

Aglets: Enabling the Virtual Enterprise

Published and Presented at ME-SELA '97.

P.E.Clements, T.Papaioannou, J.Edwards

ISBN 1 86058 066 1, p425

SYNOPSIS

The flexible integration of a range of disparate IT applications is a key requirement for modern global enterprises. The virtual enterprise, formed by a collection of collaborating companies for short term, high return, one off projects provides perhaps the most extreme example of this need. This paper proposes the use of mobile agent technology to create IT systems that can support this required system agility.

The paper describes agent technology and a particular implementation of mobile agents, IBM's Aglets Workbench. This is followed by a description of an application of the Aglet system based on a manufacturing enterprise that spans Europe, the USA and the Far East.

The paper concludes that the features of local interaction, reduced network loading, server flexibility and application autonomy which are supported by mobile agent technology all help to provide a level of agility above that provided by more conventional IT technology.

KEYWORDS

Agent Technology, Component Frameworks, Aglets, Systems Integration, Virtual Enterprise

1. INTRODUCTION

Recent developments in global commerce have witnessed the advent of the virtual enterprise, where groups of companies are forming short-term relationships to collaborate on one-off projects. These enterprises provide a clear example of the requirement for IT application agility. In these companies, high return is based on a quick kill, through meeting a short-term market need. Speed in setting up rigorous IT supported processes is of the essence. However, the arguments provided in this paper are equally applicable to any IT supported enterprise that has a requirement to modify its IT system to support required changes in its business processes, which implies any business that wants to survive in the global market.

Bloch and Pigneur(1) in their work on virtual enterprises state that “usage of inter-organisational systems will grow because of the need to *integrate disparate* organisations or individuals in the same IT-enabled processes, *independently of formal boundaries*”.

Creating these inter-organisational IT systems is one of the hurdles faced by virtual enterprises on start-up, the key problem being the interoperation of their respective software

applications. In this paper the term interoperation is used to imply that operations in separate enterprises need to be integrated in such a way that they support the realisation of necessary business processes. It is evident therefore that interoperation between enterprises will require use of common structures, mechanisms and controls.

Sims(2) states that "Interoperability and integration of two applications that have different architectures, designs and application interfaces can take several months and maybe years to accomplish". In the life of the virtual enterprise this time scale is unacceptable and if virtual enterprises are to become common, solutions to this particular problem have to be found.

This paper proposes a particular agent-based approach to the solution of this problem, which builds upon more general work being undertaken by researchers at MSI investigating how next generation manufacturing software systems will be developed, built and deployed(3). The paper describes agent technology and mobile agent technology identifying the features that might help to provide system agility. This general description is followed by an overview of experimental work based on a particular mobile agent implementation. The experimental work is to investigate and test the suitability of utilising mobile agents in a distributed, multi-platform environment to produce an order-processing solution for a globally distributed enterprise. The industrial scenario is based on information gathered from a vacuum component manufacturer whose headquarters are located in the UK.

2. AGENT TECHNOLOGY

2.1 What is an Agent?

'The term 'agent' has been picked up, widely appropriated, and in many cases misappropriated, by technical publications, lay publications, and many researchers in computer science.'(4).

The question of what actually constitutes an agent, and how they differ from a normal program, has been heavily debated for several years now(5). Agents can be loosely defined as *'software that assist people on their behalf, ...and are delegated to perform task(s), and given constraints under which they can operate'* (6). However, agents come in a myriad of different types, usually depending on the nature of their environment. What has been needed is a classification scheme to distinguish between different types of agents. This paper does not propose a further definition of what an agent is, but will adopt the Franklin and Graesser(7) classification scheme and categorise the agents dealt with here as goal-oriented, communicative, mobile agents.

Franklin and Graesser definitions are as follows :

- 1) *goal oriented agents* - agents that do not simply act in response to the environment;
- 2) *communicative agents* - those able to communicate with other agents;
- 3) *mobile agents* - those able to transport themselves from one host to another.

In order for these agents to exist within a system or to themselves form a system they require a framework for implementation and execution. This is known as the agent environment.

2.2 Agent Environments

There are a large number of agent building packages on the market that allow users to attempt to build and manage their own agents and agent systems. First, and probably foremost is the Tabriz AgentWare package from General Magic(8), which executes and manages agent-based applications on servers, and Tabriz Agent Tools for creating agent applications deployable on Web sites.

The Java Agent Template (JAT) architecture provides a set of packages that facilitate building multi-agent systems using the Java programming language. The recent release of the JATLite(9) package has gone some way to simplifying the process, by providing different 'layers' from which to abstract, thus offering a flexible base on which to build.

IBM have developed two packages – the Agent Building Environment (ABE), and the Aglets Workbench. The ABE is a toolkit for software developers that makes it easy to build an application based on intelligent agents or to add agents to an existing application. Once again the package includes a number of pre-built components from which it is possible to add agent technology to applications.

The Aglets Workbench, developed at IBM's research labs in Japan, is aimed at producing stand-alone mobile agents. The complete package offers a graphical environment for building mobile agent applications in Java, an agent server, and the specification for an Agent Transfer Protocol (ATP). The experimental work discussed later has been achieved through use of the Aglets Workbench.

3. THE AGLETS WORKBENCH

A key characteristic when considering agent technology as an aid to enabling the virtual enterprise is *mobility*. This implies that agents can move around a network according to some itinerary, i.e. the route the aglet will follow when instantiated. The Aglets Workbench supports both mobility and itinerary. The structure of the Aglets can be considered to consist of two distinct parts, the aglet core and the aglet proxy. The core is the heart of the aglet and contains all of the aglet's internal variables and methods. It provides interfaces through which the aglet may communicate with its environment. The aglet core is then encapsulated by an aglet proxy that acts as a shield against any attempt to directly access any of the private methods and variables, and can also hide the real location of the aglet from malicious aglets¹. An aglet also contains a unique identifier and an itinerary (if it needs one).

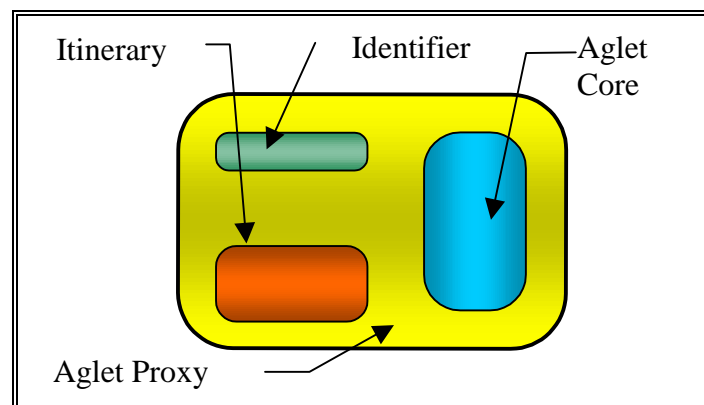


Figure 1. Structure of an Aglet

Whilst there are a number of other agent building environments that support the notion of mobility, the Aglets Workbench was chosen as the agent framework from which the experimental agents within our system were created. The principle reasons for choosing this framework were:

- it was one of the first to use the Java programming language;
- it contains the notion of agent itinerary which systems such as Telescript did not support;

¹ A malicious aglet is one that may attempt to take control of another aglet, or access its private internal data.

- it is being proposed for submission to the Object Management Group (OMG) Mobile Agent Facility RFP;
- it enforces good security facilities: for an aglet to move to a remote host, the remote host has to be running an aglet server and hence some of the security concerns were solved.

In addition to these implementation issues, mobile agents possess a number of key characteristics that were perceived as features which could support improved agility of enterprise IT systems. It was these characteristics which provided the original reason to experiment with agent technology. The remaining subsections in this section of the paper provide an over view of these key characteristics.

3.1 Server flexibility

This is one of the key differences between traditional message passing systems and systems based on mobile agents. Mobile agents can be allowed to change the behaviour of the server without the prior consent of the owner of the server. In a process dependant system this flexibility can be extremely useful.

Unlike most examples of mobile agents that travel the Internet, aglets can not migrate to host systems that do not have a compatible server, into which they can be inserted. These server objects, known as **contexts**, act as warehouses, or workplaces where aglets of all types can communicate with each other, see Figure 2. A context is a stationary object that provides a means for hosting and managing aglets in an environment that is secure from malicious aglets. Through this context an aglet is able to get information about its environment, and to send and receive messages to that environment and other aglets currently active in its proximity. If an attempt is made by a malicious aglet to by-pass the aglets proxy and access private data, the security manager will override the attempt and stop access.

3.2 Local Interaction

The utilisation of mobile agents can transfer the resource load to a local server. The agent handles any data or transactions it requires at the server, or a very near proxy, and transports *only* the results of its work. This allows for a far more efficient and flexible system if subsequent data requests are dependant on preceding ones, since the agents are able to react immediately to the requested information, without having to wait for lengthy data transfer times. Aglets communicate by passing **messages** between themselves via the context. This process entails the exchange of a message object between two aglets, and allows for both synchronous and asynchronous message passing, thus enabling aglets to collaborate and exchange information in a loose or tightly integrated fashion.

3.3 Reduced Network Load

One of the central notions of the Aglets Workbench is its ability to reduce network load by migrating an aglet to the most appropriate host on which to perform its processing. By performing all data transactions at the server, the mobile agent paradigm can substantially decrease network load compared to a traditional RPC implementation, especially in a large distributed data oriented system. This liberation of resources can itself speed up remaining RPC procedures in a multi-paradigm, distributed system.

Mobility in the Aglets Workbench is enabled by the provision of two facilities:

- 1) the Agent Transfer Protocol (ATP)
- 2) the Java Agent Transfer and Communication Interface (J-ATCI).

The ATP is an application level protocol for distributed agent based information systems and facilitates migration of the aglets over the network. Based on the naming conventions of the Internet, ATP uses the Universal Resource Locator (URL) for specifying host locations, whilst maintaining a platform independent protocol for enabling the transfer of mobile agents between networked computers. Although this protocol has been released with the Aglet Workbench its domain of use is by no means exclusive to aglets, as it offers the opportunity to handle mobile agents from any programming language and a variety of agent systems, as long as they implement the protocol interfaces.

Reinforcing the ATP at a higher communication level is J-ATCI, an independent agent protocol enabling agents to move and communicate within a network. J-ATCI is a simple and flexible programming interface that enables programmers to develop platform independent agents without having to build into them the necessary protocols for wire communication. By ensuring a native implementation of the J-ATCI designers can expect their agents to function on any platform. The J-ATCI has also been submitted to the OMG.

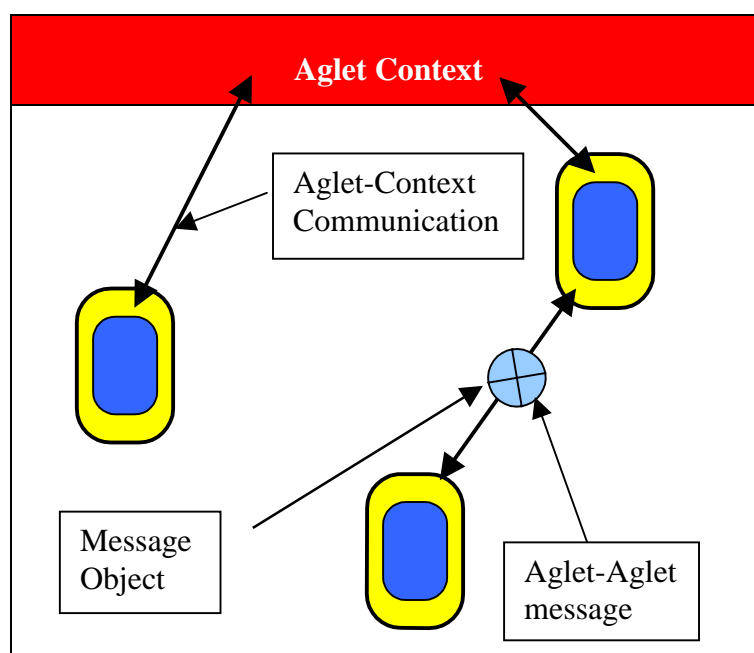


Figure 2. The Aglet Environment

3.4 Autonomy

By encapsulating within the agent the ability to make decisions for itself based on its environment the agent is allowed to continue to carry out its tasks, even when the host from which it was generated is unavailable, disconnected from the network, or even switched off.

Programming an aglet with predefined goals is achieved by implementing the **itinerary** class and associating this with the aglet. The itinerary object is the aglets travel plan, and encapsulates the instructions for the aglet on which hosts to go to, and what actions to perform once at those hosts. The aglet is thus independent of its environment and, depending on how the itinerary is implemented, more importantly autonomous.

4 AN APPLICATION OF AGLET TECHNOLOGY

Integrating an IT infrastructure to support communication between distributed companies is problematic at best. It is unlikely that any partners in a virtual enterprise will have similar networks or software, but the requirement exists for them to interoperate.

This section discusses an experiment to investigate and test the suitability of utilising mobile agents in a distributed, multi-platform environment to produce an order-processing solution for a global enterprise. The scenario is based on a vacuum component manufacturer located in the south of England, which has an extensive web of re-sellers and agents around the world, manufacturing plants in the Far East and the UK, and warehousing in the States and the UK. In this instance, the differing areas of the enterprise have been split into four distinct groups: Sales, Stock Control, Purchasing and Manufacturing. Each of these is modelled as an independent object that has its own internal business logic. In the case of a virtual enterprise these will have already existed within the separate companies as legacy systems, but must be integrated to enable an order to progress from sales, to dispatching.

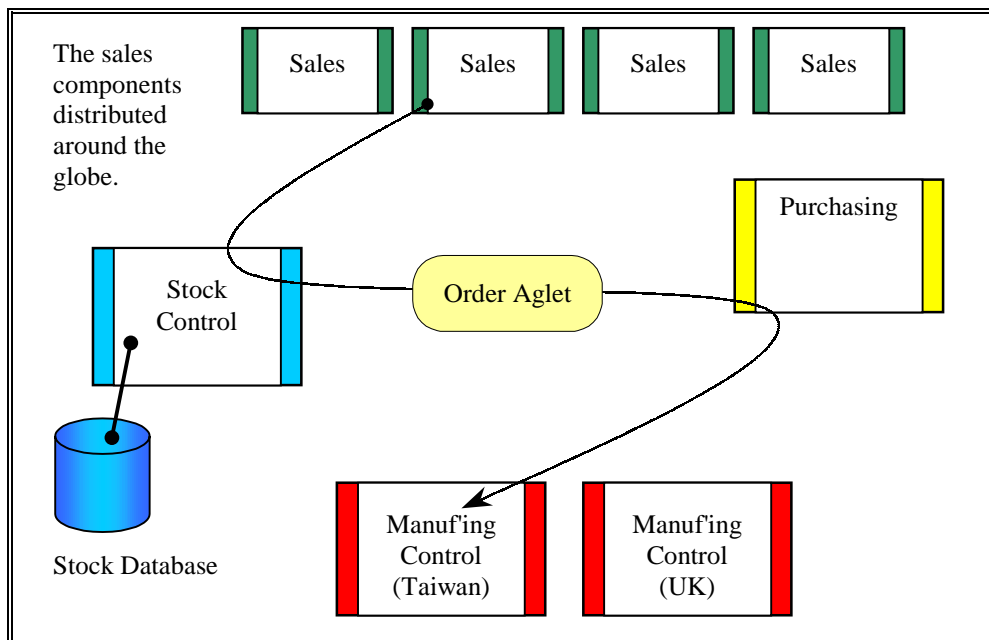


Figure 3. The Virtual Enterprise

The above diagram shows the virtual enterprise and its four functional sections. Each major component is a stationary aglet, with any business logic, or database connection it requires encapsulated within it. In a real world solution, this could be written by the host company, without any need for discourse with the other partners, since its functionality is not dependant on the others components as long as it implements the ATP and J-ATCI protocols. Thus enabling the integration of disparate legacy systems.

When an order is placed the sales components generate a new Order Process (OP) aglet. Encapsulated within the new OP aglet is an Order object and an Itinerary. Once dispatched the aglet is now autonomous, and is responsible for the order until it is resolved. The first visit will always be to the Stock Manager, where it will request the fulfilment of the order. If this can be achieved it returns to the Order Manager and reports on the due date of the order. If there are not enough products in stock, the OP reserves what is available and migrates to

the Purchases Manager, where it requests a Purchase Order to fulfil the rest of its order. On receipt of this data, it will once again migrate to a new host, the Factory Manager, where it will supply details of the required order, and obtain a date for completion of production.

5. CONCLUSIONS

The work described in this paper supports the notion that the addition of the communication layers inherent in the Aglets Workbench to existing legacy systems can provide a framework for improved integration. The flexibility of this approach is demonstrated by the ease of distribution. As demonstrated by the experiment the business logic for any major component can be distributed around any network, a LAN or even on a global scale by utilising the Internet. If a change is required in any component, perhaps triggered by some business change, the new component can be integrated easily through the existence of the Aglet framework.

It is the features of local interaction, reduced network loading, server flexibility, agent autonomy and the communication protocols supported by the IBM Aglet Workbench which have enabled this improved level of enterprise agility.

ACKNOWLEDGEMENTS

The authors thank the EPSRC for their continued support of the work.

REFERENCES

- [1] Bloch, M. and Pigneur, Y. *The Extended Enterprise: a Descriptive Framework, some Enabling Technologies and Case Studies*, Proceedings of the 2nd Int. Conf. Network Organisation Management, June 95.
- [2] Sims, J.E., *Framework for Adaptive Interoperability of Manufacturing Enterprises FAIME – A Case Study*, Proc. Of SPIE, pages 289-303, vol 2913, 1996.
- [3] Clements, P.E., *Requirements for Software Systems Which Enable the Agile Enterprise*, Submitted to ME-SELA'97
- [4] Foner, L.N. (1997) *Entertaining Agents: A Sociological Case Study*, Proceedings of Autonomous Agents '97, tbp
- [5] Petrie, C.J. (1996) *Agent-Based Engineering, the Web, and Intelligence*, IEEE Expert, December 1996
- [6] Gilbert, D. and Janca, P. *IBM White Paper : Intelligent Agents*, <http://www.networking.ibm.com/iag/iagwp1.html>
- [7] Franklin, S and Graesser, A.(1996) *Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents*, Proceedings of the 3rd Int. Workshop on Agent Theories, Architectures, and Languages, Berlin, Springer-Verlag
- [8] General Magic Inc. <http://www.genmagic.com>
- [9] Further information can be found at http://java.stanford.edu/java_agent/html/