese translation of the original FIPA 98 specification [5] is also available, which was made by Intelligent Agent Society of Japan.

FIPA ontology service is basically an agent wrapper of OKBC [2]. Open Knowledge Base Connectivity, or OKBC, is a set of applications programming interfaces (API) that connects front-end user applications and back-end knowledge bases (KBs). Like ODBC (Open Data Base Connectivity) or JDBC (Java Data Base Connectivity), OKBC connects to a wide variety of KB servers (such as Ontolingua, Loom and Cyc) in the back-end while allowing the client user applications to be able to access these KBs via standardized front-end API. The reference implementation of OKBC, which is available from Artificial Intelligence Center of SRI International and Knowledge Systems Laboratory of Stanford University, comes with C, Java and Common Lisp programming language bindings of the API. OKBC specification also defines a frame-based knowledge model written in KIF that is used to describe the contents of the API.

The problem with OKBC in FIPA perspective is that its client-server model API does not fit for the FIPA agent standard style where communicative interface of the agent is specified but programming language API is outside the scope of the specification. Therefore, FIPA decided to wrap the OKBC front-end API with ACL-speaking agent, which is called Ontology Agent (OA). (The back-end KBs are mentioned as ontology server.) Client agents of the OA utilize standard FIPA ACL (query-if, query-ref, request etc.), interaction protocols (FIPA-request and FIPA-query), white page and yellow page directory services (DF and AMS) and agent message transport (IMTP/ACC) to talk to the OA in order to access the ontology service behind the OA. Ontology clients store, modify, delete or query the ontology with the OA to share the common ontology with other client agents. Figure 4 below, which is taken from the specification document and

modified a little to better emphasize the role of OKBC, shows the reference model of FIPA Ontology Service.

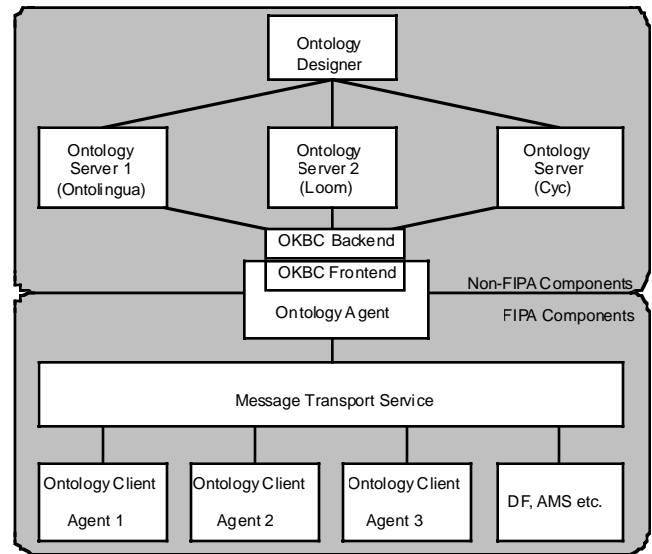


Figure 4. Ontology Service reference model (modified)

The FIPA Ontology Service specification adopts OKBC knowledge model as FIPA-meta-ontology (that is, an ontology that is used to access the Ontology Agent). It also defines actions and a predicate used in the content of the conversation with the OA, which are shown in Tables 1 and 2 below. FIPA-ontol-service-ontology consists of these actions, predicate and FIPA-meta-ontology. For the complete list of FIPA-meta-ontology, see the OKBC specification, part of which is also included in the FIPA Ontology Service specification with the permission of the original authors.

Table 1. Predicate defined in FIPA-ontol-service-ontology

Predicate	Description
(ontol-relationship <o1> <o2> <level>)	It is true if and only if there is a relationship of type level between the ontology o1 and the ontology o2.

Table 2. Actions defined in FIPA-ontol-service-ontology

Actions	Description
(assert <predicate>)	This action asserts the predicate in the ontology specified by the :ontology parameter of the ACL.
(retract <predicate>)	This action retracts (cancels) the predicate in the ontology specified by the :ontology parameter of the ACL.

(atomic-sequence <action>*)	This action introduces a transaction-type sequence of actions, which is treated as if it was a single action. The result of the action is one of the two cases: (1) all actions are successfully done; or (2) none of the actions is made. It is used to modify an existing ontology by combining the actions of retraction and assertion, for example. The mechanism to maintain the consistency inside the sequence (rollback if necessary) and to protect values from outside the sequence is dependent on the implementation.
(translate <expression> <translation-description>)	Translates the expression as specified by the translation-description. It should be used with FIPA-Request interaction protocol.

The specification includes two informative annexes. One of them sets forth an introduction to logical foundations of the ontology and conceptualization, contributed by Nicola Guarino (LADSEB-CNR). The other describes guidelines to define a new ontology used by the applications, contributed by Assuncion Gomez-Pérez (Universidad Politécnica de Madrid).

#### 4. IMPLEMENTATION OF THE ONTOLOGY SERVICE

The implementation of the Ontology Service specification is pretty straightforward based on the specification. Figure 5 shows the overall view of the internal composition of the ontology agent.

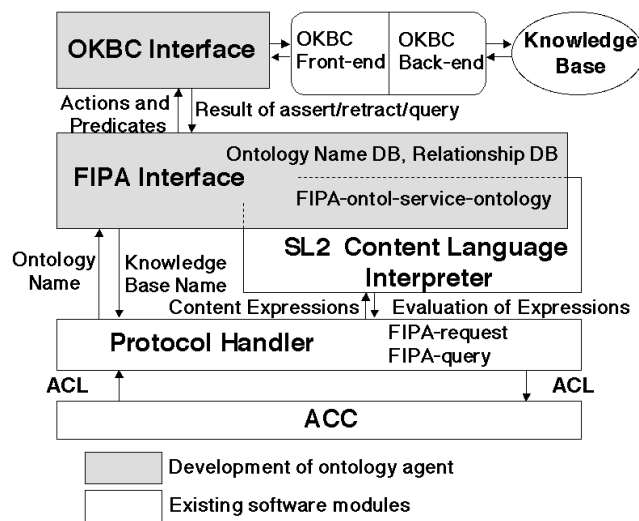


Figure 5. Overview of the ontology agent

The development of the OA is divided into two parts. The first part is an interface to the OKBC front-end. From OKBC point of view, the OA is one of the front-end user applications. This part is also responsible for managing OKBC knowledge model and FIPA-meta-ontology. The second part is the FIPA interface where the agent wrapper is implemented. The FIPA interface includes functions such as ontology naming, ontology relationship and FIPA-ontol-service-ontology management. It utilizes existing interaction protocol handlers for FIPA-query and FIPA-request, which are prescribed in the specification for client agents to talk to the OA. The content language SL2 interpreter from the agent communication language libraries of the Comtec Agent Platform is also used to express the actions, objects and predicates in the ACL message.

#### 4.1 OKBC Interface

The OKBC interface connects to the OKBC front-end using the Java binding of the OKBC API. Figure 6 below shows the OKBC interface.

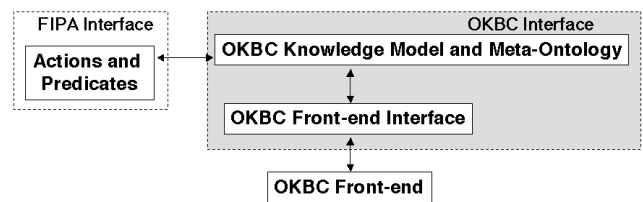


Figure 6. OKBC interface

It performs the following functions:

- Connecting to and disconnecting from the OKBC front-end;
- Creating, deleting and identifying a back-end knowledge base;
- Operations on frames, slots and facets including creating, modifying and deleting them;
- Operations on class and individual hierarchies;
- Operations on predicates associated with the frame-sentence;
- Converting the SL symbols and the OA predicates to the OKBC objects and the OKBC predicates, and converting vice versa;
- Converting the SL actions to the OKBC operations and the OKBC results to the SL expressions; and
- Handling of the OKBC errors. Depending on the severity, it either disconnects from OKBC or generates appropriate exceptions and SL error expressions to be passed to the FIPA interface.

#### 4.2 FIPA Interface

The FIPA interface is an agent wrapper that takes care of generating and interpreting SL actions and predicates, and ACL communicative acts based on appropriate interaction protocols. It also processes the registration with DF, and

management of ontology names and relationship between the ontologies. Figure 7 below displays the internal composition of the FIPA interface.

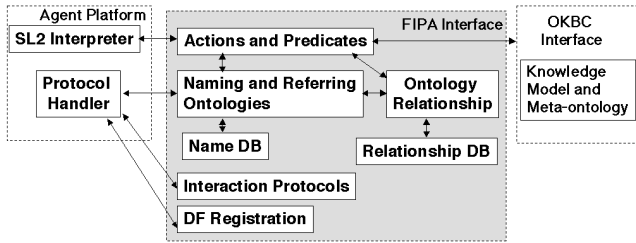


Figure 7. FIPA interface

The following are descriptions of the sub-modules.

#### 4.2.1 Naming and Referring Ontologies

This sub-module processes naming of ontologies. Each ontology must have a unique logical name, which is used in the :ontology slot of the ACL message. The logical ontology name corresponds to the physical name of the KB that is connected to the back-end of OKBC. The ontology name database manages the mapping of the logical name and physical name of the ontologies.

This module deals with the name database to insert, delete, modify and query the logical and physical names of the ontologies. These requests come from actions and predicates module and ontology relationship module.

#### 4.2.2 Relationship between Ontologies

Six relations are defined between ontologies in the specification: Extension, Identical, Equivalent, Weakly-Translatable, Strongly-Translatable and Approx-Translatable. These relationships of the ontologies are expressed in the ontol-relationship predicate with the form of (ontol-relationship <O1> <O2> <relation>). This sub-module is responsible for managing and maintaining the ontology relationship database. It inserts, deletes, modifies and answers the relationship according to the request coming from the actions and predicates sub-module and naming and referring ontologies sub-module. Table 3 below explains the definition of the relations between two ontologies.

Table 3. Relations between ontologies

Relation	Description
<O1> <O2> Extension	Ontology O1 extends the ontology O2.
<O1> <O2> Identical	Ontologies O1 and O2 are identical.
<O1> <O2> Equivalent	Ontologies O1 and O2 are equivalent.

<O1> <O2> Weakly-Translatable	The source ontology O1 is weakly translatable to the target ontology O2.
<O1> <O2> Strongly-Translatable	The source ontology O1 is strongly translatable to the target ontology O2.
<O1> <O2> Approx-Translatable	The source ontology O1 is approximately translatable to the target ontology O2.

Please look at the specification document for the complete description of these relations. It should be noted here, however, that the following properties hold between the levels of relationships:

- Strongly-Translatable  $\Rightarrow$  Weakly-Translatable  $\Rightarrow$  Approx-Translatable
- Equivalent (O1, O2)  $\Rightarrow$  Strongly-Translatable (O1, O2)  $\wedge$  Strongly-Translatable (O2, O1)
- Identical  $\Rightarrow$  Equivalent

The relationship sub-module is responsible for tracking these implications and correctly answering the queries from the clients about the relationship even if the implications are deeply nested.

#### 4.2.3 Registration with DF

Directory Facilitator, or DF, is a yellow page directory service of agents in an application domain defined in the Agent Management specification. It stores the agent description and service descriptions of the agents and answers queries about them from other agents. Agent must register with the DF in order to advertise the capabilities of itself. This sub-module prepares the agent description and service descriptions of the OA and registers with the DF using the FIPA-request interaction protocol when the OA is activated. (Registration with AMS, which is a white page directory service of the agent platform, is also a mandatory step of initializing the agent and thus automatically handled by the agent communications library.)

#### 4.2.4 Actions and Predicates

This sub-module handles the actions and predicates used in the OA. There are two cases in terms of the actions and predicates that are handled here.

- (1) OKBC-related actions and predicates

In this case, this module calls OKBC interface to process the actions and predicates.

- (2) Actions about ontology name and relationship between ontologies

If the action is about the ontology name or the relationship between the ontologies, naming and referring sub-module and ontology relationship sub-modules are called to process the request.

The actions include `assert`, `retract`, `atomic-sequence` from the `FIPA-ontol-service-ontology`, and `inform`, `request`, `query-if`, `query-ref`, `agree` and `cancel` from the standard communicative act library. The `translate` action of the `FIPA-ontol-service-ontology` is not implemented. This is the only feature that is not implemented inside the specification. The OA registers with the DF expressing that it is unable to process the `translate` action from one ontology to another.

Note that the basic policy of FIPA about the implementation of the specification is that the developer can choose which subset of the document to implement and which part not to implement, as far as the consistency of the behavior of the agent is maintained. The agent must properly register with DF and AMS about its capabilities: which functions are implemented and which functions are not implemented. (Some features, for example, which communicative acts the agent understands, are not registered with DF. This problem should be resolved in the future version of the FIPA architecture.) It is a responsibility of the applications designers (or, maybe the responsibility of the agents themselves if the agents are intelligent enough) to identify the necessary features of the agents and integrate the multi-agents by asking the DF for the service description of the agents in order to achieve the interoperability of the agents and deploy the applications.

The `atomic-sequence` action needs a special attention. This action introduces a transaction-like (i.e., the ones found in SQL RDB) sequence of actions, which is treated as if it was a single action. The result of the `atomic-sequence` action is strictly one of the two cases: (1) all actions in the sequence have been successfully done; or (2) none of the actions in the sequence has been committed. Like RDB transactions, the OA takes care about the consistency between before and after the `atomic-sequence`, and inside and outside of the `atomic-sequence`. Firstly, the OA maintains the transactions log of each action in the sequence and rollbacks to the previous state of the KB, which is just before the `atomic-sequence` is initiated, if one of the actions in the sequence fails by any reason. Secondly, while the sequence of the actions is processed, the OA provides a virtual view of the state of the KB, which is just before the `atomic-sequence` is commenced, to other transactions so that outside agents cannot see the interim, possibly inconsistent, stage of the sequence. The OA commits the changes made by the sequence of actions if all of the actions in the sequence have been successfully completed without an error.

#### 4.2.5 Interaction Protocols

This sub-module generates function closures that process the specified interaction protocols. In the OA, `FIPA-request` and `FIPA-query` are the only two interaction protocols used. The

main loop of the OA evaluates the interaction protocol handler closure to control the conversations between the OA and the client agents or the DF according to the specified interaction protocol. (Note that closures are heavily used everywhere in the program in general.)

## 5. DEMONSTRATION APPLICATION

The purpose of the demonstration application of OA is to identify the strengths and weaknesses of the OA and its client agents that constitute the real-world applications. Web-based on-line shopping mall was selected as such a practical application. Comtec's business partner, Intercraft Corporation (<http://www.intercraft.co.jp/>), has developed an on-line shopping application that is called Gumbo (<http://www.gumbo.ne.jp/>). We received the source code from Intercraft and developed a version of the application that makes use of the ontology service, which is called Ontology Gumbo. As a consequence, we could not only test the Ontology Gumbo but also compare the performance of Ontology Gumbo with the original Gumbo. The following Figures 8 and 9 show the structure of the original Gumbo and the modified part of Ontology Gumbo. Please note that Ontology Gumbo server is implemented as a client agent of the OA.

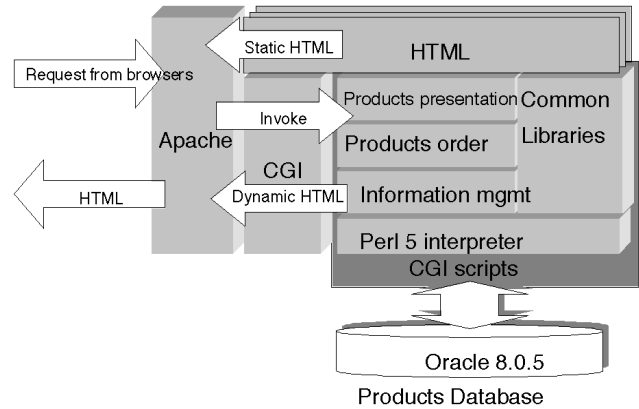


Figure 8. Original Gumbo server

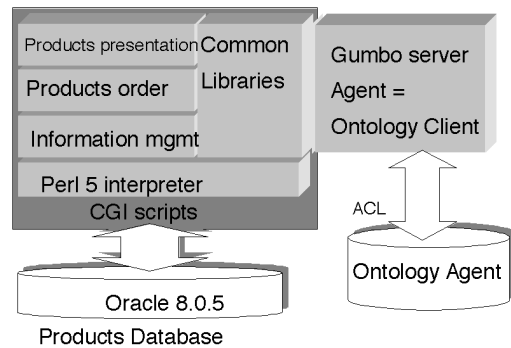


Figure 9. Ontology Gumbo server (part)

By developing the demonstration application, firstly, we wanted to know how easily an existing application could be

modified to become the ontology client agent that utilizes the OA. Then we have tested both original and Ontology Gumbo systems in terms of resource (CPU, memory, network etc.) utilization, response time, ease of systems modification such as adding and removing database schemata, and fault tolerance. Here we report some of the results.

### 5.1 Development of the Demonstration Application

The development of the demonstration application was made by a programmer who had a pretty good knowledge on Oracle, Perl, Apache and Kawa, but did not hear anything about ontology. The following Table 4 summarizes hours that actually took for the programmer to modify the original Gumbo to support the ontology service as a client agent of the OA. # of lines indicates modified or added number of lines in the program code.

**Table 4. Development hours**

Work Items	Details	Hours	# of lines
Preparation	Study on ontology	15.0	
	Setting up the development environment	3.0	
Creating the ontology	Abstraction of the schema and programming logic	6.0	
	Description of the ontology	3.0	219
	Japanese characters processing	0.5	13
	Assertion of the ontology	0.5	
Gumbo ontology client	CGI interface	3.0	57
	OA interface	4.0	149
CGI modification	Ontology client interface	2.0	70
	Utilizing the ontology	1.5	96
<b>Total</b>		<b>38.50</b>	<b>604</b>

Although Gumbo is not a very big system (total number of lines: 16144), which offers just minimal functionalities of the on-line shopping mall, it took less than one person-week to change it to use the ontology agent. Please note that half of the time is spent by preparation. It would not take this much if the same person does the similar task.

However, it should be also noted that advantage of having explicit and external ontology over implicit, internal and hard-coded ontology (database schema) was not made clear with such a small-sized applications. For example, there was no significant difference<sup>1</sup> in terms of the development time

<sup>1</sup> A reviewer suggested to compare the figures. Unfortunately, the record was lost and we are sorry we cannot present the exact numbers.

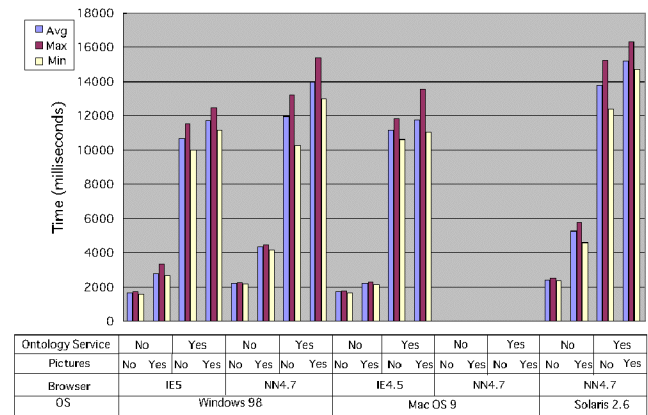
between the original and Ontology Gumbo systems when new schema had to be added and the unified end-user view must be presented. This could be partly because the developer was an Oracle specialist and new to ontology; partly because the task of integrating multiple database schemata was not appropriate for the OA.

### 5.2 Run-Time Performance Comparison

We have compared the run-time performance of the original and Ontology Gumbo systems. The hardware and software configuration is as follows:

- Ontology Agent and OKBC server – Sun Ultra 5  
270MHz UltraSPARC Iii processor, 256MB memory, 4.3GB disk, Solaris 2.6, JDK 1.2.2, OA, OKBC and TKB
- Database and web server – Sun Ultra 5  
270MHz UltraSPARC Iii processor, 256MB memory, 4.3GB disk, Solaris 2.6, JDK 1.2.2, Oracle 8.0.5, Apache 1.3.9, Perl 5.005\_03, original and Ontology Gumbo
- Gumbo client – PC  
233MHz Pentium MMX processor, 64MB memory, 4.3GB disk, Windows 98, Netscape Navigator 4.7, Internet Explorer 5
- Gumbo client – Apple iMac  
333MHz PowerPC G3 processor, 192MB memory, 6.4GB disk, Mac OS 9, Netscape Navigator 4.7, Internet Explorer 4.5
- Gumbo client – Sun Ultra 5  
270MHz UltraSPARC Iii processor, 256MB memory, 4.3GB disk, Solaris 2.6, Netscape Navigator 4.7

These machines were connected to a closed 10Base-T LAN. Typical end-user procedures of browsing a product catalog were timed for the combinations of OS, browsers, with/without pictures and with/without ontology service. The data was not successfully acquired for the Netscape Navigator 4.7 on Mac due to unstable Java Runtime Environment. Figure10 displays the results.



**Figure 10. Gumbo performance comparison**

As shown in the chart, overhead of using the ontology service often exceeds ten seconds. One of the reasons of the bad performance is that most of the Java optimization options had to be turned off in order to avoid the nasty bugs related to the optimization (JIT compiler, native thread etc.)

## 6. FUTURE WORKS AND CONCLUSIONS

Agentcities project (<http://www.agentcities.org/>) is a worldwide initiative designed to help realize the commercial and research potential of agent-based applications. The objective is to construct a worldwide network of agent platforms based on the FIPA standard. Each platform, which is called a City, will be supported by different organizations around the world and host diverse populations of agents that are able to access each other's services. Communication Technologies committed to take part in the project to host the Sendai City, where the office is located, based on the Comtec Agent Platform and the Ontology Agent. However, in order to host the city platform and to be able to connect to other cities, we have to work on some major update of the code because current platform is based on FIPA 97 and FIPA 98 specifications. Currently (as of April, 2001), FIPA has published most of the specifications as Experimental status, which means that the experimental implementations and the interoperability trials of the implementations by different developers are encouraged in order to promote the specifications to the Standard status.

Agentcities project is a great motivation for us to update the platform and add the following features to the platform:

- HTTP transport of the agent communications;
- XML representation of the ACL and SL;
- Gateway between the HTTP/XML world and existing IIOP/S-expression world;
- Deploy the Ontology Agent on top of the revised platform; and
- Performance tuning of overall platform.

In addition, we are committed to provide other cities with the ontology service, which is based on, as far as we know, the only implementation of the FIPA Ontology Service specification. It is expected that by utilizing the ontology service, heterogeneous agents developed by different participants and located in different cities will be able to share a common ontology to establish a basis for a meaningful conversation.

In conclusion, we have presented our experience of implementing FIPA Ontology Service specification. It is our big regret that we have not published a paper on the software

immediately upon the release of the source code in October 1999 and several key evaluation results were lost or not recorded. Anyway, we believe the technology of the ontology service, which combines the proven power of OKBC and standard FIPA specifications, must be fully exploited in the real-world application in order to facilitate the advanced knowledge sharing among the computers and the human beings.

## 7. ACKNOWLEDGMENTS

The authors acknowledge Information-technology Promotion Agency of Japan (IPA) for funding the development of FIPA 97 agent platform (Advanced Information Promotion Assistance Software Enrichment and Cultivation Project, under the contract 9JOUGIOUDAI569GOU) and the development of FIPA 98 Ontology Service (Next Generation Digital Technology Applications and Basic Foundations Support Technologies Development Project, under the contract 10JOUGIOUDAI908GOU). We also thank Artificial Intelligence Center of SRI International and Knowledge Systems Laboratory of Stanford University for their OKBC specification and implementation. The demonstration application would not have been made possible without the great courtesy of the Intercraft Corporation that developed the original Gumbo system and kindly assisted us to build the Ontology Gumbo. Last but by no means least, our special thanks to 1998 FIPA Technical Committee C members led by the chair Fabio Bellifemine of CSELT, and general membership of FIPA that contributed to discuss and publish the Ontology Service Specification.

## 8. REFERENCES

- [1] Bothner, P. The Kawa Scheme System. <http://www.gnu.org/software/kawa/>
- [2] Chaudhri, V. K., Farquhar, A., Fikes, R., Karp, P. D., and Rice, J. P. Open Knowledge Base Connectivity 2.0. Technical Report KSL-98-06, Stanford University KSL, 1998. <http://www.ai.sri.com/~okbc/spec.html>
- [3] FIPA. FIPA0006: FIPA 98 Specification Part 12: Ontology Service. <http://www.fipa.org/specs/fipa00006/>
- [4] FIPA. FIPA0086: FIPA Ontology Service Specification. <http://www.fipa.org/specs/fipa00086/>
- [5] Intelligent Agent Society of Japan. Japanese translation of FIPA 98 Specification Part 12: Ontology Service. <http://fipa.comtec.co.jp/fipatrans/j98v1p12.PDF>
- [6] Suguri, H. et al. Comtec Agent Platform. <http://fipa.comtec.co.jp/ap/>