A MOBILE-AGENT BASED ARCHITECTURE FOR VIRTUAL ENTERPRISES

Luís Brito, José Neves and Francisco Moura

Departamento de Informática

Universidade do Minho

Campus de Gualtar - Braga

PORTUGAL

lbrito@gil.di.uminho.pt, jneves@di.uminho.pt, fsm@di.uminho.pt

Emerging technologies that allow a two-way communication between companies, or among companies and their customers, are changing the rules of the market, facilitating the emergence of virtual entities that have to be supported by some kind of platform. Making it is the main objective of this work, materialized in a Mobile Agent Architecture (MAA) which supports the emerging world of the m-Commerce (mobile-Commerce): the MAgnUM architecture. The MAA's development was based on the principles of process and knowledge abstraction, compositionality, reuse, formal semantics, formal evaluation, and security.

1. INTRODUCTION

Virtual Enterprises (VEs) are sometimes seen as a branch of a broader *Electronic Commerce (EC)* effort. Typical approaches to *EC* are mainly reduced to catalogue sales and a simple *Business-to-Consumer (B2C)* perspective.

The lack of commonly accepted standards, the heterogeneity of the systems available at each enterprise and the different approaches to *Information Systems (IS)* by management, hinders the development of important (and highly needed) *Business-to-Business (B2B)* *EC* systems. *VEs* – seen as the strategic alliance of independent enterprises, enabling synergies that make the progress towards a common goal more profitable – fall into the realm of the *B2B* perspective.

However, the traditional approach to *EC* in the form of *Virtual Marketplaces (VMs)* can not be fully separated from the *VEs* perspective. In fact, *VMs* and *VEs* can easily be unified in a single view (Figure 1).

Many are the problems that can be addressed through the use of *VEs*. However, one of the most important is the generation of orders; *i.e.*, the enterprises, that form a broader *VE*, must be able to communicate/negotiate among themselves, so that the orders placed at a certain enterprise may be carried out by supplies of raw materials, originating at the other enterprises of the group. Therefore, each element of the set of enterprises must provide a platform for database query and insertion (Brito et al, 2000).

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: [10.1007/978-0-387-35399-9_52](https://doi.org/10.1007/978-0-387-35399-9_52)

L. M. Camarinha-Matos et al. (eds.), *E-Business and Virtual Enterprises*

© IFIP International Federation for Information Processing 2001

Figure 1 – Unifying the VM and VE views over EC dealings

A possible approach for the intelligent coordination of actions among the elements that compose a *VE* is the use of *Multiagent Systems (MAS)*. These systems provide, by definition, the independence required by each of the elements while gathering individual efforts for a common goal to be satisfied. However, communication among the agents that form the system is complex and requires, for a system to be reliable, the use of extremely complex messages, written in languages such as *KQML* (Finnin et al, 1994), (Austin, 1962), (Searle and Vanderveken, 1985).

Another approach to the establishment of strong relationships among the elements of a *VE* is the use of *Mobile Agents (MAs)*. These agents rely on an infrastructure built over the set of enterprises and eliminates the need for a community of static agents. Each agent is now able to roam through all the enterprises, following a static or dynamic map. The need for information exchange is reduced and the communication overhead for query refining over large databases is reduced.

2. DESIGNING A VIRTUAL ENTERPRISE

A *VE* may have an architectural basis that reflects some of the specificities of the enterprises that are being dealt with. However, when designing a *VE* as a set of allied enterprises, much is to be gained from the use of an homogeneous

architecture; *i.e.*, from the use of similar mechanisms to support the alliance. Therefore, it can be said that all enterprises are hierarchically equivalent (Figure 2).

Figure 2 – Two VEs with intersecting enterprises

The internal structure of each *atom* (enterprise) of a *VE* must reflect the basic needs of a system that aims at commercial exchange and, therefore, at profit. The central mechanisms for order analysis, scheduling and the search for raw materials are extremely important (Figure 3).

Figure 3 – General architecture for each atomic component of a VE

Each enterprise has to generate *probes* that act over the local databases or the remote databases situated on the other enterprises. These *probes* roam through the set of elements, following a map provided at generation time and updated during the travel. Through the information provided by local databases, reservation orders can be issued and, at a later time, on a second *probe round*, reservations are confirmed or eliminated. Taking into account that, in the lack of a certain raw material, a

compatible material should be chosen (not necessarily with the same number of units) by querying a *compatibility database* and running a *choice algorithm*.

The use of at least two rounds of probing aims at the fact that, by gathering information of all distinct enterprises, a choice can be made about who is to provide a certain material. Through a set of metrics, that may range from price to delivery time, the *probe* is able to autonomously decide on a commitment; *i.e.*, a first round is used to make reservations and gather information and posterior rounds commit the enterprise to some supply.

3. IMPLEMENTING MOBILE AGENTS: THE MAGNUM ARCHITECTURE

3.1 Distribution through CORBA

Working with distributed systems demands the use of communication infrastructures and operating system support. Although traditional programming relies on the use of abstractions such as *sockets*, the continuous growth in software complexity drove the field towards higher abstraction levels such as *Remote Procedure Call (RPC)* and *Remote Method Invocation (RMI)*.

The *Common Object Request Broker Architecture (CORBA)* effort is intended to provide easy programming, high flexibility and platform independence. *CORBA* is not only characterized for providing a programming framework but also for the existence of higher-level frameworks for business applications (*CORBAfacilities*) (Orfali and Harkey, 1998), (Rosenberger, 1998).

CORBA positions itself as a powerful alternative to *RPCs* and *RMI* due to many factors. Some of them are:

- *high abstraction level*: the programming issues are independent of hardware and software characteristics (*marshaling* is transparent);
- *availability*: *CORBA* is available for almost all platforms and can even be platform independent;
- *support for heterogeneous scalability*: clients and servers in a distributed system can be implemented in different languages, shielding the programmer from problems that arise when expanding an already existent system.

3.2 MAGNUM

Mobile agents can be classified into two main classes: agents with *real-world mobility* and agents with *network mobility*. Agents with *real-world mobility* are able to roam through a set of positions (not necessarily predefined) without network connectivity relying, many times, on an human carrier (*e.g. Personal Digital Assistants – PDAs*). Agents with *network mobility* are nothing more than software agents that rely on a computer network (with no predefined characteristics) in order to position themselves on each host. For the purpose of a general *VE* it can not be

assumed that an human carrier is available, therefore agents with *network mobility* are the best choice.

The construction of a mobility infrastructure has much to gain from using *CORBA*, which is the base of the *MAGnUM* (*Mobile Agents at the University of Minho*) model (Brito, 2000). This model bases itself on two architectural levels: the *host infrastructure* and the *base class structure*.

The host infrastructure is composed of a series of distributed objects called *MAGnUM Nests* that support the migration process. The *Nests* are extended by an interface that is, by itself, a *CORBA* object and by another object, the *MAGnUM CriptoNest*, that deals with all cryptographic issues among the *Nests* (cryptography is important not only for encryption purposes, but also for creditation). At an higher level, a *CORBA Naming Service* provides location services for all distributed *Nests*. The support for scalability enables the introduction of new objects into the system such as a *Trusted Agent* (*TA* – used for cryptographic protocols (Schneier, 1996), (Stinson, 1995)) and a *blackboard* (for communication purposes) (Figure 4).

Figure 4 – Possible topology of the distributed infrastructure provided by *MAGnUM*

The *base class structure* is a set of abstract Java classes, that support the elements that were defined as the most important for any mobile agent. Each implementation that uses *MAGnUM* has to extend the provided classes and implement some of the abstract methods. From a top class that deals with the high-level concept of mobile agents (*MobileAgent*) and implements the concept of object serialization (through the implementation of the *Serializable* class), spawns a set of subclasses: *IdCompound*, *CriptoCompound*, *GPSCCompound* and *HistoryCompound* (Figure 5). The *IdCompound* is responsible for the identification characteristics of a mobile agent (such as the name of the creator). The *CriptoCompound* enables all cryptographic exchanges that extend beyond the mobility infrastructure. The *GPSCCompound* provides the agent with a particular view of the world he is traveling in. The *HistoryCompound* enables the agent to charge itself with all interesting events that arise during its travel through a set of *Nests*.

Figure 5 – Class diagram for the base class structure following a *Booch* notation

The migration between two *Nests* is controlled by a set of restrictions. These restrictions enable each host, where a *Nest* is placed, to determine if an agent is able to enter and execute its code. The restrictions can be classified as one of two kinds:

- *High-Level Migration Restrictions (HLMR)*: these restrictions are placed on some of the characteristics of a mobile agent, such as creator and identifications;
- *Low-Level Migration Restrictions (LLMR)*: these restrictions are placed mainly on low-level characteristics such as the origin IP address or the transport port.

4. ENABLING VIRTUAL ENTERPRISES THROUGH MagnUM

The concept of *VE*, stated earlier, can be easily implemented through the use of *MagnUM*. This architecture enables the desired kind of enterprise with a set of interesting characteristics:

1. *support for intermittent connectivity*: the enterprises that compose a *VE* are closely tied due to their commercial relationships, however, the physical distance between them may be extremely large, leading to *Wide Area Networks (WANs)* that lack, many times, the desired *Quality of Service (QoS)*;
2. *reduction of the total amount of traffic exchanged among enterprises*: taking into account the use of *WANs* that span over large distances, the amount of traffic exchanged among the enterprise might be important (e.g. the cost for long distance connections);
3. *local work over each enterprise's databases*: the enterprise's databases tend, many times, to be extremely large making query and update operations time consuming and unsafe over open *WAN* environments such as the *Internet*.

A *probe* generated in the *MAGNUM* architecture is composed of a set of classes (extending the base classes that define a mobile agent and contributing, therefore, for software reuse), however, the most important method to be implemented is *run*. Such a method controls the main behavioral mechanisms determining migration needs and executing the database probings, reservations and confirmations. A possible implementation for the first round over the set of enterprises is:

```
...
switch (roundcounter)
{
  case 1: // connect to the raw materials database
    db1.connect ("rawmaterials");
    // connect to the compatibility database
    db2.connect ("compatibles");
    // connect to the reservations database
    db3.connect ("reservations");
    // reserve products
    availableneeds = db1.get_available (needs);
    db3.reserve (availableneeds, "raw");
    if (!equalNeeds (availableneeds, needs))
    {
      unavailableneeds = difference (needs, availableneeds);
      availablecompat = db2.get_available (unavailableneeds);
      db3.reserve (availablecompat, "compatible");
    }
    reservationsinfo.put ("raw"+gpsc.current_pos,
                        availableneeds);
    reservationsinfo.put ("compatible"+gpsc.current_pos,
                        availablecompat);

    db1.close ();
    db2.close ();
    db3.close ();
    // jump to next nest
    gpsc.jump ();
    break;

  ...
}
...
```

The number of following rounds depends on the strategy of the *VE*. If it is decided to use only two rounds then, the second round can be used to confirm all reservations made on the first round and any following step initiates the end of any further migration (the agent "dies"). The code is as follows:

```
...
case 2: if (gpsc.current_pos == 0)
    prunerreservations (reservationsinfo);
    // connect to the reservations database
    db3.connect ("reservations");
    // confirm reservations
    db3.confirm (reservationsinfo);
    db3.close ();
    // jump to next nest
    gpsc.jump ();
    break;
default: gpsc.die ();
}
...
```

The *VE* implemented through the use of *MagnUM Nests* has the ability to be highly dynamic. Through the interface of each *Nest*, a distributed object by itself, each enterprise may be able to enter or abandon the *VE* through the use of simple migration restrictions (e.g. if an enterprise wishes to abandon a *VE* it can do so by identifying either the hosts that compose such *VE* or the set of mobile agents – *probes* – that they are able to launch).

5. CONCLUSIONS

For a particular *VE* architecture that uses a set of homogeneous enterprises, the use of a mobile agent architecture to induce relationships among a set of enterprises is promising. Mobile agents are able to roam a distributed environment and, due to autonomic behavior, are able to take basic decisions. The advantages of this kind of process over systems such as traditional *MAS* relate to communication and processing power management. A *VE* system based upon mobile agents is able to: endure intermittent connectivity, reduce the information traffic among the set of enterprises and provide extra security and better processing power management through local execution.

The implementation of a mobile agent architecture must rely on the use of tools that enable communication and distribution. The use of low-level tools such as *sockets* is undesirable, highly unscalable and hard to maintain. New technologies, such as *CORBA*, enable a reduction of the programming and maintenance effort. The *MagnUM* architecture relies on *CORBA* and its benefits over other technologies such as *RPCs* and *RMI*.

VEs need to be highly dynamic. An enterprise must be able to position itself in or out of a set of enterprises. The *MagnUM* architecture supports such behavior through the use of high-level and low-level migration restrictions. By restricting the movement of the mobile agents that roam the *VEs* space new synergies among enterprises might be formed or broken.

REFERENCES

1. Austin, J. L., *How to do Things with Words*, Oxford, Clarendon, 1962.
2. Brito, L., *MagnUM - Mobile Agents at the University of Minho*, Research Report, Universidade do Minho, 2000.
3. Brito, L., Novais, P., Neves, J., *A General Agent-Based Architecture for Production Planning in Electronic Commerce Scenarios*, ESM2000 - European Simulation Multiconference, Ghent, Belgium, 2000.
4. Finnin, T., Fritzson, R., McKay, D., McEntire, R. *KQML as an Agent Communication Language*, In Proceedings of the 3rd International Conference on Information and Knowledge Management (Gaithersburg, Massachusetts, USA), 1994.
5. Orfali, R., Harkey, D., *Client/Server Programming with JAVA and CORBA*, 2nd edition, John Wiley & Sons, Inc., 1998.
6. Rosenberger, J., *CORBA in 14 days*, SAMS Publishing, 1998.
7. Searle, J., Vanderveken, D., *Foundations of Illocutionary Logic*, Cambridge, Cambridge University Press, 1985.
8. Schneier, B., *Applied Cryptography*, 2nd edition, John Wiley & Sons, Inc., 1996.
9. Stinson, D. R., *Cryptography: Theory and Practice*, CRC Press, 1995.