Software Agent-Oriented Frameworks for Heterogeneous Information Access

Zakaria Maamar
Computer Science Department
Research Center on Geomatics
maamar@ift.ulaval.ca

Bernard Moulin
Computer Science Department
Research Center on Geomatics
moulin@ift.ulaval.ca

Laval University, Ste-Foy, Qc, G1K 7P4, Canada

Abstract

In this paper we propose an approach used to design an interoperable environment for distributed and heterogeneous systems. To overcome such constraints, we suggest to develop software agent-oriented frameworks which provide users with relevant and up-to-date data from such systems. We present an application of this approach to the SIGAL project in which we are developing an interoperable environment for georeferenced digital libraries.

1 Introduction

With the growing number of information technologies, modern organizations tend to be more and more based on disconnected sets of operational systems, that evolve in *distributed* and *heterogeneous* environments. In this context and considering the increasing needs of organizations, sharing informational resources has become important. Furthermore, to remain competitive, such organizations have to interconnect their systems in order to provide users with relevant and up-to-date data. The capability of interaction of multiple distributed and heterogeneous systems is called *Interoperability*.

There exists a number of studies in the field of systems interoperability, in particular those conducted by the *Object Management Group* (OMG) [OHE96]. OMG aims at applying object-oriented technology to distributed contexts, by suggesting a communication infrastructure called *Common Object Request Broker Architecture* (CORBA). In fact, OMG's objective is to set up an object-oriented client/server architecture. When several objects populate the same software environment, the authors in [CP95] propose to integrate them within a framework. An object-oriented framework is an organized environment for running a collection of objects. However, it is not easy for designers to develop such frameworks and to monitor their evolution. Therefore, our work aims at providing guidelines for designing frameworks [MM97a].

An interoperable environment has to make local systems interoperate while remaining *autonomous* and *independent* from each other. To this end, we propose to introduce several specialized components called *software agents* [Nwa96] which are the *front-ends* of the interconnected systems and have the capabilities to act on their behalf. Furthermore, software agents can assist users in the accomplishment of their tasks, collaborate with each other to jointly solve different problems, and answer users' needs. In order to enable these software agents to cooperate, we suggest to gather them into *teams* which will build up *software agent-oriented frameworks* [MM97b]. In this paper, we present how software agent-oriented frameworks can be used to set up an *interoperable environment*, which will provide *Intelligent Access to Heterogeneous and Distributed Systems*.

Our research on software agent-oriented framework

Proceedings of the 4th KRDB Workshop
Athens, Greece, 30-August-1997

(F. Baader, M.A. Jeusfeld, W. Nutt, eds.)

http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-8/

is applied to the SIGAL[1] project, in which we are developing an interoperable environment for Georeferenced Digital Libraries (GDL). A GDL is an information base describing geodocumentary resources available in an organization. Currently, each GDL is characterized by its own informational content, its own presentation formats, and its own processing functions. Therefore, it is difficult to access concurrently several GDLs in order to answer users' needs. As a solution, we are developing an interoperable environment of GDLs based on software agent-oriented frameworks [MM97a].

The paper is organized as follows. Section 1 proposes an overview of interoperability issues. Section 2 presents the basic notions of software agents-oriented frameworks. Section 3 provides a description of GDLs. Section 4 presents the main characteristics of the SIGAL environment. Section 5 summarizes the points brought out by our study.

2 A Software Agent-Oriented Framework

Designing software agent-oriented frameworks is a recursive activity that is applied in its turn to frameworks, teams of agents, and software agents.

A *software agent-oriented framework* (Fig. 1) offers a set of *services* that can be requested either by users or by other frameworks [MM97b]. A framework is an environment composed of a *framework supervisor* and one or several *teams of software agents*. The services provided by a framework are performed by different agent teams set up by the framework. These teams are composed of agents selected from a *bank of software agents*. This bank contains several agents having different functionalities which depend on the application to be developed and on the characteristics of the systems to be interconnected.

Teams of agents are structured in different ways according to their responsibilities in the framework. A team is also characterized by a *team supervisor* and a set of roles that agents must fulfill according to their capabilities [MM97a].

Services provided by a framework satisfy specific users' needs such as information search across distributed and heterogeneous systems. When a service is invoked by a user, the framework's supervisor agent activates a *realization scenario* [MM97b], which specifies the functional and structural characteristics of the

[1]It stands for "Système d'Information Géographique et Agent Logiciel" (Software Agent and Geographic Information Systems). It is a project jointly undertaken by the Computer Science Department and the Research Center on Geomatics at Laval University.

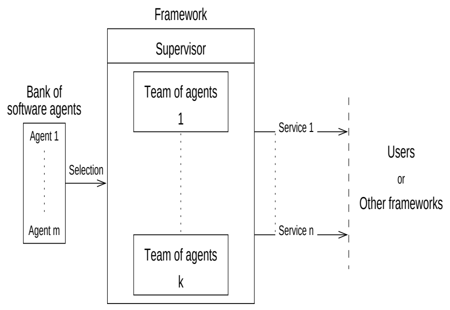

Figure 1: Representation of the Software Agent-Oriented Framework Concept

teams of agents that will perform the various operations required to carry out the service.

A realization scenario is composed of the following elements: software agents' types and roles, scheduling sequences indicating in which order agents should perform their plans, the list of internal and/or external information resources which provide knowledge required by the agents to achieve their operations, and finally an evaluation procedure used to monitor the operations of various agents.

3 An Overview of Georeferenced Digital Libraries

Within an organization that manages spatially referenced data, several types of documents are used to describe the presence and the nature of these data. Such documents include topographic maps, cover maps, aerial photographs, satellite remote sensing images, etc. Managing such documents is a complex task; each one is characterized by its own scale, content, quality, sources, and format. Hence, GDLs can be very useful and helpful. GDLs can improve knowledge of the nature of data, identify the responsibility of "who does what, when, and how", and inform about the physical location of the documents.

Currently, users are accustomed to use GDLs independently from each other. Yet, a more complex task is to combine different GDLs to fulfill their needs. This new reality claims for new alternatives for GDLs interoperability.

If a person visits the several available GDLs, she will easily notice that [PBLM97]:

1. the content of the GDLs differs from one to another;
2. the data standards are different, if not absent;
3. different words are used to represent the same concept and *vice versa*;
4. the user interfaces always differ; etc.

All these differentiating elements would need that users adapt to each GDL's requirements and understand their different information and structural characteristics; which is an impossible task.

4 Presentation of the SIGAL environment

In this section, we describe the major characteristics of the SIGAL environment, in terms of architecture, frameworks, agents, and services. We also describe the knowledge used by the SIGAL's components and an implementation of this environment.

4.1 SIGAL's Architecture

Before elaborating the SIGAL environment, we examined several issues such as the distribution of GDLs, their access rights, communication support channels (local and/or wide-area networks) and similar studies in the field of information systems interoperability [Hsu96, Ker97, PSS92].

Based on these considerations, we proposed an *interoperable multiframework architecture* for the SIGAL environment [MM97b]. This architecture (Fig. 2) is characterized by the use of a *client/server* approach, by the introduction of the technique of *mirror sites*, by the proposition of three types of frameworks (server, client, and local-source), and, finally, by the creation of client frameworks by the server frameworks, whenever it is necessary.

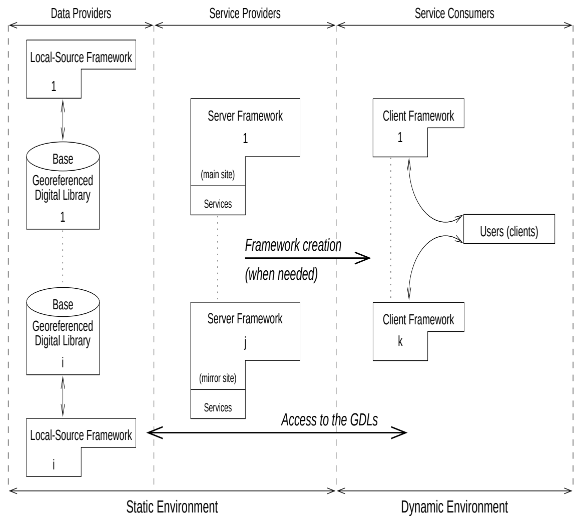

Figure 2: Multiframework Architecture of the SIGAL Environment

By analogy to integration shells [BMCd97] or information agents [SDP$^+$96], the *local-source frameworks* maintain the autonomy and the independence of the GDLs in the interoperable environment SIGAL. Therefore, local-source frameworks interface with GDLs using communication networks, and process the data requests sent by the client frameworks.

The *server framework* is the backbone of the SIGAL environment; it monitors all the operations needed to support the services offered to the users and to other frameworks. To avoid *overloading* the server framework, we suggested to duplicate it on *mirror sites*. However, in such a situation, it is important to maintain the coherence of the common information between the server frameworks.

When users need information from several distributed and heterogeneous GDLs (which means an intelligent access to these GDLs), they invoke relevant services from the server framework. The invocation of such services initiates the creation of a *client framework* on the *user's machine* [MM97b]. The server framework delegates operations to the client framework and limits its involvement to the monitoring of these operations. Once all operations are completed, the client framework can either be deleted or stored for later use.

Two categories of services are offered by the SIGAL environment; i.e., the server frameworks [MM97b]:

1. *User services* correspond to user query processing, that fulfill needs involving several GDLs. The user formulates his information query independently from the distribution and heterogeneity constraints of the GDLs.

2. *SIGAL services* support insertion or deletion of a GDL, as well as the modification of GDLs' informational content.

Based on SIGAL's architecture and on the services offered by SIGAL environment, different types of agents are identified [MM97b]: *Coordinator-Agent*, *Domain-Agent*, *Help-Agent*, *Interaction-Agent*, *Interface-Agent*, *Knowledge-Agent*, *Learning-Agent*, *Resolution-Agent*, and *Scenario-Agent*. In this paper, we explain some of them.

1. *Domain-Agent*: it resolves knowledge disparities among GDLs by using the common knowledge of the Ontology Base.

2. *Interface-Agent*: it assists users in specifying their needs by proposing a set of *formulation patterns*.

3. *Knowledge-Agent*: it knows the protocols through which a GDL accepts requests and provides back information. This agent also monitors informational updates occurring in the GDL.

4. *Resolution-Agent*: it processes user's queries. The resolution process may require the decomposition

of a query into sub-queries, each of which is sent to the appropriate GDL.

4.2 SIGAL's Frameworks

The SIGAL environment uses three types of frameworks: server, client, and local-source. In what follows, we describe the local-source framework.

Each *Local-source framework* (Fig. 3) interfaces with a GDL of the SIGAL environment. It is assigned to a specific server framework in order to keep it informed about the changes occurring in the informational content of the GDL. When such a change occurs, the Domain-Agent of the server framework updates its *Ontology Base*; these updates are then propagated to the other mirror servers. This mechanism preserves the *coherence* of the various copies of the Ontology Base through the network. All update operations are specified by *update scenarios*, that belong to SIGAL services. A local-source framework also processes *data requests* directed to its associated GDL.

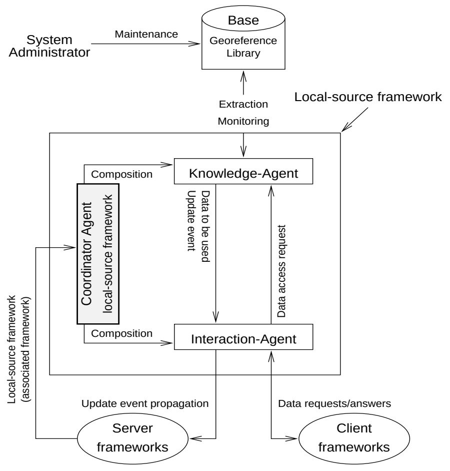

Figure 3: Local-source framework internal architecture

The local-source framework is made up of three agents set in one team of agents (Coordinator, Interaction, and Knowledge). The *Coordinator-Agent* sets the other agents and monitors the operations performed in the local-source framework. The *Knowledge-Agent* interacts with the GDL in order to get the data requested by the *Resolution-Agents* of the client framework. This data is transmitted by the *Interaction-Agent*, which allows the local-source framework to interact with server and client frameworks.

4.3 SIGAL's Ontology

When defining an ontology, we aim at resolving knowledge disparities that may occur when heterogeneous information sources must interact. By establishing an ontology, we offer a common terminological basis for the various interconnected systems, hence reducing the risks that users get inconsistent information.

In SIGAL, ontological disparities exist at different levels. First, being generally developed in an independent way, GDLs present disparities in the *vocabulary* used to describe their data, which makes it difficult for users to consult several GDLs simultaneously. Second, current GDLs do not *help* users when formulating their requests. Moreover, a user has to express his needs according to his own vocabulary and to his own comprehension of the domain. Unfortunately, a user is not always able to get the information he is searching for, because GDLs cannot adapt to these terminological differences (disparity between user's ontology and GDL's ontology).

When defining the SIGAL's ontology, we have used the concept of *meta-data* [Hsu96], defined as *data which describes data of the analyzed domain.* The description allows to specify the data structures, its domain values and its functional and semantic interpretations. We have developed a meta-data model (Fig. 4) which provides a concise description of the information manipulated by each GDL. This model is called Ontology Base and is managed by the Domain-Agent of the server framework.

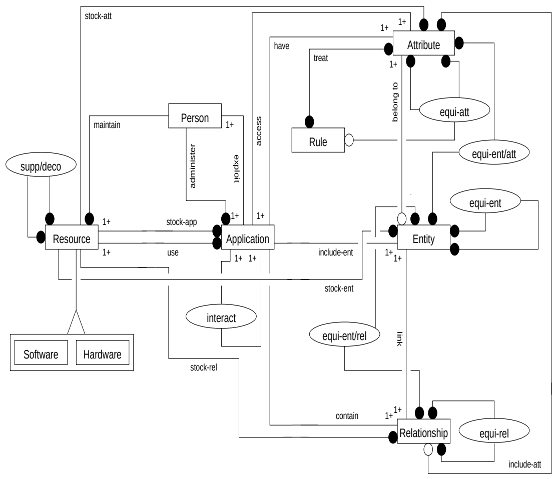

Figure 4: The Meta-Data Model of the SIGAL Environment

The meta-data model is composed of *meta-entities* and *meta-associations*. For example, the meta-entity *Entity* describes all the existing entities in the GDL data models. The meta-entity *Rule* identifies *semantic-equivalence links* that exist between attributes belonging to different GDLs. These equivalence links are managed by the meta-association *equi-attr*.

4.4 SIGAL's User Query Processing

In an interoperable environment, a user's query is usually decomposed into a set of *sub-queries*. Each one is sent to the system which contains data required to satisfy this query. The sub-queries and their *syntactical* and *semantical* forms depend on the functional type of systems to interoperate. Therefore, we propose the *autonomy* and *complementarity* criteria, defining what is the nature of the sub-queries to generate and how are they related. These two criteria are defined as follows:

1. *Autonomy criteria*: local systems are independent and do not need to cooperate. A sub-query, identical to the user query, is generated for every system which is able to satisfy the query. Hence, the sub-queries obtained from the query decomposition are semantically *identical*, but use *different* concepts in their formulation.

2. *Complementarity criteria*: local systems are not independent and need to exchange data. A sub-query is generated for each system which contains part of the information needed to satisfy the query. Hence, the sub-queries are semantically *different* and use *different* concepts in their formulation.

The SIGAL environment satisfies the autonomy criteria.

In the SIGAL environment, a user query processing is decomposed into four steps. The first step specifies user's needs by using the concepts of the ontology. The next step identifies the needed GDLs according to the *autonomy criteria* and provides with the adequate sub-queries. The third step processes these sub-queries by requesting data from their associated local-source frameworks. The last step provides the user with answers to his needs. All these steps particularly rely on the knowledge of the Ontology Base, which considers semantic disparities and implantation structures of the concepts used by each GDL.

4.5 SIGAL's Prototype

Our objective is to set up an *interoperable environment* for GDLs. In order to reach this objective, we have developed software agent-oriented frameworks and distributed them across a network. In order to manage the evolution of these frameworks, we need communication protocols that enable us to specify messages and data exchanges. Concerning messages, several protocols are available such as Hypertext Transfer Protocol (HTTP) and CORBA. For data exchanges, the mostly used protocol is Microsoft's Open Database Connectivity (ODBC) driver. However, as a result of the rapid development of Internet and Intranet technologies, another protocol has been suggested: the Java Database Connectivity (JDBC) driver [PM96].

In the development of the SIGAL prototype, we have used the Object Request Broker[2] (ORB) *VisiBroker for JAVA* developed by Visigenic Company, JAVA as a programming language (particularly, its applet mechanisms), and the JDBC driver in order to connect our several informational resources. We have implemented the three types of frameworks which compose a distributed system. Figure 5 illustrates the SIGAL prototype. Currently, It is dedicated to the realization of user services. The following are relevant aspects of this prototype:

1. The client applet represents the client framework and runs on the user's computer. This applet, which integrates the JDBC driver, is able to invoke the server framework services and to send users' queries to the local-source frameworks after their specifications are obtained from the server framework.

2. The server framework is developed on a Web site. This framework uses the JDBC driver in order to connect the Ontology Base.

3. The ORB VisiBroker for JAVA is used to establish communications between the client applets, the server frameworks, and the local-source frameworks.

4. Microsoft's Access database system is used to manage the Ontology Base.

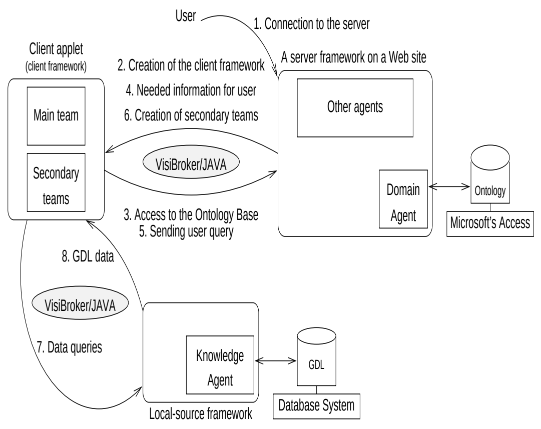

Figure 5: Prototype of the SIGAL Environment

[2]An ORB is a software that respects the conventions defined by OMG for the development of an object-oriented distributed system.

5 Summary

In this paper, we presented the major characteristics of a software agent-oriented framework and its application to the SIGAL project. A framework is made up of teams of software agents, that are able to fulfill services offered to users by the framework. These agent teams can be set up in a unique framework or in several frameworks leading in the latter case to the creation of an interoperable multiframework environment.

A prototype for the SIGAL environment has been developed and is still under improvement. The prototype uses the *JAVA* language to specify framework functionalities and the ORB *VisiBroker for JAVA* to specify the behavior of the SIGAL environment's components (frameworks, teams, agents) during distributed operations.

Acknowledgments

This research is supported by the Natural Sciences and Engineering Research Council of Canada and by the Research Center on Geomatics of Laval University. The authors thank G. Babin and Y. Bédard for fruitful discussions on several issues presented in this paper.

References

[BMCd97] G. Babin, Z. Maamar, and B. Chaib-draa. Metadatabase meets distributed ai. In Peter Kandzia and Matthias Klusch, editors, *International Workshop on Cooperative Information Agents '97 (CIA'97)*, number 1202 in Lecture Notes in Artificial Intelligence, Springer Verlag, pages 138–147, Kiel, Germany, February 1997.

[CP95] S. Cotter and M. Potel. *Inside Taligent Technology.* Addison-Wesley Publishing Company, 1995.

[Hsu96] C. Hsu. *Enterprise Integration and Modeling — the Metadatabase Approach.* Kluwer Academic Publisher, Boston, Mass., USA, 1996.

[Ker97] L. Kerschberg. Knowledge rovers: Cooperative intelligent agent support for enterprise information architectures. In Peter Kandzia and Matthias Klusch, editors, *International Workshop on Cooperative Information Agents '97 (CIA'97)*, number 1202 in Lecture Notes in Artificial Intelligence, Springer Verlag, pages 77–100, Kiel, Germany, February 1997.

[MM97a] Z. Maamar and B. Moulin. C-FOAL : une méthode de conception par frameworks orientés-agents logiciels de systèmes multiagents interopérables. In *Third International Symposium on Programming and Systems*, Springer Verlag, pages 374–396, Algiers, Algeria, April 1997.

[MM97b] Z. Maamar and B. Moulin. Interoperability of distributed and heterogeneous systems based on software agent-oriented frameworks. In Peter Kandzia and Matthias Klusch, editors, *International Workshop on Cooperative Information Agents '97 (CIA'97)*, number 1202 in Lecture Notes in Artificial Intelligence, Springer Verlag, pages 248–259, Kiel, Germany, February 1997. Springer Verlag.

[Nwa96] H.S. Nwana. Software agents: An overview. *Knowledge Engineering Review*, 11(3):1–40, September 1996.

[OHE96] R. Orfali, D. Harkey, and J. Edwards. *The Essential Distributed Objects Survival Guide.* John Wiley & Sons, 1996.

[PBLM97] M.-J. Proulx, Y. Bédard, F. Létourneau, and C. Martel. Catalogage des données spatiales sur le world wide web: Concepts, analyse des sites et présentation d'un géorépertoire personnalisable GEOREP. *Revue Internationale de Géomatique*, 1997. Accepted.

[PM96] P. Patel and K. Moss. *Java Database Programming with JDBC.* Coriolos Group Books, 1996.

[PSS92] M.P. Papazoglou, Laufmann S.C., and T.K. Sellis. An organizational framework for cooperating intelligent information systems. *International Journal of Intelligent and Cooperative Information Systems*, 1(1):169–202, 1992.

[SDP$^+$96] K. Sycara, K. Decker, A. Pannu, M. Williamson, and D. Zeng. Distributed intelligent agents. In *IEEE Expert*, July 1996. http://www.cs.cmu.edu.