Distributing Intelligence
Bridging the 'Agent' - 'Agent' Divide

Donie O'Sullivan, Richard Evans, David Kerr,
Intelligent Systems Unit,
Broadcom Éireann Research Ltd.
Kestrel House, Clanwilliam Place,
Dublin 2,
Ireland.

{do,re,dave}@broadcom.ie

Abstract

The term 'agent', which is currently in fashionable use, hides a schism between network enabled technology as represented by Java, and dynamic, adaptable software technologies built upon autonomous, intelligent components. This paper describes a method of distributing agents through a multi-agent system, and which aspects of an agent's functionality are important when choosing its implementation language.

To date, it appears that most commercially available agent development toolkits are based on technologies such as Java (RMI), as opposed to classical AI languages such as Prolog and Lisp (Lange and Chang, 1996). This set of technologies (in common with CORBA) are 'network aware' in that they have mechanisms to support open and standard distributed communication implemented at the language level. These Network aware languages provide the agent toolkit developer with a ready-made route to the creation of distributed multi-agent systems. A consequence of embedding network awareness within the language has been to tie such systems to procedurally based paradigms and inevitably to result in an impoverishment of AI capability.

This contrasts with the area of multi agent systems which has arisen from Distributed AI (Alty et al., 1994) and is built closely on the capabilities of AI and exhibits adaptability and 'Intelligence' through the use of rule bases, machine learning, constraint satisfaction etc. One notable feature of these capabilities is that they are often enabled through the use of declarative languages which treat instructions and data in a similar manner thus allowing programmed behavior to be dynamically modified. In this context Multi Agent Systems refers to the dynamic modification of agent abilities and the ability to reason about and co-operate with other agents in the common environment thus promoting emergent behavior.

A surprising aspect of the current agent scene is the seeming unawareness of this fundamental distinction. Perhaps the use of a common terminology (the term Agent) is responsible.

This paper surveys the characteristics of common agent development tools and technologies and their applicability to the development of intelligent multi-agent systems, and concludes that there is an urgent need to bridge this divide. The key is the provision of an open communications architecture based on a strong distributed computing standard e.g. Common Object Request Broker Architecture (CORBA) and Internet Inter-ORB Protocol (IIOP) (Mowbray and Zahav 1995) into which can be slotted a variety of agent types. This allows the creation of heterogeneous agent systems with the choice of implementation language and the level of intelligence appropriate for each component agent (Kerr et al. 1998). A consequence of this approach is to put agent communication languages center stage as a technology to enable the integration of legacy components within multi-agent systems.

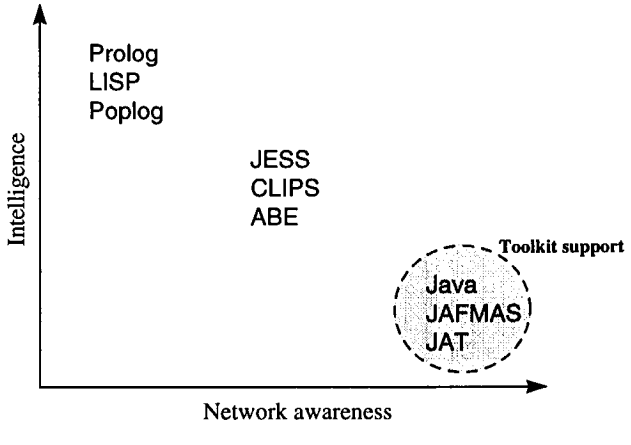

Figure 1: Tradeoff between Intelligence and Network Awareness for different Agent implementation technologies

Figure 1 illustrates the strengths and weaknesses of a number of Agent languages. It shows that a traditional AI

implementation language such as Prolog is strong for implementing an Intelligent Agent, but weak at distributing that agent throughout a network, whereas a language such as Java is strong at achieving distribution, but a weak choice for the development of an Intelligent Agent.

This paper also describes work undertaken by the authors and others towards the realization of a platform for the development of multi-agent systems, with heterogeneous agent behaviors and characteristics (various levels of 'intelligence') within an open communications architecture. Specifically we describe the Agent Services Layer (Somers et al. 1997), which is an agent development platform, built on top of CORBA, which supports the development of multi-agent systems, composed of heterogeneous agents, i.e. agents developed for specific purposes in languages tailored to that purpose. To date there is support for the development of agents in a number of languages including Prolog, CLIPS, JESS, C / C++ and Java.

The FIPA standard for Agent Management (Chiariglione et al. 1997) presents an opportunity to achieve the desired result as it mandates the use of the Inter-ORB Interoperability Protocol (IIOP) as the baseline[1] for interoperability between different agent systems. The need to develop Intelligent Agents which exploit the FIPA Agent Communication Language (ACL) in a sophisticated way and which comply with the FIPA standard for agent management should act as a key motivation for integrating suitable Intelligent Agent development technologies within an open communications architecture, in this case IIOP.

The choice of implementation language is often least thought about when implementing an agent system. A choice is either not available to the programmer (in the case where the multi-agent system does not use any sort of middleware) or else the programmer's habits or current trends dictate the language. A more ideal scenario is where there is no constraint on the implementation language, and each agent in a multi-agent system in analysed at design time, to determine the services it is to provide to the multi-agent system. This analysis will lead to some indication of appropriate implementation language (albeit declarative versus procedural).

As an example, consider an application (Chiariglione et al. 1997b) where the following agents have been identified as forming the backbone of the multi-agent system. (i) User profiling agent (ii) Legacy database agent (iii) travel booking agent. One of the first tasks in the implementation of such a system (after selection of an appropriate agent architecture, which is platform and language independent) is to select the implementation language for each of these agents. The User profiling agent has to handle adaptivity (to profile the user) as well as storing user profiles. The legacy database agent is responsible for providing access to an external data through the use of the ACL. The travel booking agent is responsible for communicating booking requests to the appropriate travel booking platform (which is not under the developers control).

[1] This does not preclude the use of other interoperability protocols, it simply means that IIOP must be supported.

Several key facets of these agent types govern the implementation selection process. Firstly the User profiling agent requires adaptivity, potentially learning, and the ability to provide recommendations to the user. The choice of language should therefore easily allow for these traits, therefore, a suitable language would be declarative (e.g. Prolog or LISP). The choice of implementation language for the legacy database agent should be governed by the API which the legacy database provides to external programs. There is no 'intelligence' necessary, therefore, a suitable language could be Java or C++. Finally the travel booking agent will not necessarily require intelligence, but will require the ability to communicate with other multi-agent systems, so support for a protocol such as IIOP, in addition to a procedural language will be suitable.

The ability to choose the appropriate implementation language for each agent allows the agent programmer to develop the agent in such a manner that its characteristics closely match those of the chosen language. This ability is largely a function of the platform within which the agents reside and therefore the agent platform is possibly the key decision the developer of a multi-agent system should make.

References

Lange, D. B. and Chang, D. T. 1996. IBM Aglets Workbench, Programming Mobile Agents in Java, A White Paper. International Business Machines Inc.

Alty J. L. et al., 1994. ADEPT - Advanced Decision Environment for Process Tasks: Overview and Architecture, Internal Report, QMW College London.

Mowbray, T. J. and Zahavi, R. 1995. The Essential CORBA: Systems Integration Using Distributed Objects, Framingham, Mass.: Object Management Group.

Kerr, D.; O'Sullivan, D.; Evans, R.; Richardson, R. and Somers, F. 1998. Experiences using Intelligent Agent Technologies as a Unifying Approach to Network and Service Management, In Proceedings of IS&N '98, Antwerp, Belgium.

Somers, F.; Evans, R.; Kerr, D. and O'Sullivan, D. 1997. Scalable Low-Latency Network Management Using Intelligent Agents, in Proceedings of ISS '97, Toronto, Canada.

Chiariglione L. et al. 1997. FIPA 97 Specification Part 1: Agent Management. Turin, Italy: Foundation for Intelligent Physical Agents.

Chiariglione L. et al. 1997b. FIPA 97 Specification Part 4: Personal Travel Assistance. Turin, Italy: Foundation for Intelligent Physical Agents.