Autonomous Agents and Multi-Agent Systems, 5, 103–111, 2002
© 2002 Kluwer Academic Publishers. Manufactured in The Netherlands.

M&M's: Building Binary Software Components for Supporting Mobile-Agent Enabled Applications

PAULO MARQUES, LUIS SILVA AND JOÃO GABRIEL SILVA {pmarques, luis, jgabriel}@dei.uc.pt

CISUC, Computer Science Department, University of Coimbra, 3030 Coimbra, Portugal

Abstract. Mobile Agents provide a new promising paradigm for developing distributed applications. Nevertheless, although the basic concept has been around for some years and many agent platforms are available both from the industry and research community, there are currently few examples where the technology has been deployed in the real world. One important reason for this is that using the current available agent frameworks it is quite difficult to develop applications without having to center them completely on the agents and on the agent infrastructure. In this paper, we present the M&M project, taking place at the University of Coimbra. In this project, we are developing an extensive component-based framework that enables ordinary applications to use mobile agents in a flexible and easy way. By using this approach, applications can be developed using current object-oriented approaches and become able of sending and receiving agents by the simple drag-and-drop of mobility components. The framework was implemented using the JavaBeans component model and provides integration with ActiveX, which allows applications to be written in a wide variety of programming languages. By using this framework, the development of applications that can make use of mobile regents is greatly simplified, which can contribute to a wider spreading of the mobile agent technology.

Keywords: mobile agents, component-based development, JavaBeans

1. Introduction

Mobile Agents (MAs) provide a new promising paradigm for developing distributed applications. Nevertheless, although the basic concept has been around for some years, and many agent platforms are available both from the industry and research community, there are currently few examples where the technology has been deployed in the real world.

We believe that one important factor preventing the widespread of mobile agents is the lack of proper support for application development. Research has been mainly focused on the agent technology and on mobility issues rather than on the support needed for real-world application development. This issue has two aspects: the programmer and the user.

1.1. The programmer

From the point of view of the programmer, constructing an application that uses mobile agents is a difficult process. Current mobile agents systems force the

development to be centered on the agents, many times requiring the applications themselves to be coded as a special type of agents—*stationary agents*. When this does not happen, special interface agents (*service agents*) have to be setup between the application and the incoming agents. These agents must know how to speak with the mobile agents and with the application. Although the mobile agent concept—*a thread that can move to another node*, is a very useful structuring primitive, all the currently required setup to use it is an overkill that prevents acceptance by the developers.

Since basically anything that can be done with mobile agents can be done using simple client/server remote method evocations, the reasoning goes: "Mobile agents do not give me any fundamentally different (and needed) mechanism, and at the same time force me to develop systems in a completely different way. Why should I bother?"

The problems include: the mobile agent concept is not readily available at the language level; the applications have to be centered on the mobile agents; and a complicated interface between the agents and the applications must be written. The programmers want to develop applications as they currently do. Agents will typically play a small part on the application (90-10 rule: 90% traditional development, 10% mobile agents). Current systems force exactly the opposite.

1.2. The user

From the viewpoint of the user, if an application will make use of mobile agents, it is necessary to first install an agent platform. The security permissions given to the incoming agents must also be configured and the proper hooks necessary to allow the communication between the agents and the application must also be setup. While some of these tasks can be automated using installation scripts, this entire setup package is too much of a burden for the average user. Usually, the user is not concerned with mobile agents nor wants to configure and manage mobile agent platforms. The user is much more concerned with the applications than with the middleware they are using in the background. In the currently available mobile agent systems, the agents are central and widely visible. They are not the background middleware but the foreground applications.

The term mobile code also has very strong negative connotations that make the dissemination of the MA technology difficult. The user is afraid of installing a platform capable of receiving and executing code without his permission. This happens even though the existence of mobile code is present in technologies like Java, in particular in RMI and JINI. The fundamental difference is that in those cases, the user is shielded from the middleware being used. In many cases, using mobile agents do not pose an increased security threat, especially if proper authentication and authorization mechanisms are in place. However, because the current agent platforms do not shield the user from the middleware, the risk associated with this technology is perceived as being higher, which causes users to back away from applications that use it.

In the M&M project, at the University of Coimbra, we have developed an extensive component-based framework for easily supporting the mobile-agent concept when creating applications. In our approach there is no agent platform. The applications become capable of sending, receiving and interacting with mobile agents by using well-defined binary software components [l7]. If a Visual Development Environment is used, this can be accomplished by the simple drag-and-drop of components from a component palette and by visually configuring their properties and interconnections. This is a step forward over the traditional development approaches used in the platform-based MA systems. Because in this approach there is no mobile-agent platform and because the emphasis is put on the applications and not on the agents, we call this approach *ACMAS—Application-Centric Mobile Agent Systems.*

The main goals of the project are twofold: (a) to address the problems that inherently make the development and deployment of mobile-agent based applications difficult; (b) to investigate the tradeoffs associated to building mobile-agent enabled applications using a component-based approach.

The rest of this paper is organized as follows. Section 2 discusses the ACMAS approach being explored in M&M. Section 3 presents the application areas being addressed to validate the approach. Section 4 discusses some lessons learned. Section 5 presents related work. Finally, Section 6 gives the conclusion of the paper.

2. The M&M project

2.1. The ACMAS approach

In the ACMAS approach, the applications are developed using industry object-oriented development best practices and can additionally become agent-enabled by incorporating the mobility components. In this approach, the emphasis is put on the development of applications and not on the agents. Each agent arrives and departs of the application that it is specific. The application knows the interface of the agents and the agents know how to interact with the applications (Figure 1).

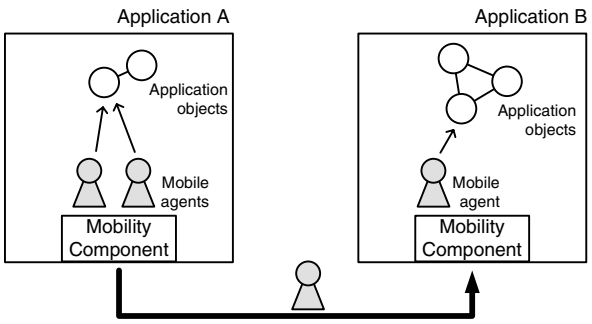

Figure 1. Applications become agent-enabled by using mobility components.

From the perspective of the programmer, all he has to do is to use the mobility components and write the agents. The agents arrive and departure directly from the application without the needing a fully blown agent platform. From the user point of view, he just sees an ordinary application. The user is completely shield from the usage of mobile agents taking place in the application.

In ACMAS, the applications are developed using three different kinds of components (Figure 2):

- Mobile-agent support components;
- Third-party off-the-shelf components;
- Domain specific components.

Mobile-agent support components provide the basic needs in terms of mobile-agent infrastructure. We currently have components for supporting the migration of agents between applications, components for supporting different inter-agent communication mechanisms, components for agent tracking, security and others.

Third-party off-the-shelf components are components that are commercially available from software makers and can be used for building the system. Currently there

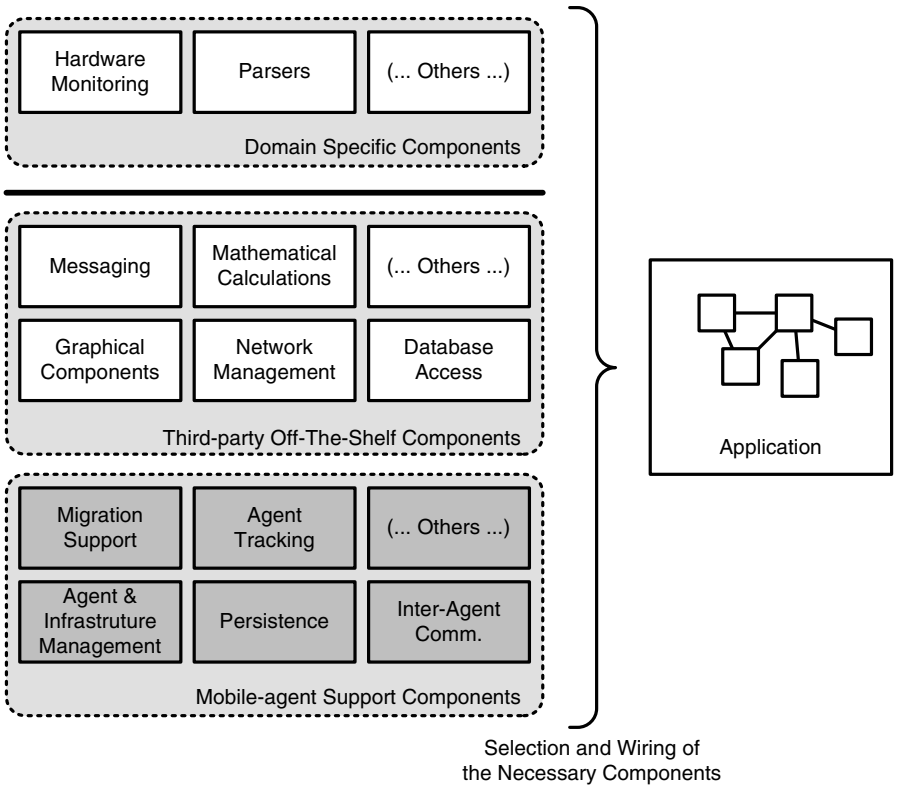

Figure 2. The applications are developed by wiring different kinds of software components.

is a large variety of components available for the most different things, like accessing databases, designing graphical user interfaces, messaging and others. All these components can be used for building the applications without having to re-implement the required functionalities.

Domain specific components are modules that must be written in the context of the application domain being considered, providing functionalities not readily available off-the-shelf. For instance, while implementing a particular application it may be necessary to write special parsers for extracting information from files, or to write supporting services for allowing agents to monitor the hardware of a machine. These modules can be coded as components and incorporated into the application.

One important point in ACMAS is that while developing an application only the components required for that particular application domain have to be included. Also, because the features available to the programmer are implemented in separate binary components, which have well-defined boundaries, it is possible to expand the package without influencing already developed applications. Each time a new feature is required or a new service implemented, this can be done by creating a new component. This allows a high degree of flexibility, since the component palette is constantly being enriched with new components. At the same time, the new features do not force the applications to become heavier or bulkier since only required functionalities are introduced in each application.

2.2. Supporting ActiveX

The component package was implemented in the Java programming language and using the JavaBeans [15] component model. Nevertheless, we have also decided to support the ActiveX component model, that is based on Microsoft's COM [11]. We have taken this decision because of the wide adoption of this component model in the software industry and in research. By supporting ActiveX, a much wider usage base and a much richer set of environments where to experiment with the framework is available.

Another important point is that any language that has the necessary mechanisms to make use of COM and ActiveX is able to use the mobility components. At the present, most languages that available for the Windows platform [9] have this capability. By supporting ActiveX, the developer is no longer limited to write applications in Java. Applications can be coded in languages like Visual C++ [4], Visual Basic [2] or Delphi [18]. This happens without having to implement any special layer between the components and the target languages. During our experiments, a simple instant messaging application was written in Visual C++, Visual Basic and Java. The agents migrated and executed in all the client applications independently of the language where they were written.

3. Application domains being explored

In order to evaluate the strengths and weaknesses of the ACMAS approach, when compared in the traditional platform-based model, two application domains were

selected:

- Accessing Information Systems in Disconnected-Computing Environments;
- Network Management.

3.1. Accessing information systems in disconnected-computing environments

Over the last few years, mobile phones, laptops and personal digital assistants (PDAs) have become commonplace. Laptops and mobile phones are already reshaping the way people work. As mobile devices become more powerful, users are starting to expect to have access to information in any place they are, by using such devices.

To complicate maters, today's Corporate Information Systems (CIS) are being deployed using a three-tier architecture [10]. Thus, accessing the databases is no longer enough. It is becoming increasingly important to have mechanisms that allow mobile end-devices to access and interact with the business logic present in the middle-tier.

Mobile agents are a very interesting approach to software development in disconnected computing environments [3, 12]. The advantages of using mobile agents in mobile computing include:

- Connections must only be up for receiving and sending the agents.
- Data must not be transferred from the server to the client: the agents just process it at the server.
- Multiple interactions occur locally at the server, between the agents and server processes.

We are currently exploring the ACMAS approach for building systems where mobile agents provide the base mechanisms for allowing client applications to interact with the business logic present on the information systems [6]. ACMAS is especially interesting in this context since the requirements at the client-side are very different from the requirements at the server-side. ACMAS provide a very flexible way of addressing these different requirements by having different components deployed in the applications of the end-devices and on the server.

3.2. Network management

Network Management has always been one of the most privileged demonstration fields for MA technology. In fact, it was during a previous project [13] on this area that we have identified many of previously mentioned problems of the platform-based approach.

While the previous application domain gives us the opportunity to experiment directly at the application level, for network management we are building a

domain-specific component palette that gives applications and agents the services that we found to be the most interesting on this application field:

- Multi-level delegation of management support.
- Distribution of management services across unstructured topologies.
- Disconnected or very low bandwidth operation.
- Dynamic service deployment, reconfiguration and relocation.
- On-the-fly extension of installed management services.
- Heuristic data collection over large network domains.

Working on a component palette for the specific domain of network management is giving us the opportunity to assess the limitations and strengths of the implemented extensibility mechanism for supporting new services [7], and it is also allowing us to evaluate ACMAS in the context of building non-hierarchical management meshes supported by mobile agents.

4. Lessons learned

From building the ACMAS component palette and developing some prototype applications, we have already gathered some important lessons.

When developing applications based on mobile agents, it is a lot more easy to use components to agent-enhance the applications than to center all the development around agents, where complicated setups have to be done. This is especially important if it is necessary to use other middleware like CORBA [10] or SNMP [14]. While current MA frameworks do not integrate well with existing middleware, applications using the mobility components can transparently use other middleware solutions.

After presenting some demonstrations of applications that are mobile-agent enabled, the reaction of the users was very positive. One key point for this was that they were not aware of the mobile agents but only of the results obtained from the applications. This clearly contrasts with our experience on presenting applications based on classical platform-based systems, where the use of agent-technology typically raised concerns and some suspicion.

Developing component for supporting mobile agents can be hard. When it comes to security, it is not trivial to design an approach where the mobility components do not impose restrictions on the application. In addition, when developing distributed network components (e.g., agent tracking support), managing the configuration of the components becomes complicated since there is no central point to address. This typically requires that the configuration must be replicated on the existing components. Technologies like Sun's InfoBus [15] may help to solve some of these problems.

For a more complete account of the experiences we had while implementing the framework, please refer to [6–8].

5. Related work

The two works most related to ours are JIAC [1] from TU-Berlin and Gypsy [5] from TU-Vienna.

JIAC is a component toolkit for building intelligent agent systems for telecommunication applications. In JIAC components are scripts that can be plug in into an agent backbone or into a place.

Gypsy is a component-oriented mobile agent system, where everything is a JavaBean component. Agents are components that run inside of places. Places can be assembled by connecting several components and host the agents.

While on these approaches, the main objective is to develop mobile agents and agent platforms by combining different components, in ACMAS the key idea is to embed sufficient support for agent mobility inside of the applications, bringing the focus back to the application development.

6. Conclusion

In this paper, we have presented the M&M project, which aims to address some of the questions that make the usage of mobile agent technology difficult. In our approach, the applications become agent-enabled by means of software components that allow them to receive and send agents and provide the necessary services.

We believe that this approach will contribute to a easier dissemination of the mobile agent paradigm because:

- The applications can be developed using traditional OO techniques.
- The applications become agent-enable by the simple drag-and-drop of components in Visual Development Environments.
- The final user does not have to deal with agent platforms but only with applications.
- Integration with ActiveX provides a wide usage platform on where to use mobile agents.

In addition, using a component-based approach brings several benefits when compared to a platform-based approach:

- Agents only migrate to the applications they are interested in, and not to a platform that runs all agents for all the applications. This contributes for having a much more lightweight and robust system.
- By creating components that represent new services, the framework can be easly extended.
- Mobile agents are just like any other middleware and coexists pacifically with other programming tools. This is specially important when supporting legacy applications.

After having developed a comprehensive component pallete, we are presently working on validating the approach by experimenting in two application domains:

information systems in disconnected computing environments and network management. Applying the ACMAS technology on these areas will certainly provide a valuable learning experience.

Acknowledgments

This investigation was partially supported by the Portuguese Research Agency FCT, through the program PRAXIS XXI (scholarship number DB/18353/98) and the M&M Project (ref. POSI/33596/CHS 1999), and through CISUC (R&D Unit 326/97).

References

1. S. Albayrak and D. Wieczcorek, "JIAC—A toolkit for telecommunication applications," in *Proc. Intelligent Agents for Telecommunication Applications Workshop* (IATA'99), Stockholm, Sweden, 1999.
2. F. Balena, *Programming Visual Basic 6.0*, Microsoft Press, 1999.
3. D. Kotz, R. Gray, S. Nog, D. Rus, S. Chawla, and G. Cybenko, "AGENT TCL: Targeting the needs of mobile computers," *IEEE Internet Computing* vol. 1, no. 4, pp. 58–67, 1997.
4. D. Kruglinski, S. Wingo, and G. Shepherd, *Programming Visual C++*, Fifth ed., Microsoft Press, 1998.
5. W. Lugmayr, "Gypsy: A component-based mobile agent system," in *Proc. 8th Euromicro Workshop on Parallel and Distributed Processing* (PDP2000), Rhodos, Greece, 2000.
6. P. Marques, L. Silva, and J. Silva, "A flexible mobile agent framework for acessing information systems in disconnected computing environments," in *Proc. Third Int. Workshop on Mobility in Databases and Distributed Systems MDDS'2000*, Greenwich, UK, September 2000, to appear.
7. P. Marques, L. Silva, and J. Silva, "Addressing the question of platform extensibility in mobile agent systems," in *Proc. Int. ICSC Symp. on Multi-Agents and Mobile Agents in Virtual Organizations and E-Commerce (MAMA'2000)*, Wollongong, Australia, December 2000, to appear.
8. P. Marques, L. Silva, and J. Silva, "Building domain-specific mobile-agent platforms from reusable software component," in *Proc. IEEE 2000 Int. Conf. on Software, Telecommunications and Computer Networks (SoftCom'2000)*, Split and Dubrovnik (Croatia), Trieste and Venice (Italy), October 2000, to appear.
9. Microsoft Corporation, *Microsoft Windows Products Homepage*, http://www.microsoft.com/windows/default.asp.
10. R. Orfali, D. Harkey, J. Edwards, and R. Crfali, *Instant CORBA*, Wiley: New York, 1997.
11. D. Rogerson, *Inside COM*, Microsoft Press, 1996.
12. A. Sahai and C. Morin, "Mobile agents for enabling mobile user aware applications," in *Proc. Autonomous Agents 98'*, Minneapolis, USA, 1998.
13. L. Silva, P. Simoes, G. Soares, P. Martins, V. Batista, C. Renato, L. Almeida, and N. Stohr, "JAMES: A platform of mobile agents for the management of telecommunication networks," in *Proc. Intelligent Agents for Telecommunication Applications Workshop (IATA'99)*, Stockholm, 1999.
14. W. Stallings, *SNMP, SNMPv2, and CMIP*, Addison-Wesley: Reading, MA, 1993.
15. Sun Microsystems, *JavaBeans Specification 1.01*, Sun Microsystems, 1997, available at http://www.javasoft.com/beans.
16. Sun Microsystems, *InfoBus 1.2 Specification*, Sun Microsystems, 1999, available at http://www.javasoft.beans/infobus.
17. C. Szyperski, "Component software," *Beyond Object-Oriented Programming*, Addison-Wesely: Reading, MA, 1998.
18. S. Teixeira and X. Pacheco, *Delphi 5 Developer's Guide*, Sams, 1999.