A Mobile Object Workbench

Michael Bursell, Richard Hayton, Douglas Donaldson, Andrew Herbert
APM Ltd Poseidon House, Castle Park, Cambridge CB3 0RD United Kingdom
Email: apm@ansa.co.uk Fax: +44 1223 359779

Abstract: Existing mobile agent systems are often constructed with a focus on intelligence and autonomy issues. We have approached mobility from a different direction. The area of distributed systems research is quite mature, and has developed mechanisms for implementing a "sea of objects" abstraction. We have used this as our starting point, and added to this the ability for objects to move from host to host, whilst maintaining location-transparent references to each other. This provides a powerful and straightforward programming paradigm which embraces programming language semantics such as strong typing, method invocation and encapsulation. We have built a Mobile Object Workbench on top of a flexible Java middleware platform, which can be used as a the basis for a Mobile Agent System. In this paper we examine the philosophy and design of the Mobile Object Workbench, and describe how this is being extended to provide a security framework oriented towards agents.

1 Introduction

Software agents, and in particular mobile agents, are currently an active area within both the research community and the commercial sector [1], [2]. However, despite obvious interest, the commercial take up of mobile agent technology has been slow, and the search for a 'killer app' continues. We believe that one of the reasons for this caution is that the mobile agents approach has been billed as an alternative to the traditional distributed systems paradigm. Typically, the transition to agent based programming requires a programmer to learn a new programming language, and discard existing programming tools and network services.

This sets up a chicken and egg situation for agent systems: until they can provide all the services and integration capabilities of current middleware such as CORBA [3] and DCOM [4], developers will be reluctant to build large systems using them.

We believe that the correct approach is to extend existing programming paradigms by adding autonomy and mobility facilities. This will allow developers to use these facilities where appropriate, and standard distributed systems techniques elsewhere.
We therefore present ‘agent facilities’ as a natural extension to the programming environment used for existing distributed system development.

This approach has many additional advantages when coupled to a network-oriented object programming language such as Java: method invocation extends naturally to remote method invocation, and strong typing support offers greater safety over ad-hoc messaging found in some current agent systems. Java has specific features that benefit our approach. Interfaces may be considered as the published access points for services, and objects as the means by which these services are implemented. This separation is important, as it allows some other object to act as a proxy for a real service object - by implementing the same interface, and providing the same semantics by communicating with the remote service. The construction and use of proxy objects can be made transparent to the application programmer, and it is easy to imagine how proxies may be used to represent services that are themselves mobile.

The notions of object and interface are fundamental to the design of distributed systems, and is formalized in the ODP reference model [5]. RM-ODP defines a number of distribution transparencies. Existing platforms such as CORBA provide access and location transparencies - the ability to communicate with an object regardless of its location or network address. The Mobile Object Workbench that we have constructed adds two additional transparencies: relocation transparency - a client need not be aware that a service has moved - and migration transparency - a service need not be aware that it itself has moved. Together these form a basis for the construction of mobile agents. The other ODP transparencies are replication, persistence, transactions and security. These are being addressed within the wider context of our project.

The mobile object workbench described in this paper is an implemented system that provides objects with access, location, relocation and migration transparency within the Java object model. It is not in itself an agent system; however it provides many of the basis facilities required by a mobile agent framework, and provides an easy transition path for programmers wishing to incorporate agent behaviour within the context of a larger distributed system. The work on the Mobile Object Workbench has been undertaken as part of the FollowMe ESPRIT project (No. 25,338) on support for mobile users. Other partners within the project are designing a fuller agent system on top of our Mobile Object Workbench [6].

1.1 Related Work

Many existing agent systems have been built from scratch, and have had to contend with both agency and distribution issues. This division of effort has led to general weaknesses in distribution abstractions: for example untyped message passing is common (e.g. [7]).

Many agent system designers have tackled the problem of implementing mobile code by basing their systems on scripted languages, simplifying the mobility of code across heterogeneous systems, and allowing control of its execution (e.g. suspension and resumption) (e.g. [8]). The disadvantage of this approach is that introduces new