# AgentSpace as a Framework to Support Interlibrary Cooperation

**Alberto Silva, José Delgado**
{Alberto.Silva, Jose.Delgado}@inesc.pt
INESC & IST Technical University of Lisbon,
Rua Alves Redol, nº 9, 1000 LISBOA, PORTUGAL

***Abstract***

*In this paper we discuss and argue that the agent-based approach is suitable to design and to develop interlibrary cooperation and universal access to open electronic libraries. Additionally we introduce the AgentSpace system as a generic framework to support, develop and manage this class of applications.*

# 1 Introduction

Digital libraries should provide several services to different entities. The Figure 1 depicts a very simplified and generic schema for this specific problem domain. Basically there are two main participants: the librarians and the end-users or lecturers.

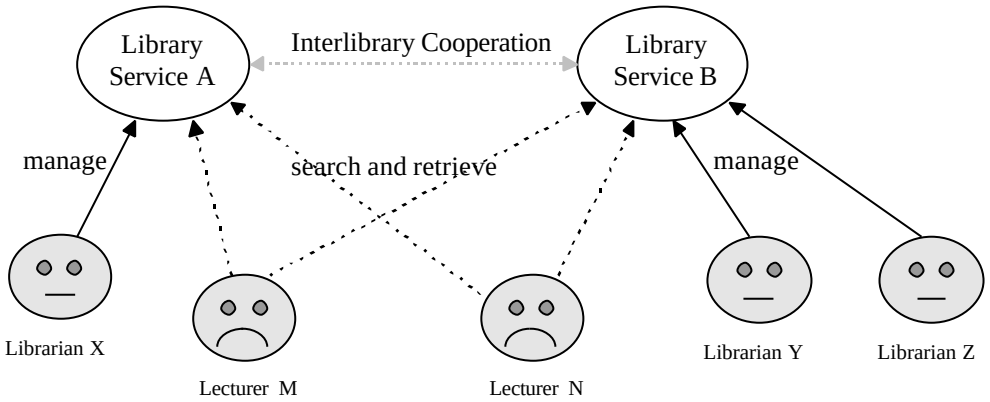


*Figure 1: The traditional vision of the Digital Library – a simplified and generic schema.*

Librarians are responsible by keeping library information; recording and cataloging books and other physical/digital documents (e.g., technical papers, news, images, movies, and discourses); managing users-lectures and users-other librarians; exchanging catalogue information, etc.

On the other hand, lecturers search and retrieve documents based on their respective profiles as well as on the capabilities provided by the library. They've got electronic versions of the documents they want and eventually they must pay for that. Additionally a lecturer may be member of different libraries and so they must learn and interact with a disparate number of different systems and interfaces (this is the reason why they aren't usually happy).

The interlibrary cooperation concerns a restricted vision of the picture of the Figure 1. The interlibrary cooperation involves interaction among a dynamic number of different libraries and their librarians. Its main issue is "how can libraries exchange information/knowledge among themselves in order to improve and synchronize their own knowledge-bases and respective functionalities?". Obviously, one critical point involves the definition of meta-information

standards related to documents, users, and even libraries themselves. However we don't attack this issue in this paper – we assume it is already handled.

The main contribution of this paper is the conception of an agent-based application to support the interlibrary cooperation between a dynamic number of libraries in an open and large-scale environment such as the Internet. We call this application ILC, short for "InterLibrary Cooperation". Additionally we introduce the AgentSpace architecture as a suitable framework to support, develop and manage this class of applications.

The paper is organized as follows. In the Section 2 we present our own definition of agent and agent-based application. In Section 3 we overview the AgentSpace architecture and main components. In Section 4 we identify the main participants and their corresponding relationships related the ICL agent-based approach. We also discuss its main advantages and challenges. Finally, Section 5 summarizes the current status of our work and presents some expectations to the future.

# 2 Agents and Agent-Based Applications

Due to the proliferation of agent definitions with different point of views, scopes and possible applications – see for example [BTV96,GK95,SMD97a] – we start this paper by defining the meaning of an agent in the context of this work.

## *2.1 Agents and Mobile Agent Systems*

An *agent* is a software entity with a well-known identity, state and behavior with autonomy to somehow represent its user. The agent's user might be a human or an organization (enterprise, community, etc.).

From a more technical point of view, an agent can be implemented as an active object of medium granularity. This means that an agent is an instance of some defined class – with its own group of threads, state and code – identified by a unique global identity.

From a higher-level, conceptual perspective, the agent is a basic but powerful concept to think about and to design complex, distributed, dynamic applications as those enabled by the Internet.

From yet another perspective – the human-computer interaction perspective – agents may be viewed as a new interface paradigm to help the end-user to access this new class of Internet applications, including electronic commerce applications. In this world, the end-users change the way they interact with the computer, from direct manipulation (e.g., word processors, web browsers, and so on) to indirect management (e.g., Web information search). Using agents, users can delegate a set of tasks to be done by agents, instead of doing these tasks themselves. This new paradigm is especially attractive to help the users in complex, tedious or repetitive tasks and in open, dynamic, vast and unstructured information sources such as those found on the Internet.

## *2.2 Agent-based Applications*

We define an *agent-based application* (or ABA for short) as a dynamic, potentially large-scale distributed application in an open and heterogeneous context such as the Internet. The basic conceptual unit for designing and building ABAs is the agent as defined above.

The notion of ABA is quite novel by itself. An ABA is not a typical application that is owned and managed by some person or some organization. Instead, an ABA is best understood as a web of agents each owned and managed by a number of entities with different (and possibly conflicting) goals and attitudes, hosted in different computing platforms, such as workstations and mobile phones.

# 3 AgentSpace Overview

AgentSpace's main goals are the support, development and management of ABAs as described in Section 2. AgentSpace involves the management of several related objects, such as: contexts, places, agents, users, groups of users, permissions, ACLs (access control lists), security managers, tickets, messages, and identities.

The AgentSpace's goals are provided through three separated but well-integrated components as depicted in Figure 2:

- The *AgentSpace server* (*AS-Server*) is a Java multithreaded process in which agents can be executed. The AS-Server provides several services, namely: (1) agent and place creation; (2) agent execution; (3) access control; (4) agent persistency; (5) agent mobility; v) generation of unique identities (UID); (6) support for agent communication; and (7) optionally a simple interface to manage/monitor itself.
- The *AgentSpace client* (*AS-Client*) supports – depending on the corresponding user access level – the management and monitoring of agents and related resources. The AS-Client is a set of Java applets stored on an AS-Server's machine in order to provide an adequate integration with the Web, offering Internet users the possibility to easily manage their own agents remotely. Furthermore, the AS-Client should be able to access several AS-Servers, providing a convenient trade-off between integration and independence between these two components.
- The *AgentSpace application programming interface* (*AS-API*) is a package of Java interfaces and classes that defines the rules to build agents. In particular, the AS-API supports the programmer when building: (1) agent classes and their instances (agents) that are created and stored in the AS-Server's database for later use; and (2) client applets (that are stored in the AS-Server's file system or in the AS-Server's database) in order to provide an interface to agents.

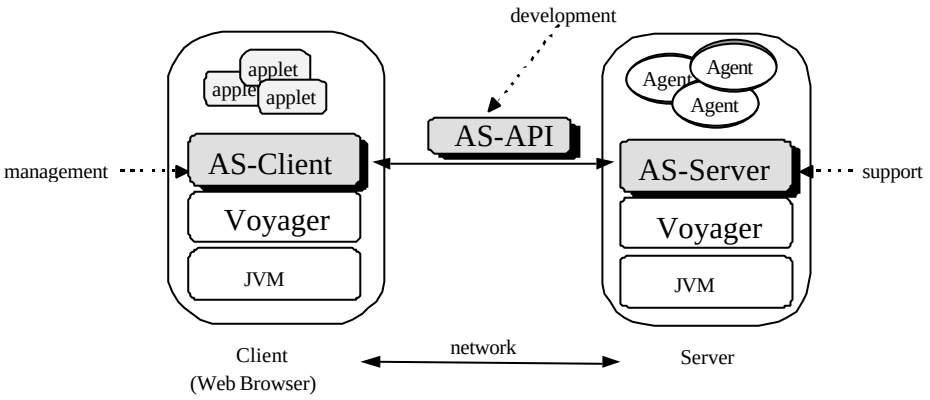


*Figure 2: AgentSpace main components.*

These clients/applets can be either generic mini-applications – such as the AS-Client itself, see above – or specific to some particular agent, for example, to input data or to present a report.

Both server and client AgentSpace components run on top of Voyager and Java Virtual Machine (JVM). They can execute in the same or in different machines. As depicted in Figure 2, the agents run always on some AS-Server's context. On the other hand, they interact with their end-users through (specific or generic) applets running in some Web browser's context.

Voyager [Obj97] from ObjectSpace, is a powerful middle-ware, that is, a communication infrastructure between Java applications much more powerful and complete than Sun's RMI. Voyager has also revealed itself to be very adequate to the majority of our needs, e.g. high-level performance, dynamic stub generation and small system objects.

# 4 The ILC Application

Figure 3 depicts the agent-based proposal of an open, dynamic and large-scale distributed Digital Library. There are several concerns when compared with the traditional schema depicted in Figure 1.

- First, library-agents should embed the functionality and data structures manage already by legacy services.
- Second, in order to handle the management of meta-libraries and related exchanged information/knowledge, a specific agent should be defined – the "library broker agent", or "broker agent" for short. This agent is responsible by providing interlibrary (-agent) cooperation; to define and manage all meta-information involved, namely of documents, libraries and eventually end-users. The broker agent may be managed directly by some librarian, or a restricted set of designated librarians. Nevertheless, it should be accessed indirectly by all other participants (other librarians and lecturers) through their respective specific agents.
- Third, end-users don't have to interact with a disparate number of different library services (this is why they are happy in the figure). Instead, they just interact with their respective lecturer-agents, which then should interact, in some specific agent-communication language, with other agents (e.g., library agents and broker agents).

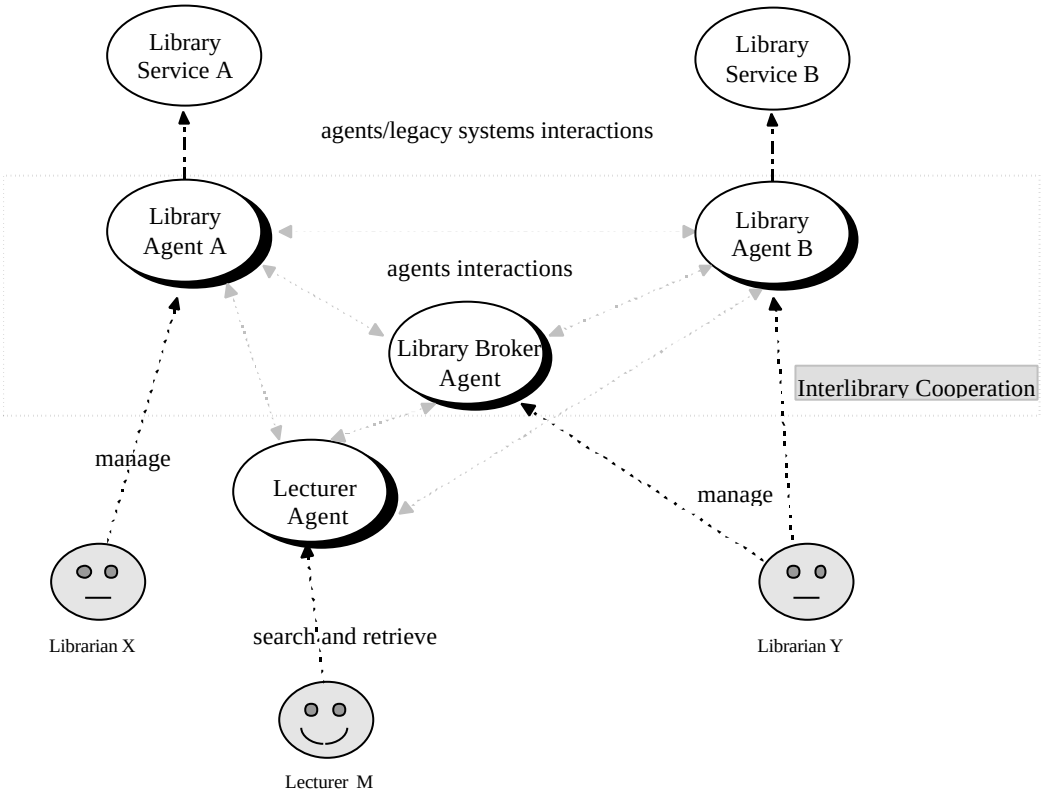


*Figure 3: The agent-based vision of the Digital Library – a simplified and generic schema.*

ILC application has a number of characteristics and requirements that have been dealt with independently in the past. It is their combination that poses problems. It should be:

- *Autonomous*: Each library creates and maintains their own agents using their own resources and/or using resources from others.
- *Heterogeneous*: Each library has bought, got used to and used different interfaces, machine architectures, programming languages, database systems, communication packages, operating systems and so on.

- *Open*: Some agents may depend on other agents and applications, even from external organizations. This means that agents will have to inter-operate with other (legacy) information systems (applications, databases and so on).
- *Dynamic*: Agents will be added, updated and removed at any time without previous notice. They will have to cope with unavailability, new interfaces, oscillating bandwidths, and other variable characteristics.
- *Robust*: Agents will have to tolerate different kinds of failures on machines, networks or at any level of software. For example, agents cannot stop executing just because a company is rebooting their gateway to the outside world.
- *Secure*: The overall system should provide different levels of security depending on each particular part of the whole application. There will be public, place-specific and administrative access control lists.

All these characteristics make ILC potentially very difficult to implement and use. However, we believe agent support systems, and in particular AgentSpace, will help developers build and manage them.

The agent-based approach, comparatively with the traditional Internet/Web approaches [SMD97b] presents the following advantages:

- *Decentralized and scalable*: It is *decentralized* because the only centralized point is the broker agent and *scalable* since this congestion point can always be attenuated or even eliminated by the incremental introduction of more brokers.
- *Dynamic*: It is *dynamic* because all agents can, with more or less flexibility, enter or leave the system. Additionally, the functionalities and complexity of the various agents may evolve independently.
- *Autonomy*: It promotes the *autonomy* and *flexibility* of each library because they now become responsible for the development and maintenance of their own applications and particularly of its own agents.
- *Indirect interaction*: It offers a *single interface* because it hides the disparate complexities and user-interfaces of each application. As a consequence, the end-user only needs to interact directly with its own customized agent and eventually a few specific (management) agents.

Nevertheless, this approach also raises some problems/challenges that are referred to in the following points:

- In dynamic and open environments such as those we have discussed, *it is difficult to define and promote common agent-based application protocols and APIs*. All or the main involved libraries need to agree on a *common agent protocol* to allow agents to communicate. However, this protocol is specific to this application and so does not need to comply with any existing standard – usually, a restricted number of pioneer libraries should design and agree a common protocol; then all other libraries and related organizations just should accept and adopt it.
- Also, due to the non-existence of experience, models, techniques, tools and environments there is nowadays a *great difficulty in designing and building* really agent-based applications.
- *End-users are afraid* to use or to delegate tasks on their software agents; *insecurity*; and *new business models* that require new experiences, new education, and new human-machine interaction paradigm.
- Finally, several research and commercial agent systems are emerging, such as the AgentSpace that we have overviewed, each with its own *proprietary agent model.* However, we expect that the MAF (Mobile Agent Facility) specification, in the OMG scope, solve, at some level, this interoperation issue.

# 6 Conclusions

In this paper we argue why the agent-based approach may be adequate to support open, dynamic and large-scale distributed applications such as the InterLibrary Cooperation domain.

Due to space limitations we have not the opportunity to show why the AgentSpace framework is suitable to support ILC. Nevertheless it is important to refer some of the most important aspects.

Firstly, there is no use or explicit reference to network-enabled classes (like virtual objects in the Voyager framework, or like stubs and skeletons classes in RMI) in the process of creating agents (as well as places). Secondly, all agents are created through the factory method in a transparent, clean and easy process. Additionally AgentSpace provides a very extensible and elegant way to handle security policies/strategies related to the access and interactions between agent and end-users, and between agents themselves. Basically, one security strategy (i.e., a security strategy class) is attached to the agent (or place) object just in the moment of its creation. An other novel aspect of AgentSpace is the well-integrated association between users and agents/places. This mechanism, intrinsic by default in AgentSpace, provides an easy and clean way to develop and manage this class of applications.

The AgentSpace first version is now ready to be used by other programmers. Amongst other applications, AgentSpace is being used to implement a generic "Agent Manager Tool" (with a graphical user interface, the AS-Client), and a prototypical electronic payment system based on SET [RM98,RMS98]. In the future we will use AgentSpace in the COSMOS project [COS97] that will build a framework to negotiate contracts on the Internet, and attempt other experiments, such as for dynamic updating of components. We hope these applications will provide many criticisms and feedback to further improve AgentSpace.

We have also made the implementation available for download at the following address: http://berlin.inesc.pt/~agentspc/.

## References


[BTV96] J. Baumann, C. Tschudin, J. Vitek, editors. *Proceedings of the 2nd ECOOP Workshop on Mobile Object Systems (Linz, Austria)*. Dpunkt, 1996.

[COS97] Ponton, COGEFO/CEFRIEL, Hamburg University, INESC, Interzone Music Publishing, Oracle UK, and SIA. *COSMOS – Common Open Service Market for SMEs*. ESPRIT Research Project Proposal. 1997. Starting 1st April 1998.

[GK95] M. Genesereth, S. Ketchpel. *Software Agents*. Standford University, 1995. `http://logic.standford.edu/sharing/papers/agents.ps`

[Obj97] ObjectSpace. *Voyager Core Package Technical Overview.* 1997.

[RM98] A. Romão, M. Mira da Silva. An Agent-Based Secure Internet Payment System for Mobile Computing. Accepted to the *International Conference on Electronic Commerce'98*. Hamburg, Germany. 1998.

[RMS98] A. Romão, M. Mira da Silva, A. Silva. Prototype Implementation of an Internet Payment System with Mobile Agents. Submitted to the *Mobile Agents'98*. Stuttgart, Germany. 1998..

[SMD97a] A. Silva, M. Mira da Silva, J. Delgado. Motivation and Requirements for AgentSpace: A Framework for Developing Agent Programming Systems. In *Proceedings of the Fourth International Conference on Intelligence in Services and Networks* (IS&N'97). Cernobbio, Italy, 1997.

[SMD97b] Alberto Silva, Miguel Mira da Silva, José Delgado. A Survey of Web Information Systems. In *Proceedings of the Conferência WebNet'97 – World Conference of the WWW, Internet & Intranet.* Toronto, Canada. 1997.