

Using Mobile Agents To Improve the Alignment Between Manufacturing and its IT Support Systems.

T.Papaioannou and J.Edwards

Department of Manufacturing Engineering

Loughborough University

Loughborough, Leicestershire, LE11 3TU, UK

Tel: +44 1509 228250 Fax: +44 1509 267725

Email T.Papaioannou@lboro.ac.uk or J.M.Edwards@lboro.ac.uk

Abstract

This paper proposes the notion that Mobile Agent Technology can improve the alignment between IT systems and the real world processes they support. This can aid enterprise agility, particularly where distributed information is a feature, as in the virtual enterprise.

A model of the manufacturing Sales/Order process is proposed. The sales order is shown to be a naturally mobile element of the model. The subsequent decomposition of the agent types during detailed design and implementation reveals an abstract pattern for database query using agent technology. Finally, an overview of the security issues associated with mobile agents is discussed.

Keywords

Mobile Agents , Virtual Enterprise, System Integration, Agile Systems, Mobile Agent Security

1 INTRODUCTION

The ability to effectively manage, manipulate, distribute and access an enterprise's information is key to competitiveness within the global market place [20]. Developments in Information Technology (IT) have provided database systems that help support this need. However, increasing levels of flexibility are being demanded by those companies who compete in the very rapidly changing sectors of the market.

This globalisation and advance in IT systems has led to the emergence of the Virtual Enterprise [2]. This type of enterprise is made up of a number of co-operating companies who are generally physically distributed but work together to meet some market demand. As the operations of the enterprise are distributed geographically, so must the information systems that support them [20]. Typically, these relationships are short-term collaborations on one off, high return projects where time to becoming effective in the market is of the essence. Management of information remains a base requirement of these virtual enterprises, who are therefore charged with the integration of several separate IT systems to form an operational system in as short a time as possible. The creation of IT systems that can support this level of flexibility will be of great advantage to these types of enterprise.

Object and component technology have demonstrated the advantage of trying to reflect natural, real world composition in the architecture of IT Systems. Mobile Agent Technology offers an additional and more natural abstraction that can be used to enhance an IT architecture to produce an improved image of the real world it is trying to simulate. Such an improved image offers a better base from which to make changes in line with required change in the real world [7].

Much has been written on the advantages offered by multi-agent systems in support of enterprise IT [3]. This paper focuses on the benefits to be gained through the use of Mobile Agents. The paper first introduces the traditional distribution mechanisms, before detailing the characteristics of mobile agents that may provide additional benefit when building manufacturing enterprise systems. The implications of Java and its facilities for database connectivity (known as JDBC¹) are discussed, before an agent-based model for supporting the Sales/Order process within a distributed enterprise is proposed. This process is based around the access and manipulation of information generally stored in legacy information sources. The decomposition of the system identifies three key agent types within this model. Subsequent analysis of one of these leads to the derivation of an abstract pattern for database query using agent technology.

Mobile Agent systems have been criticised on grounds of decreased system security. The final section of the paper provides an overview of the security issues associated with mobile agents and a brief discussion on applying these to a working system.

¹ Contrary to popular opinion JDBC is not an acronym for 'Java Database Connectivity', but a name trademarked by Sun Microsystems[1]

2 DISTRIBUTION

The main protagonists offering solutions to the problems of distribution are the Object Management Group (OMG) with their Common Object Request Broker Architecture (CORBA), Microsoft with their DCOM specification and Sun with the Remote Method Invocation (RMI) API. All three examples offer systems that bring the advantage of location transparency to the integration issue, and whilst this is an important requirement for distributed systems, it does have some drawbacks. When two components, A and B, are communicating via an ORB, it appears to A that B is resident on the same systems. However, this may not be the case. Component B could be on a completely different host or network, or with the emergence of the Internet as a global network, the component could be on the other side of the world.

- In general: Traditional distribution mechanisms achieve communication between different objects by 'hiding' the location of these objects from each other. Their messaging systems allow only mobile data to be passed between objects and locations. Thus, current systems can be characterised as using location transparency and mobile data to enable communication between distributed objects.

The important difference between the CORBA, DCOM and RMI is that CORBA is a specification formulated by a consortium. Both RMI and DCOM are controlled by a single vendor although RMI is perhaps more open in its nature. When dealing with distribution these technologies are being challenged by the relative simplicity of Internet technology.

2.1 CORBA

CORBA is the product of a consortium of over eight hundred companies, known as the Object Management Group (OMG). The OMG has approached the problem of handling the interaction of distributed components by creating interface specifications, and *not* code. Distributed components of the system are able to describe their interfaces using the Interface Definition Language (IDL) and subsequently interoperate through the underlying Object Request Broker (ORB). It is the ORB that provides the communication backbone through which all of the components are able to interact. The OMG contend that components communicating via an ORB do not need to be aware of the mechanisms used in that communication, and in fact are able to discover each other at run time allowing extensive flexibility and configurability. However, the reality is that in the currently available commercial ORB's all that can be discovered at runtime are a component's methods and arguments as described by its IDL definition. This is insufficient for the realisation of systems based on loosely coupled components using late binding.

2.2 OLE/DCOM

OLE and the Distributed Component Object Model (DCOM) are Microsoft's version of a software architecture that allows applications to be assembled from binary software components and interoperate across a network. DCOM is the underlying architecture that forms the foundation for higher-level software services normally provided by OLE. OLE is a unified environment of object-based services with the ability to both customise those services and arbitrarily extend the architecture through custom services,

with the overall purpose of enabling integration between components. OLE services span various aspects of commonly needed system functionality, including compound documents, custom controls, inter-application scripting, data transfer, and other software interactions.

2.3 RMI

Java **Remote Method Invocation** (RMI) is a distributed object model for the Java programming language that retains the semantics of the Java object model, making distributed objects easy to implement and to reuse. RMI is a relative newcomer to the distributed object framework world, but combines aspects of existing technologies such as the Modula-2 Network Objects system and Spring's subcontract whilst also including some novel features made possible by the Java language itself.

2.4 Internet Technology

Internet technology has proved to be effective for connecting a mix of different types of computers and computer networks, while also providing location independence to information. Simple, open protocols like TCP/IP, HTTP and HTML have led to the proliferation of diverse information sources and allowed the creation of vast heterogeneous networks, populated by distributed devices that can be both client and server. Further, analysts predict that the next five years will see the evolutionary convergence of the Internet, Intranets and traditional IT models such as client/server and peer to peer [1]. It would thus appear to be an appropriate approach for building integrated, distributed IT systems to support the virtual enterprise where world-wide distribution is an important consideration. However serious problems with this distribution technology remain.

2.5 Distribution Problems

The saturation of network bandwidth, and problems of server availability, particularly when part of the network in question is the Internet, means that remote database access as required by the virtual enterprise model could mean ineffective IT support for the business. As well as data timeliness, factors such as data integrity and security are also a concern when dealing with the Internet.

Mobile Agent technology may be able to overcome this set of problems through its novel characteristics of local interaction [6] and support for disconnected operation.

3 MOBILE AGENT TECHNOLOGY

The notion of Mobile Agents was established in 1994 with the release of a white paper by White [23] that described a computational environment known as "Telescript". In this environment running programs were able to transport themselves from host to host in a computer network. However, the notion of mobile computation had been suggested long before, in "Objectworld" [21], a hypothetical computing environment geared towards information dissemination in which all objects could be mobile.

Within the scope of the work described in this paper, a mobile agent will be defined as:

"a software agent that is able to autonomously migrate from one host to another in a computer network."

By their very nature, mobile agents are inherently distributed. As such, they must be executable across a variety of platforms and operating systems to achieve their full potential. In a small, private network there may only be one configuration upon which they must work, but their true advantage comes from being able to migrate to disparate systems and continue functioning. This need has influenced the way in which mobile agent systems are created in that it is the norm for these systems to be written in some type of script or bytecode that can be interpreted. Interpretation removes the need to recompile the agent on arrival at a new host, and places the onus on ensuring an environment exists at the host that is capable of uniformly executing the agent on arrival. This model for ubiquity has been used most powerfully by the Java programming language [11]. Most examples of mobile agent systems have a server or some type of running environment in which the agents are executed [12]. Figure 1 illustrates this point, showing mobile and static agents operating within two separate agent servers

These servers act as workplaces where agents of all types can communicate with each other, they provide a means for hosting and managing agents in an environment that is secure from malicious agents. (A malicious agent is one that may attempt to take control of another agent, or access its private internal data). Through this server, an agent is able to get information about its environment, and to send and receive messages to that environment and other agents currently active in its proximity. Mobile agents are able to communicate by passing messages between themselves via the server environment.

- In General: Mobile Agent systems are quite different to traditional distribution systems prevalent in industry today. Mobile Agent systems may be characterised as providing local interaction for communicating objects, and providing facilities for mobile logic *and* data.

These characteristics of local interaction through both mobile logic and data provide a novel set of features for mobile agent systems. It is these features of disconnected operation and application autonomy, reduced network loading, intelligent information retrieval, and server flexibility that can help to provide a level of agility above that provided by more conventional IT technology [4].

3.1 Disconnected Operation and Application Autonomy

The advent of small mobile devices such as laptop PCs, and their exploitation within modern enterprises raises the problem of disconnected operation for a set of users brought up on network systems with continuous access to information and shared resources. Mobile agent systems can add a new dimension to this type of architecture through their ability to relocate from a mobile client to a remote host and continue processing on the network while the client is disconnected. On reconnection, the mobile agent can return to the client having fulfilled some task, typically information retrieval and filtering.

By encapsulating within a mobile agent the ability to make decisions for itself based on its environment, the agent is capable of continuing with its tasks autonomously, even when the client at which it was generated is unavailable.

A body of research exists that has focused on disconnected autonomy as a prime lever for information retrieval and dissemination[14][19], but this capability is equally applicable to shop floor traceability, order tracking, and sales order processing. This feature is exploited within the work described in this paper to enable sales staff to participate in the sales order processing system using mobile clients on laptop PCs.

3.2 Reduced Network Loading and Intelligent Information Retrieval

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. Local Interaction enables this more efficient and flexible system. Typically data requests that are dependant on preceding ones, are able to react immediately to the requested information, without having to wait for lengthy network data transfer times. By using the mobile agent approach to encapsulate logic within a request for information any filtering or dialog related to the required information can take place on the remote host

3.3 Server Flexibility

This is one of the essential differences between traditional message passing systems and those based on mobile agents. Mobile agents can be allowed to change the behaviour of a server, without the prior consent of the owner of the server. In a process dependant system, this flexibility can be extremely useful. This behaviour would obviously not be desirable in a public system, but in a controlled environment it could be extremely useful to update business logic at a remote server by simply dispatching a mobile agent designed to update that logic.

4 JAVA, MOBILE AGENT ENVIRONMENTS AND DATABASE ACCESS

The agents discussed in this paper can be classified in line with Franklin and Graesser [10] as goal oriented, communicative, and mobile i.e.:

- Goal oriented – they do not simply act in response to the environment
- Communicative – they are able to communicate with other agents
- Mobile – they are able to transport themselves from one host to another.

In order for these agents to exist within a system or to form a system themselves, they require a framework for implementation and execution. This is known as the agent environment as introduced in Section 3.

It can be argued that Java has become the *de facto* language for writing mobile agent environments. Systems using it include Aglets, Voyager, Mole, JATLite and Concordia [1][24][17][7]. The use of this platform neutral language provides the portability demanded by the mobile agent paradigm. The platform neutrality forms a good basis for enterprises requiring system agility, or wishing to form virtual enterprises, although accessing legacy systems is still an issue. This problem has been recognised, and the recent availability of facilities for database access via the Java JDBC interfaces has improved conditions.

4.1 JDBC

Java Version 1.1 introduced the JDBC Application Programming Interface (API); a set of classes and programming interfaces for executing Structured Query Language (SQL) statements which are modelled loosely on Microsoft's Open Database Connectivity(ODBC) standard. JDBC allows Java programs to access ODBC compliant databases in pure Java, without using C++ stubs, which makes it possible for developers to write database applications using a pure Java API. JDBC is a '*low-level*' interface allowing developers to invoke SQL commands directly and providing a base upon which to build higher level interfaces and tools.

Direct access to ODBC from a Java program has previously been achieved by using the JDBC-ODBC bridge supplied free as part of the standard Java Developers Kit (JDK) [15]. However, there are several key issues that JDBC addresses with respect to ODBC access with Java that provide incentives for using the new approach:

- ODBC uses a C interface, and calls from Java to native C code produce drawbacks in security and portability;
- There are language differences between C and Java e.g. Java does not use pointers, whereas ODBC uses them extensively;
- When using ODBC the driver manager and drivers must be installed manually on every client machine. If the driver is to be portable and secure on all Java platforms then the driver must be written solely in Java.

With a pure Java JDBC driver it is possible to serialise and encapsulate the driver within a mobile agent and dispatch it to a remote server, where it can be automatically installed and loaded. This could have advantages for enterprise integration where changes to a remote database server can be achieved with a single agent.

5 INDUSTRIAL SCENARIO

The scenario upon which the applications work in this paper is based was derived from data collected during a case study of a leading vacuum component manufacturer based in the UK. A simplified model describes their Sales/Order process. Figure 2 provides an overview of the model that involves Sales, Distribution and Manufacturing domains. It can be seen from Figure 2 that the Sales Order object does not reside in any of these domains, rather it naturally moves between them to achieve its goal of order completion.

5.1 A Mobile Agent Model for the Sales/Order Process

In the Sales/Order model in Figure 2 there are seven component types. The proposed agent architecture shown in Figure 3 reflects this real world model but concentrates on the interaction between the sales operation and the distribution point through the sales and ordering process. This requires three different agent types, one mobile and two static. It is understood that there will also be a need to generate new

agents for The company has a requirement to add new sales agents to their existing network and possibly new distribution points in the Far East and USA. For this to be achieved there is a need for an agile and flexible IT system. This model is also influenced by the requirements for sales order processing of a generalised manufacturing and distribution based virtual enterprise.

Purchase Orders or internal Works Orders, when an ordered product is not currently in stock. Although the implementation requires only one mobile agent type it should be noted that at run time potentially hundreds of different mobile order agents are active. Thus, the system as a whole is composed of a mass of mobile agents and several important static agents.

The model is based on the scenario faced by the vacuum component manufacturer. Sales outlets are distributed around the world, whilst the central stock warehouse and manufacturing plants are based in the UK. The areas of the model enclosed by dotted lines show the processes outside the scope of the initial phase of the research; these will be added to the model in the future.

In the Agent architecture, when an order is placed by a customer, the Sales agent generates an Order agent whose mission is to fulfil the order and return a delivery date to the sales agent. The mobile Order agent is dispatched to the StockControl agent at a distribution point server. On arrival, it initiates communication with the resident StockControl agent, and requests the fulfilment of its order by passing over an `Order` object. The StockControl agent queries the stock database to see if enough products are in Stock. If there are enough products, the agent then returns a `DeliveryDate` object to the Order agent which itself returns to its parent Sales agent.

The three key agent types identified when modelling sales and distribution enquiries are Sales Agents, Order Agents and StockControl Agents. These are discussed in the following section.

6 AGENT TYPES

6.1 Sales Agents

Sales Agents are static, GUI based agents that are responsible for generating Order Agents and dispatching them to the StockControl Agent. These could be resident in a very slim client for a bank of sales persons working on terminals or NetPCs, or hosted in a laptop for a travelling sales person. These must be capable of keeping track of all current orders that have been placed.

6.2 Order Agents

Order Agents are mobile, encapsulating a single order per agent; they are responsible for completion of the Sales/Order process for that order. Valid outcomes could be reporting to the Sales Agent a delivery date for the product or an allocation for materials and an internal works order number. Order Agents have no GUI.

6.3 StockControl Agent

The StockControl Agent is static, with no GUI² It is responsible for handling all requests for products, parts, or materials, and thus must interface to the stock control database. If the current stock levels are unable to satisfy an order, it must be able to use the encapsulated Product ID to derive the Bill of Materials. How this is achieved would depend on the existing IT system. The BOM's may be kept in a separate database, or be part of the same legacy system. There will also be a need to generate new agents for Purchase Orders or internal Works Orders, when the ordered product is not currently in stock.

When generating StockControl agents to unify the variety of database systems typically seen within a virtual enterprise it became apparent that some required aspects were particular to each database, whilst others were generic to all StockControl agents. In considering this problem the use of a generic 'Database Query Agent' was conceived which could be used as a base pattern for all StockControl agents in the system. The advantage of such a technique is the consistency and reusability inherent in using a pattern.

7 QUERY AGENT

If the proposed system was to truly enable system agility, then it must be capable of querying a variety of new or legacy databases, not just ODBC compliant ones, since in the main, legacy systems are not ODBC compliant.

By taking a modular approach to the design of the Query Agent and using established OO principles, the authors have derived an effective and reusable agent pattern that affords a flexible and extensible infrastructure to the system designer. The Query Agent can be decomposed into several key components, as follows.

7.1 The Infrastructure

This is dictated by the agent environment for which the Query Agent is being developed. It must include mechanisms for dispatching, retrieving, shutting down and restarting agents in a suitable host. Most of this functionality is generally available through the host environment or by subclassing from abstract classes provided by the environment. For example, in the authors' implementation using the Aglets Software Development Kit (ASDK) it is usual to extend the abstract Aglet class, and then provide implementation specific details if required.

7.2 The Identifier

The Identifier plays an essential role in system security. Whilst it is more usual for mobile agents to have the need to carry an Identifier, static agents must also be able to prove their credentials. An essential role of the StockControl Agent is to generate PurchaseOrder Agents and WorksOrder Agents in order to fulfil

² No GUI refers to the unattended running mode of the agent. A hidden GUI is often used for configuration and monitoring of agents.

unsatisfied orders. Part of the Identifier is handed to these child agents, as proof of their origin on dispatch to another host.

7.3 The Communication Package

Communication methods vary with different agent environments, as do the communication requirements of differing agent based solutions. In some examples, simple `String` matching is sufficient for communication. This can be seen in basic Aglet examples. The problem of achieving semantic level communication between agents is the subject of a large body of research. The use of KIF and KQML [9] is typical of the more advanced approaches that are being proposed. To handle this variety of communication methods, the proposed pattern includes the Comms Package that can be interchanged by the designer with respect to the system requirements. Its role is to receive the incoming communication from arriving or querying agents and translate this into a format the Business Logic Unit or Database Handler can understand.

7.4 Business Logic Unit

In the Sales/Order scenario when an Order Agent is dispatched by the Sales Agent, it encapsulates an Order object. On arrival at the StockControl agent, it is tasked with attempting to fulfil the encapsulated order. This task in itself can require some simple logic. For example, the Order agent may only know that it requires 100 widgets by Tuesday, the StockControl agent may know that a widget must be supplied with a grommet and 2 nuggets. Thus, the Order actually requires 100 widgets and grommets, plus 200 nuggets. Since all the Order agents will require this same logic it is clear that it is better suited to being part of the StockControl agent. By simply handing over the order object and awaiting an answer, the size of the Order agent is kept small, reducing network traffic.

7.5 The Database Handler

The Database Handler deals with all the details of connecting to a database, retrieving information, updating the database, or even switching databases transparently to the requesting agent. It works in tandem with the business logic unit to fulfil the request of the Order agent.

Figure 4 shows the benefits of adopting a modular approach to the design of the Query Agent. The examples shown address a large percentage (but by no means all) of the real world situations and the methods currently being employed in the use of databases within an enterprise. The modular Database Handler can be interchanged for whatever the situation requires.

In the case where a virtual enterprise, that is already running a mobile agent system, requires a new collaborator, accessing the new information contained within their databases requires only the production of a new Database Handler. If their IT system is sufficiently advanced this could be as easy as linking a JDBC-ODBC bridge to their middle tier, and providing a host for the agent. It is understood that access is not all that is required, there remains the difficult problems of understanding the schema used in the new

database before specific information can be retrieved. Work towards this goal can be seen in the efforts of the EDI, STEP/PDES community [6].

7.6 DataHandler Implementation

For the implementation of the StockControl agent and its resident DataHandler a database was created using Microsoft Access. Its contents were based on a products and materials list obtained during the case study. This database was then made globally accessible using the dbAnywhere server available from Symantec. The mobile agent environment used to develop and test the implementation was the ASDK, and the StockControl agent was hosted within the Tahiti server supplied with the Aglet package. It was decided that the best way to access the database was to use the driver supplied by Symantec and the JDBC-ODBC Bridge. The implementation was also tested using the native ODBC support present in the Win32 platforms.

In the construction of the two DataHandlers a common set of functionality emerged which evolved into two distinct Java classes that were very light weight, but invaluable in connecting and testing database connections. They are `ConnectionInfo` and `DBConnector`.

7.6.1 DBConnector

This class is used to perform all operations in connecting to a JDBC compliant data source. For systems that allow it, the driver name can be put into `System.properties`. If not, the driver is just loaded using `Class.forName`. Connection details are encapsulated within an instance of `ConnectionInfo`.

7.6.2 ConnectionInfo

This class is used to encapsulate all Connection details required to connect to a JDBC compliant database. Normal use is with the `DBConnector` class. It supports serialisation in order to allowing saving the details to file.

In the construction of the Sales/Order agent architecture, it became apparent that to the untrained the hardest part of setting up a StockControl agent was in making the connection, via the DataHandler, to the database. Whilst on the surface a relatively simple task, there are several variables that must be set correctly, and a number of JDBC interfaces that must be used accurately. To alleviate the problems this caused the DataConnector tool was produced to automate some of these tasks. Details of this are described in [18].

8 SYSTEM SECURITY

Security is one of the issues always being raised as an impediment to the progress of Mobile Agent technology. In this section, we provide a brief analysis of the risks involved, and then discuss some simple steps that can be taken to minimise those risks.

8.1 Host Security

Like any other type of code that is downloadable and executable, mobile agents pose a threat to the host and server in which they are running. In fact, the similarity between mobile agents and computer viruses is striking. However, the fundamental difference is the intent of the programmer.

8.1.1 Agent Attacks Host

The most obvious risk is that of an incoming agent attacking the server or host. In this case a mobile agent arrives at a new host and proceeds to attack the server in a number of ways. These attacks could be in the form of attempts to corrupt or exhaust the local resources (file system, memory, database, etc); Denial of Service attacks where the agent effectively attempts to crash the server; falsification of credentials in an attempt to disguise the identity of the real owner, or real agent; and denial, in that the agent later denies all knowledge of some preceding communication that really did take place.

8.1.2 Third Party Attack

This type of attack is quite similar to those currently prevalent on the Internet whereby a person attempts to 'hack' a server in an attempt to remove its presence from the Internet, fatally halt the execution of the server, or obtain confidential details from the server. These are also Denial of Service attacks and generally revolve around some type of repeated action. E.g. flooding the attacked host with repeated dispatches of the same agent.

8.2 Agent Security

In contrast to the similarity between the attacks of virus and mobile agents on remote hosts, mobile agents can also come under attack from other agents. They are also susceptible to attack from the host in which they are being executed.

8.2.1 Attack from another Agent

In most circumstances, any running agents at a host will have to co-habit with other agents, be they resident or guests. This shared execution space means that a legitimate agent may be exposed to a malicious agent that attempts to attack it in any number of ways. This attack could be an attempt to obtain internal private data from the agent or an attempt to either terminate the agent, or get it to terminate itself.

8.2.2 Attack from the Host

As mentioned in Section 3 most current mobile agent systems are interpreted. By its very nature, interpretation implies that a copy of your agent is 'played' through the interpreter. This fact may allow a malicious host to later replay your agent to replicate its behaviour, or obtain the same services. There is also the possibility that the host may attempt to extract private data, e.g. a credit card number, from the agent as it executes.

8.3 Network Attacks

In this type of attack some one attempts to either tamper with or copy your agent as it is *en route* to its new host. This can take the form of subsequent replay of your agent, as described above; eavesdropping upon a network to monitor agent-agent and host-host remote communications in an attempt to obtain useful information; tampering with your agent *en route* so that it may subsequently perform an illegal action at the host, for which you would be responsible.

8.4 Protection from Attack

With all these methods of attack, it may appear justified to assume that Mobile Agents are too fraught with risk to use as a viable technology. In the case of a completely public server, this may currently be true, although there has been considerable progress made in defining approaches for combating this problem [16]. However, when the network in question is closed, for example an Intranet or extraNet, and/or tightly controlled, most of these issues disappear.

With the constant improvement to digital signing and public key encryption, the types of network attack discussed above can almost be entirely avoided. In the scenario presented in the paper all the agent hosts could be tightly restricted to accept agents only from other 'trusted' hosts. By 'trusted' we mean a host that we have previously defined as one that we are sure will not attempt anything malicious. Within the virtual enterprise we must assume that all our partners are trustworthy. Any agent attempting to migrate to one of these hosts from an untrusted site should be refused entry. Since we are now sure of the origin of the agents in our system, we should be able to assume that none of the agents will intentionally corrupt or exhaust any local resources or attack other agents.

By defining a list of trusted hosts and defining a policy for authentication, the risk of using mobile agents can be compared to receiving email from a friend or business partner. They could send a virus by accident and equally, a badly programmed agent that had not been properly tested could be dispatched to a host and exhaust all the windowing resources.

8.5 Secure Agent Retraction

Part of the justification for using mobile agent technology was its support for disconnected operation. For a travelling sales person this could be extremely useful since they are able to dispatch an agent to fulfill an order whilst they switch off their laptop and continue onto the next site. However, although the agent is likely to move to only a few known sites, if the sales person has been 'out of the loop' for a couple of days, a new configuration for the remote servers might have appeared. For example, perhaps a new StockControl agent has been added at a new site, and the mobile agent has been advised of this when querying a known StockControl agent. If this happens before the list of trusted sites has been updated on the laptop the mobile agent might never be able to return home.

The implementation discussed in this paper handles this situation by allowing mobile agents to request a retraction home. Retraction is meant as the opposite of dispatch. On completion the mobile agent sends a

message to its Master, the SalesAgent, asking to be pulled back. Encapsulated within the message is their unique identifier, which the SalesAgent can use to check against its list of valid Order agents. A code fragment illustrating this principle is show below.

```
public void returnHome() {  
    try {  
        Message msg = new Message("RetractMe");  
        msg.setArg("url", getHostingURL());  
        msg.setArg("id", getAgletID());  
        getMaster().sendOnewayMessage(msg);  
    } catch (Exception e) {  
    }  
}
```

Also included is the current host of the mobile agent. The Sales Agent is constantly available for messaging and any incoming message is passed to its message handler. The corresponding reaction to a message from an Order agent is show below.

```
if (msg.sameKind("RetractMe")) {  
    try {  
        URL host = (URL)msg.getArg("url");  
        AgletID aid = (AgletID)msg.getArg("id");  
        retractAglet(host, aid);  
    }  
}
```

In this way, the underlying IT system can ensure it remains flexible, even handling the addition of a new StockControl agent into the equation gracefully. This type of flexibility is paramount to the needs of a virtual enterprise in its efforts to be competitive.

9 CONCLUSIONS

This paper proposes the notion that mobile agent technology provides a useful software paradigm that enables IT system designers to model and implement their systems in a more natural reflection of the real world they simulate and support. A direct relationship is established between the mobile elements of a distributed information system and the agent based architecture of the IT system upon which it is implemented. This improved alignment can itself be a significant aid in enabling IT systems to evolve in line with the real world they represent. In addition, mobile agent technology can help in the rapid formation of these information systems, which can be vital when supporting the creation of virtual

enterprises. Typically, the adoption of Java as the *de facto* language for mobile agent environments, and its database connection capabilities provided by the JDBC interfaces, ensures mobile agents are capable of integrating with legacy and ODBC compliant data sources.

A model generated from a case study identifies three key types of agents required in the Sales/Order process. The authors propose a pattern for database query using agent technology, and a tool to aid rapid integration of existing data sources. Implementation based upon this pattern has provided flexibility additional to that inherent to the agent paradigm.

Finally the issue of security in mobile agent systems has been discussed and an approach for maintaining system security has been presented.

10 ACKNOWLEDGEMENTS

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

11 REFERENCES

- [1] Aglets: <http://www.xxx.ibm.co.jp/aglets>
- [2] Ball et al, 1997, "Enterprise Enablement for Java Applications", white paper, XDB Systems.
- [3] Camarinha-Matos, L. M., Afsarmanesh, H., Marik, V., "Intelligent Systems for Manufacturing, Multi-Agent Systems and Virtual Organizations", Kluwer Academic Publishers, 1998, ISBN 0-412-84670-5
- [4] Chess, D., Harrison, C., Kershenbaum, A. "Mobile Agents: Are They A Good Idea ?", in "Mobile Object Systems, Towards one programmable Internet", Edited by Vitek, J., Tschudin, C., Springer-Verlag Lecture Notes in Computer Science 1222, 1997, ISBN-3-540-62852-5.
- [5] Clements, P.E., Papaianou, T. and Edwards, J.M., "Aglets: Enabling the Virtual Enterprise", Proceedings of the 1st International Conference on Managing Enterprises - Stakeholders, Engineering, Logistics and Achievement, ME-SELA '97, Wright, Rudolph, Hanna, Gillingwater and Burns (eds), Mechanical Engineering Publications, Loughborough University, July 1997, pp 425-432, ISBN 1-86058-066-1.
- [6] Clements PE, Coutts, IA, Edwards, JM, "A model based approach to enterprise wide information support" published in "Information Infrastructure Systems for Manufacturing". Edited by JBM Goossenaerts, F Kimura and JC Wortman, Chapman Hall, ISBN 0-412-78800-4.1997.
- [7] Concordia: <http://www.meitca.com/hsl/projects/concordia>
- [8] Coutts I, Edwards J, "Support for Component Based Systems:Can Contemporary Technology Cope?", Intelligent Systems For Manufacturing, Edited by L.M. Camarinha-Matos et. Al., Kluwer Academic Publishers, 1998, ISBN 0-412-84670-5, pp279-288.
- [9] Finin, T., Labrou, Y., Mayfield, J., 1997, "KQML as an agent communication *language*", MIT Press, Cambridge, to appear. Available at <http://www.csee.umbc.edu/kqml/papers/>
- [10] 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, Published as Intelligent Agents III Springer-Verlag, Berlin, 1997, pp 21-35.
- [11] Gosling, J., Joy, B., Steele, G., "The Java Language Specification", Addison Wesley, 1996, ISBN 0-201-63455-1
- [12] Gray, R., "Agent Tcl: A flexible and secure mobile-agent system", PhD thesis, Dept. of Comp Sci, Dartmouth College, June 1997.
- [13] Hamilton, G., Cattell, R., Maydene, F., "JDBC Database Access with Java", Addison Wesley, 1997, ISBN 0-201-30995-5.
- [14] Hofmann, M. O., McGovern, A., Whitebread, K. R., "Mobile Agents on the Digital Battlefield", Proceedings of the Second International Conference on Autonomous Agents, ACM Press, 1998, pp219-225.
- [15] JDBC Web Site at <http://www.javasoft.com/jdbc>
- [16] Neuenhofen, K., Thompson, M., "A Secure Marketplace for Mobile Java Agents", Proceedings of the Second International Conference on Autonomous Agents, ACM Press, 1998, pp212-218.
- [17] Odyssey: <http://www.genmagic.com/technology/odyssey.html>
- [18] Papaioannou, T., Edwards, J., "Mobile Agent Technology in Support of Sales Order Processing in the Virtual Enterprise", Intelligent Systems For Manufacturing, Edited by L.M. Camarinha-Matos et. al., Kluwer Academic Publishers, 1998, ISBN 0-412-84670-5, pp23-32.
- [19] Rus, D., Gray, R., Kotz, D., "Transportable information agents", Journal of Intelligent Information Systems, 9:215-238, 1997
- [20] System Software Associates Inc., "BPCS Client/Server Distributed Object Computing Architecture", White Paper, 1995
- [21] Tschudin, D., "Objectworld", Office Automation, Springer-Verlag, 1985.
- [22] Wright D.T, Burns N.D, "Impact of Globalisation on Organisational Structure and Performance", Proc. of the Organisational Management Division, International Association of Management 14th Annual Conference. Toronto, Canada, August 2-6, 1996, pp. 58-63
- [23] White, J. E. "Telescript technology: the foundation for the electronic marketplace", White Paper, General Magic, Inc. USA, 1994.

[24] Voyager: <http://www.objectsspace.com/Voyager>

12 BIOGRAPHY

Todd Papaioannou received an Honours Degree in Engineering from Loughborough University, before becoming a member of the MSI Research Institute where he is currently in the final stages of his PhD. His primary research interests lie with mobile agent technology, distributed systems and enterprise integration. He has been an active member of the mobile agent community and is a co-founder of The Mobility mailing list. A list of his papers can be found at <http://luckyspc.lboro.ac.uk/Docs/Papers>.

John Edwards gained his PhD from Loughborough University in 1994. Having spent 13 years in UK process and manufacturing industry, being involved in the creation of computer control and information systems, he joined Loughborough University in 1987. During his 11 years at Loughborough he has been involved with the Systems Integration Group and the MSI Research Institute and he has recently formed his own research group Researching Evolutionary Development (RED) techniques in IT. His role within the Manufacturing Engineering Department is as a Lecturer and principal investigator on a number of UK government funded research projects.

13 FIGURE HEADINGS

Figure 1 Static Agents, Mobile Agents and Agent Servers

Figure 2 Unified Modelling Language (UML) concurrent object message diagram of the sales order scenario

Figure 3 Proposed Agent Model for Sales/Order Processing

Figure 4 The Query Agent pattern and examples of Data Handler modules

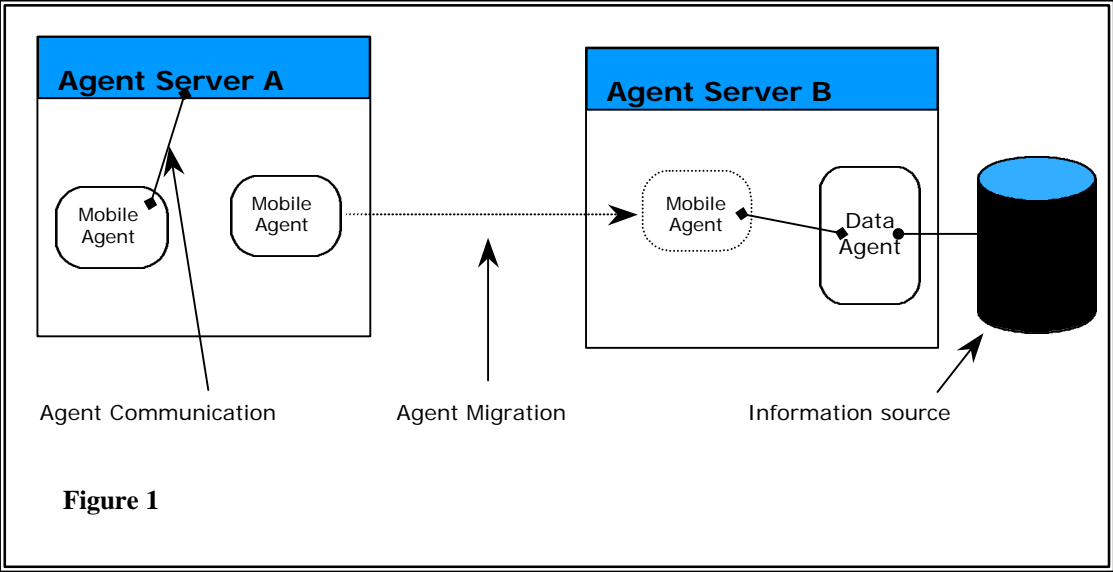
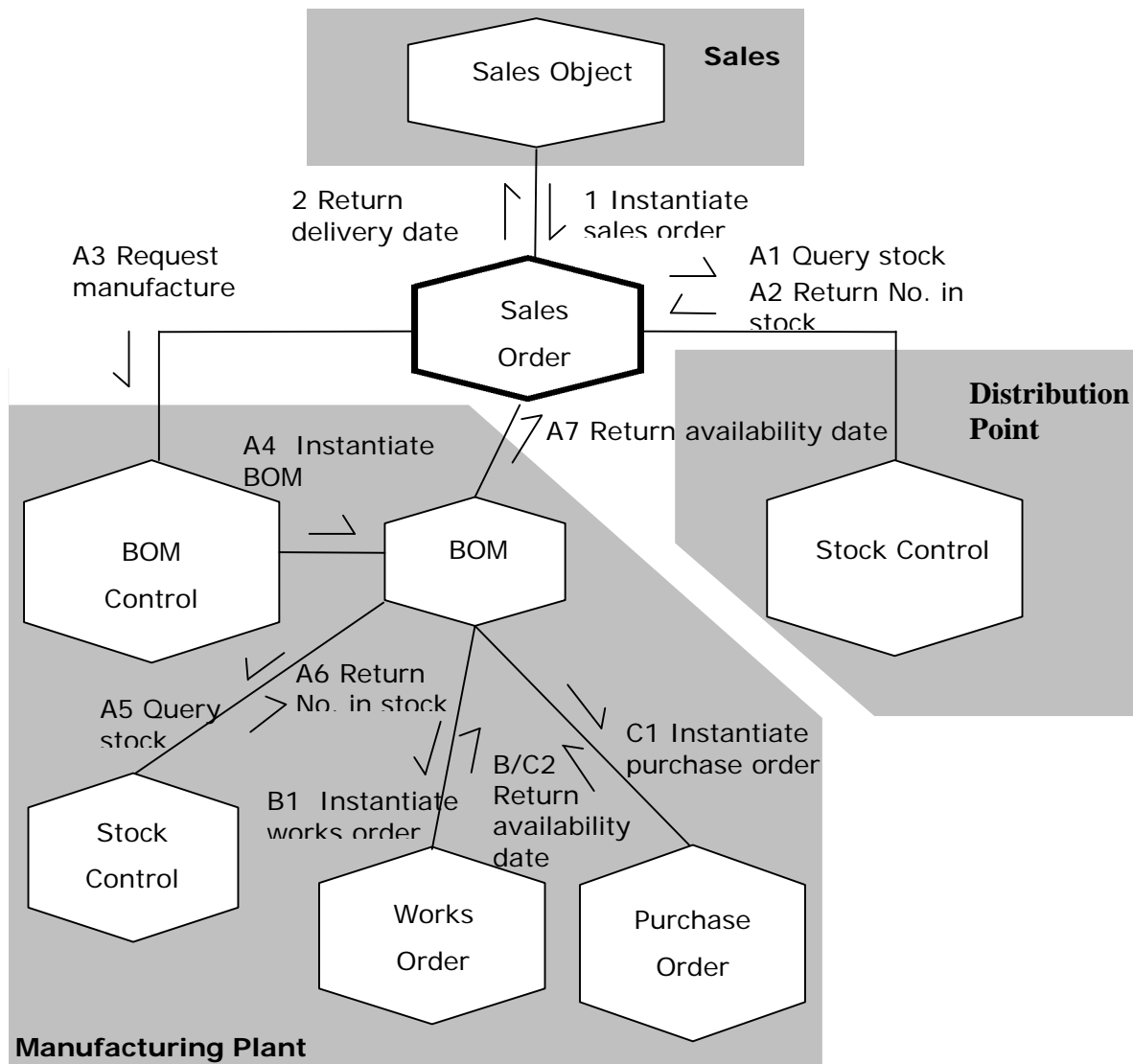


Figure 2



An order is placed with the salesman by a customer. The sales object instantiates a Sales Order, which has a Product ID associated with it. A query can now be made to the Stock Database at any one of a number of distribution points requesting allocation of the full product, or subparts if assembly is still required. Successful completion returns the allocation to the sales order with a delivery date. If the order can not be supplied from stock, it is possible to generate a Bill of materials (BOM) from the Product ID. The BOM holds details of all the subparts of the product. A query is then made to the stock database requesting allocation of raw materials to manufacture any or all of the subparts. This allocation is then associated with a newly instantiated Works Order, which is subsequently dispatched to the production scheduler. Any bought out material requirements result in the instantiation of Purchase Orders.

Figure 3

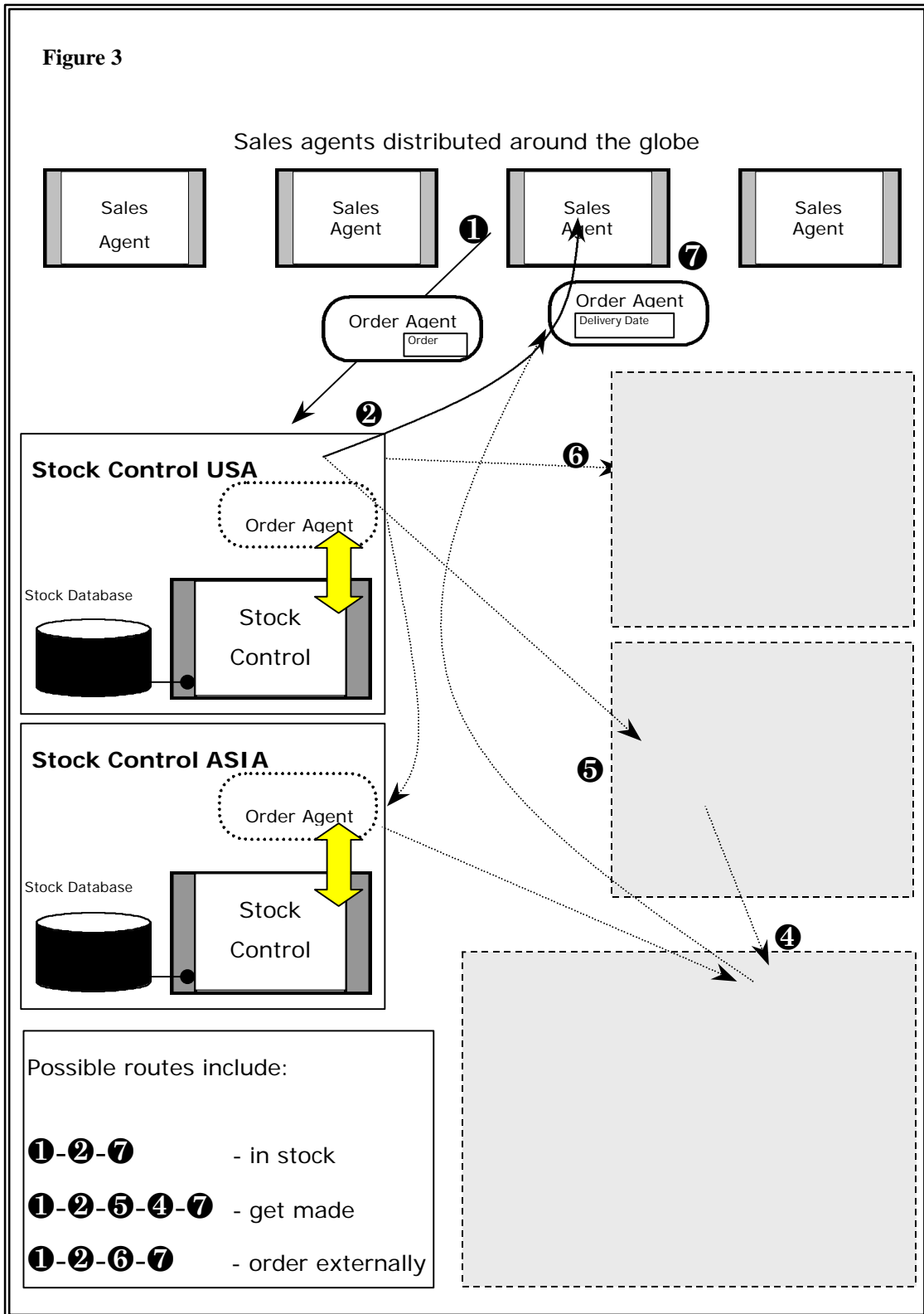


Figure 4

