UNIVERSITY OF SOUTHAMPTON

DEPARTMENT OF ELECTRONICS AND COMPUTER SCIENCE

A MOBILE AGENT ARCHITECTURE FOR DISTRIBUTED INFORMATION MANAGEMENT

Jonathan Dale and David C. DeRoure

University of Southampton

March 1997

Presented at VIM'97

A Mobile Agent Architecture for Distributed Information Management

Jonathan Dale and David C. DeRoure

Multimedia Research Group

Department of Electronics and Computer Science

University of Southampton

UK

{jd94r, dder}@ecs.soton.ac.uk

Large-scale networked environments, such as the Internet, possess the characteristics of distributed data, distributed access and distributed control; this gives the user a powerful mechanism for building and integrating large repositories of distributed information from diverse resources. However, few support tools have been developed to allow the user to take advantage of the distributed nature of their information.

To help address this problem, we advocate the integration of two technologies; distributed information management principles to allow users to create, disseminate, discover and manage their information, and mobile agent technology to provide the flexibility, scalability and dynamism necessary to develop such distributed information management applications.

We present four principles that we have identified as being key to achieving distributed information management and an architecture where mobile agents can move across distributed environments, integrate with local resources and other mobile agents, and communicate their results back to the user. We also describe a number of prototype distributed information management agents that we have developed.

Keywords: distributed information management, mobile agents, internet resource integration, open hypermedia

1. Introduction

There is now wide awareness of the concept of a global multimedia and hypermedia information space. Inter-networked environments, such as the Internet and corporate or institutional intranets, have made more electronic information available to more people than ever before; this is set to progress rapidly with the increasing penetration of information networks into homes. Information, once primarily text-based, has quickly branched into images (including three-dimensional graphics), sound and video clips; audio and video streaming is also being deployed.

Yet networks of information present the electronic information community with a fundamental problem: how is information to be created, published, organised, managed, navigated and searched on a global scale? Information management has only been considered previously on relatively small scales and has dealt primarily with text- and image-based media; for example, libraries, encyclopaedias, thesauri and dictionaries. For the most part, these sources of information have had to be painstakingly compiled and cross-referenced by hand.

Current technology in this area provides relatively few and generally unsophisticated tools for information management. Those tools that do exist tend to be aimed at a specific information resource and lie very much within the area of information discovery and retrieval. If we consider the World Wide Web (WWW) as a case study, the prominent tools that exist are relatively simplistic search mechanisms for retrieving information; few tools exist for keeping track of and managing documents and the hypermedia links between them. Recently, these types of information management activities have been termed *distributed information management* [21].

We believe that mobile agent technology can be employed to build distributed information management tools. Mobile agents provide flexibility and a convenient abstraction upon which to build applications;

they answer distribution and scalability issues through mobility, and their dynamic and modular nature addresses the problem of integration with legacy systems.

In this paper, we present an architecture that supports mobile agents which carry out distributed information management tasks to help the user discover, create, maintain and integrate information which is distributed across heterogeneous protocols, systems and data formats. We have developed a number of prototype distributed information management agents to illustrate the applicability of the mobile agent architecture, which helps the user integrate and manage their data across Internet protocols and between Internet systems such as WWW servers.

In section 2, we present four main tasks that we have identified as distributed information management activities. Section 3 details the mobile agent architecture that we have developed and describes the intended function of each core agent and their interactions. Section 4 describes the realisation of the architecture through a prototype developed in the April language [19] and a number of sample agents that we have implemented. Section 5 presents a discussion on related work and is followed by our conclusions.

2. Distributed Information Management

Distributed information management (DIM) is an initiative being promoted by the UK Technology Foresight panel on IT and Electronics to develop methods for managing change and evolution in distributed systems, particularly targeting multimedia applications across large scale, publicly accessible networks. The issues to be addressed by this programme of work include [21]:

- The management of massive multimedia information stores distributed across large numbers of media-capable servers.
- Managing precision and consistency of information in federated systems.
- The extension of distributed technology to capture the semantic understanding of applications.
- Detached information handling, subsequent arbitration and reconciliation.
- Automatic content-based indexing, analysis and transformation and retrieval of information sources.

The following sections describe four issues which we have identified as being key to achieving successful DIM activities [9]. We see these tasks as being represented by cooperating communities of agents which interact in an intelligent manner to perform tasks, whether initiated by the user, or autonomously and pro-actively as a reaction to changes in the local environment or as a fulfilment of an agent's goal set [28].

2.1. Resource Discovery

The purpose of resource discovery (which includes information retrieval) is to search through distributed information systems that are known to the user and to present other sources of relevant information. Automatic resource discovery tools are needed because there are too many discovery information systems for the user to search manually. However, the searching algorithm must be accurate to ensure that relevant data is not overlooked and that irrelevant data is discarded before it reaches the user.

Another function of resource discovery is resource monitoring: the active task of notifying the user when the contents of a resource change. This is particularly useful if the user is monitoring temporal media, for example stock prices, but can also be used to indicate when a user should revisit information resources.

Most current information retrieval and resource discovery algorithms are text-based in nature, as with search engines for the WWW. These can deal with text documents and textual meta-data, but there is little available support for content-based retrieval of other media.

2.2. Information Integrity

As information becomes distributed across wide areas, information integrity becomes an issue to be addressed [8]. Due to the problems of latency, intermittent connectivity and variable service availability, it is difficult to ensure that consistency updates are made in a timely fashion. Additionally, when considering collaborative working environments, versioning and update control need to be implemented to ensure that alterations are not lost.

In terms of distributed hypermedia systems, for example Microcosm: The Next Generation (TNG) [12] and the WWW, the problems of hypermedia link and document consistency must also be handled. Link consistency deals with ensuring that the integrity of links is maintained, even if the source or destination anchor moves. In most cases, link inconsistency is relatively easy to deal with, since it either involves removing the link (if it is no longer valid), or re-pointing the start or end anchor to the new location.

Document consistency is a much more difficult problem, since it deals with the contents of a document changing and can also imply link inconsistency. If the end anchor of a link points to a keyword in the centre of a given document and that keyword is subsequently deleted, how is this resolved?

Unfortunately, in many such instances of inconsistency, some form of user intervention will be required to ensure that the damage is repaired correctly. In all but the most trivial of cases, consistency algorithms can do little more than highlight the problem for the attention of the user. However, in Open Hypermedia Systems that employ separate link databases, such as TNG, the task of consistency checking is made simpler since interaction only occurs between the link databases. Where link information is embedded in a "closed" hypermedia system, such as the WWW, interaction must occur between all documents within the system.

2.3. Navigation Assistance

Navigation assistance is the process of assisting the user in navigating some form of information resource. This resource could be the information contained within a distributed information system or could be the information generated by a number of resource discovery algorithms. Either way, a good navigation assistance algorithm could be employed to protect the user from information overload.

Oren [22] sums up the role of navigation assistance algorithms as "the human reference librarian who does not comprehend the material in articles being sought, but does understand the conventions of card catalogues, abstract collections, citation indexes and bibliographical references. Because these relations can be made explicitly in hypertext they can be utilised without, for instance, having any deep comprehension of the meaning of any article title."

Wilkins [27] further describes navigation assistance as an algorithm that can balance the following requirements:

- Reduce the number of possible destinations for a user to choose from when too many are offered.
- Expand the number of destinations for a user to choose from if none are apparent.
- Gather and intelligently combine existing sources of information that already exist within the system and present the user with one combined source of information instead of many.

2.4. System Integration

A key aspect of DIM is the ability to manage information resources that are available across heterogeneous networks, heterogeneous platforms and are represented by heterogeneous protocols and data formats. To this end, there is a need for DIM tools to be able to integrate with a wide range of distributed information systems.

However, integration is more than protocol conversion since there is also a semantic problem to overcome. For example, how are hypermedia links translated between the WWW and TNG? If there is more information represented in a TNG link, how is this stored within a WWW link? Furthermore, is it possible to apply links across all distributed information systems? If so, where are these links stored and which system has the responsibility of resolving and managing them?

There are two complementary approaches to system integration: arming DIM tools with the necessary information to be able to converse with multiple distributed information systems, and equipping them with the necessary information to make consistent and meaningful semantic link and data conversions.

The Open Hypermedia Working Group, which meets in association with the ACM Hypertext conferences, is working on an Open Hypermedia Protocol (OHP) to addresses these interoperability issues [7].

2.5. Agents for DIM Tasks

Although agents are a sub-field of AI, agents for the Internet are moving from considering the fundamental questions of what is intelligence and intelligent reasoning and how can it be manifested in a computer? Such Internet agents typically may not have intelligence as their core; they are as intelligent as they need to be to complete the task they have been given. This is a belief held by Brooks [1] when he states that "...intelligence is in the eye of the observer" and that the intelligent behaviour of an agent arises as a result of its interaction with its environment.

The agents that we envisage for representing our DIM tasks are based around a fundamental belief that these agents should help users with tasks that they encounter and perform on an everyday basis. Jennings [15] uses the term *ubiquitous computing* to describe the shift from the computer and its software being just a user-directed processor to more of a delegating partnership.

Therefore, our agents conform to the weak agent classification identified by Wooldridge, et al. [28] and possess the following characteristics:

- *Autonomy*. Once launched with the information describing the bounds and limitations of their tasks, DIM agents should be able to operate independently of and unaided by their user.
- *Social ability*. To effect changes or interrogate their environment, DIM agents must possess the ability to communicate with the outside world.
- *Reactivity*. DIM agents need to be able to perceive their environment and respond to changes to it in a timely fashion.
- *Proactivity*. To help DIM agents to be adaptive to new situations, they need to be able to exhibit proactivity, that is, the ability to effect actions that achieve their goals by taking the initiative.

Where appropriate, another characteristic that we feel our DIM agents should be able to possess is *mobility*, since not all of the resources that an agent needs to access will be within the local environment. Client-server architectures are not sufficient in themselves to yield efficient use of bandwidth, a problem where the Internet is suffering greatly. Consider the case where an agent (client) wishes to retrieve some data from a remote server; if the server does not provide the exact service that the client requires, for example the server only provides low-level services, then the client must make a series of remote calls to obtain the end service. This may result in an overall latency increase and in intermediate information being transmitted across the network which is wasteful and inefficient, especially where large amounts of data are involved. Moreover, if servers attempt to address this problem by introducing more specialised services, then as the number of clients grow so the amount of services required per server becomes infeasible to support.

Harrison, et al. [14] highlights some desirable characteristics of mobile agents:

- *Efficiency*. If an agent can move across networks to the location where resources reside, then network traffic can be reduced since the agent can preprocess data and decide which is the most important information to transfer. This is a crucial aspect when considering users who connect through a low bandwidth link.
- *Persistence*. Once a mobile agent is launched, it should not be reliant on the system that launched it and should not be affected if that node fails. The concept of an agent moving between network nodes gives it the ability to 'survive' and to reach as many resources as possible. This is useful for mobile computer users due to the fact that they can log on, launch an agent, log off and check later on its progress.
- *Peer-to-peer communication*. A failure of the client-server paradigm is the inability of servers to communicate with other servers. Mobile agents are considered to be peer entities and, as such, can adopt whichever stance is most appropriate to their current needs. For example, when a mobile agent is interrogating a resource it takes the role of a client; when another mobile agent wishes to query it, then it becomes a server. This allows for great flexibility in dealing with network entities and distributed resources.
- *Fault tolerance*. In a client-server relationship, the state of the transaction is generally spread over the

client and the server. In the event of a network or server failure during a request, it is difficult for the client to reclaim the situation and re-synchronise with the server because the network connection will have been lost. However, since mobile agents do not need to maintain permanent connections and their state is centralised within themselves, failures are generally easier to deal with.

It is useful to distinguish between two aspects of the mechanism for migration: *transportation* of the program code to the remote location, and *activation* of that code. The former can be achieved in advance of activation, for example by prior installation in a library, and this permits some security issues to be addressed in advance of need and just once. Alternatively it can be achieved at runtime, with the code (and probably associated state information about the agent process) migrating to the remote location and being checked on the fly. Both regimes have a place in distributed information management applications.

The next section goes on to describe the architecture that we have developed to facilitate mobile and static agents for DIM tasks.

3. An Architecture for Mobile Agents

We have designed an architecture for mobile agents [4] to allow them to operate in a heterogeneous, networked environment such as the Internet. Our design is based on an investigation of the requirements of a number of diverse projects in the distributed information management area. It addresses some of the issues raised by Chess [3] regarding the safety and security of systems across the Internet as mobile agents become more prevalent. The architecture is not able to provide a complete solution to all of these issues, due to the fact that some specific solutions are required, but it can assist by being configured to the specific needs of the application domain.

The mobile agent architecture (figure 1) comprises both *static* and *mobile agents*; static agents provide resources and facilities to mobile agents, and mobile agents move between network domains taking

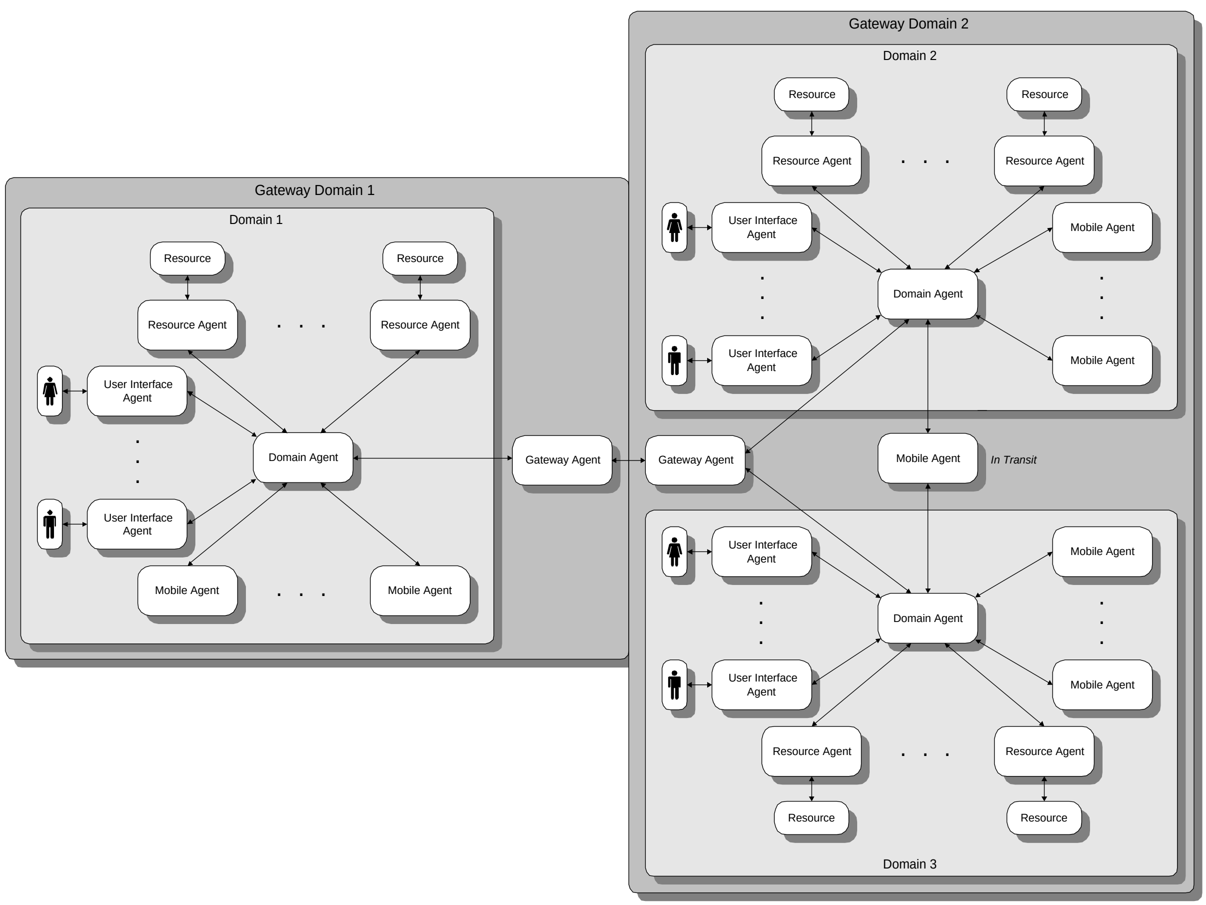

Figure 1 : Core Agents in the Mobile Agent Architecture

advantage of these resources to fulfil their goals. A *domain* is a logical boundary used to delimit nodes, agents and resources into manageable and distinct entities.

It is envisaged that mobile agents will possess the ability to migrate to the most appropriate location (given the constraints of security, cost, distance and other such factors) and will comprise behaviour and some form of knowledge base. The behaviour segment of an agent will describe the reasoning ability of the agent and also its functionality, for example, a resource discovery agent or a link maintenance agent. The knowledge-base will contain the 'intelligence' of the agent, that is, its experiences, its goals and its beliefs. This segment will also contain temporary information that the agent is working upon before it is returned to the user for appraisal.

The next sections take each core agent of the mobile agent architecture and describe its general activities and purpose within the infrastructure.

3.1. Domain Agent

The domain agent is a static agent that coordinates the activities that occur within a domain. The domain agent offers a number of services related to information resources, users and agents within a given domain:

1. It provides a migration service to mobile agents wishing to leave the domain. The domain agent is responsible for negotiating passage to the recipient domain and for ensuring that the mobile agent is transferred successfully. If the migration of a mobile agent is rejected, then it is the duty of the domain agent to restart the agent to allow it to choose a new destination.
2. It authenticates and performs a validation check on mobile agents wishing to enter the domain. Agents that cannot be authenticated or fail the validation check are rejected (see point 1).
3. It launches received mobile agents in a suitable run-time environment which will be related to the amount of trust given to the mobile agent. Agents from trustworthy nodes may be allowed access to more information resources than agents from unknown nodes. A suitable run-time environment will depend upon the access level granted to the mobile agent and the functions that it wishes to perform.
4. It mediates access to information resources at a domain level. When a mobile agent enters a domain, the domain agent performs a security check upon it and allocates the mobile agent a domain-wide permission. This permission can subsequently be used by resource agents to determine whether to allow the mobile agent access to a resource, and by other mobile agents to decide whether to communicate and trade information with the mobile agent (cf. resource mediation in section 3.2).
5. It provides a central registration area within a domain where static and mobile agents can register and advertise their resources and interests. Thus, a domain is also a meeting point which allows agents to gather and share information to resolve their goals.
6. It is the initial point of contact within the domain. All message interactions between static and mobile agents are initially routed through the domain agent, which allows agents to communicate in an asynchronous fashion. However, once two agents have become aware of each other's presence (via the domain agent), they can initiate a direct and synchronous communication between themselves, to transfer high-bandwidth data for example.
7. It ensures that the domain does not become overwhelmed by mobile agents. This can be achieved by limiting the number of agents that can exist within the system at a given time or by implementing a charge for processor cycles and resource access. This way, even if a mobile agent attempted to monopolise a resource, it could only do so for as long as its credit lasted.
8. It advertises (brokers) public information resources and the presence of mobile agents to enquiring agents from both inside and outside of the domain. In this way, mobile agents can interrogate the contents of a particular domain before moving, to ensure that it contains information resources that will help it to achieve its goals. It also provides a mechanism for agents to locate each other.
9. It periodically announces its presence (through the gateway agent, section 3.5) to the larger community of domain agents. This allows each domain agent to obtain a list of other domains that is available for mobile agents to use in determining their next jump point.

The domain agent is the central manager within a domain and is ultimately responsible for ensuring that security is enforced within the domain and that agents can communicate with information resources and each other.

3.2. Resource Agent

Resource agents are static agents that exist within a domain to provide a level of abstraction between an information resource and requesting mobile agents. The purpose of a resource agent is to mediate access to a particular information resource for a mobile agent; the resource agent understands how to access the resource and also understands the permission structures associated with the resource.

As such, a resource agent has a number of functions:

1. It is fully conversant with the protocols of the information resource. The information resource should be completely accessible by mobile agents, so that user intervention directly on the resource is not required. This implies that the user has a set of DIM mobile agents that are able to access, modify, locate and manage their information resources through the respective resource agents.
2. It provides an ontological description for each of the services offered by the resource. These services are the methods by which mobile agents interrogate, update and manage resources.
3. It mediates access to the resource at a resource level. When a mobile agent requests a service, the resource agent can either grant access according to the mobile agent's domain-wide permission (allocated by the domain agent), or it can impose further constraints and checks upon the mobile agent itself (cf. domain mediation in section 3.1).
4. It advertises the presence of the information resource by registering the services available with the domain agent. In this way, mobile agents can be aware of what information resources are present within a domain (by querying the domain agent) before they migrate to a particular domain.

The interaction between resource agents and mobile agents forms the crux of the DIM aspect of the architecture. The flexibility that mobile agents are afforded in accessing these information resources and the manner in which they interpret the results will determine their usefulness to the user.

However, it is not envisaged that information resources are just distributed information systems; they can be any system which presents an external interface through which it communicates or can be accessed. This form of legacy integration allows resource agents to be developed for any type of resource; electronic mail, USENET news, databases, the WWW and so on.

3.3. Mobile Agent

Mobile agents, as their name suggests, are the components within the architecture which can migrate between domains. They are the mechanism by which the users exercise control over their own distributed information resources and gain access to other shared information resources, through the relevant resource agents.

Initially, a user launches a mobile agent within a given domain, which is called the *host domain*. Mobile agents are equipped with a set of goals specific to the user's task that describe the nature and limits of their functionality, for example, a resource discovery agent is a mobile agent with a different goal set than, say, a navigation assistant agent. In addition to the limits on functionality that users place on their mobile agents, it is probable that the mobile agents themselves will encounter limits (in the form of security, authentication, validation and other restrictions) that exist within domains. In some cases, these limits will compromise the goals that have been given to the mobile agent, for example, lack of funds to pay for resources.

The essential functions of mobile agents will be defined by the specific DIM tasks that they are allocated. However, they have the following interactions with the architecture:

1. Mobile agents determine where to migrate to next by initially querying the local domain agent for a list of domains of which it is aware. The mobile agent can then use this information to contact each domain agent and determine which one offers a set of information resources which are compatible with its own goal set.
2. Mobile agents are authenticated by an electronic signature that they carry. This signature may be encrypted but must certainly be verifiable with the host domain of the mobile agent or a third-party

domain. Additionally, mechanisms must be employed to ensure that mobile agents are not compromised during transit or within a domain, and that the behaviour of a given mobile agent is consistent with the behaviour that it was given when it was launched. As it is impossible to guarantee these requirements, since a faulty or malicious agent may be launched from an authentic domain, mobile agents should be controlled as tightly as possible and consequently given as minimum an amount of permission as possible.

3. Mobile agents possess the characteristic of persistence; they use migration as a mechanism to achieve longevity. In this way, they are not reliant on the host domain that launched them. This is particularly useful in nomadic computing, where the user is only connected to the network for short periods of time.
4. Mobile agents may communicate not only with resource agents, but also with other mobile agents to achieve their goals.
5. Mobile agents should transmit the results of their findings and actions regularly to their user on the host domain. This way, the user can ensure that the mobile agent is functioning correctly and it can also help prevent the mobile agent from becoming too large (in terms of size) to move or to process.

Mobile agents are the ultimate effector of change within a DIM environment, and as such, need to be treated cautiously. The possibility for mobile agents to effect changes or gain access to private information must be closely monitored by the domain agent and the individual resource agents. A comprehensive authentication, validation and access permission system is vital to ensure that faulty or malicious agents can do no harm to a system and that legitimate mobile agents can fulfil their goals. However, it is not the function of the architecture to provide a complete solution to the security issue, but it can be augmented by incorporating existing security solutions, such as IPv6 at the network transport layer, data encryption at the presentation layer and electronic signatures at the application layer.

3.4. User Interface Agent

The user interface agent provides a level of abstraction for the user away from the details of the mobile agent architecture. Maes [18] describes an interface agent as "...a *personal assistant* who is collaborating with the user in the same work environment." It is essentially an interface agent that performs the following tasks:

1. It launches mobile agents on behalf of the user and tracks their progress and position.
2. It provides mobile agents with a communication point through which they can return the results of their tasks. User interface agents may need to possess local mobility to allow for node failures, since they need to be executing even when a user is disconnected from the system.
3. It organises and pre-processes information returned from mobile agents into a form that is suitable for the user. This may involve filtering out replicated information, presenting urgent information to the user quickly, rendering information using the tools that the user is most familiar with, etc.
4. It provides domain agents with the authentication credentials of a user's mobile agents. It is the task of the user interface agent to ensure that the requesting entity is itself an authentic domain agent, to make certain that authentication credentials are not given to unauthorised systems.

User interface agents provide the user with a window onto their agents, their status, their results and the mobile agent architecture. The creation of new distributed resource management agents could either be the task of the user to write (or reuse from existing templates with different parameters), or the task of the user interface agent to generate automatically, based upon the interpreted requirements of the user. The latter approach is most desirable, since the user interface agent can then take on more of the user's burden by automating certain tasks given by the user and by anticipating other tasks in advance.

3.5. Gateway Agent

A gateway agent is a static agent that provides access to and from domains. It allows a number of domains, related perhaps by geographic, commercial or other commonalities to be treated as a single domain, called a *gateway domain* (figure 1). Gateway agents offer the following services to the domains within its locale:

1. It provides firewalling for a corporation or organisation where information within the gateway domain is of a protected or secure nature. Mobile agents can be restricted both on entry to and on exit from the collective domain.
2. It equips mobile agents entering a gateway domain with the suitable components to allow the mobile agent to communicate within the collective domain.
3. It provides a hierarchy of domains where queries from domains lower in the hierarchy can be dealt with and resolved by domains higher in the hierarchy. It is envisaged that the hierarchy will be similar in nature, though not restricted to an agent-like DNS, where gateway agents provide access to countries at the root level, regions at the next level and so on down to the constituent domains at the leaves.

Gateway agents are the ideal place at which corporations, authorities and countries can ensure that local ordinances are observed within their domains. Some organisations might need special security precautions to protect confidential information, such as patient's records. They also provide a mechanism for ensuring that undesirable material does not enter the domain and restricted information does not leave the domain.

Through the developed hierarchy of gateway domains, gateway agents add scalability to the system, since messages that are for agents within a given domain or gateway domain only travel within that domain. The burden of dealing with messages to destinations unknown can be spread across the hierarchy, thus alleviating the pressure on individual gateway and domain agents.

4. Infrastructure and Exemplar DIM Agents

With reference to section 3, it can be seen that each agent within the architecture offers some level of modularity and abstraction:

- domain agents abstract security and agent management issues,
- resource agents abstract distributed information resource access and management issues for mobile agents,
- mobile agents abstract DIM task issues for user interface agents,
- user interface agents abstract user interface control, and distributed information resource access and management issues for the user, and,
- gateway agents abstract agent and domain naming and higher-level security issues for domain agents.

These agents provide the facilities and functionality that DIM agents can take advantage of to achieve their tasks. The next sections describe the implementation of the architecture and the realisation of some prototype DIM agents that exist within the architecture to unify a user's access to their distributed information resources.

In order to experiment with this design, we have prototyped the architecture in a programming language called April (Agent Process Interaction Language) [19]. We evaluated and reviewed other agent languages when considering a language for implementing our architecture and agents. These included Agent-TCL [13] and Java with Remote Method Invocation (RMI) [24]. We chose April because it supports the key features necessary for our investigation, namely:

- A uniform communication mechanism based upon the concept of electronic mail. An agent can email a message to another agent on the Internet and the April subsystem handles the internal communication protocol and message transfer.
- A higher order programming discipline which allows April programs, modules, functions and procedures to be first class values within the language. This has the major benefits that it enables April programs to be mobile and to be programmed dynamically across the Internet.
- Referencing through publishable *handles* which allows agents to publish their handles as services through which other agents can communicate without previously being informed of the handle.
- A mechanism for limiting the execution of an agent through the resources it wishes to use, the

messages it wishes to send and the operations it wishes to perform. Moreover, messages between agents can be encrypted and authenticated to ensure that they are from reliable sources and are in an uncorrupted form. These features go some way to ensuring that mobile agents are not faulty or malicious and thus cannot perform harmful actions.

4.1. Core Architecture Agents

Each of the core agents within the architecture comprise the following components:

- a unique Internet name, based upon a model of DNS hierarchies (an April handle),
- a core behaviour set that describes the basic functions available to the agent,
- an extensible behaviour set that describes the services that the agent can perform, and,
- a free-form knowledge base for storing working and long-term information which is preserved across migration.

The key data structure available to all agents for representing communication messages and knowledge bases is the *hierarchical message*. A hierarchical message is an extension to the standard Microcosm flat message [5] and is realised through a tree structure that contains a root node, branch nodes and leaf nodes; the root node is the top of the tree, branch nodes provide pathways through the tree and leaf nodes represent actual data values. Each node in a hierarchical message is represented by a tuple of the form `(tag, value)`, where `tag` is a keyword descriptor for the branch or leaf node and `value` contains either a sub-branch or a data value.

The knowledge base of an agent is represented as a hierarchical message of information that the agent can access and maintain. When a new agent is started, it receives a default knowledge base that contains information about its identity, its goal set and its evaluation criteria. The agent also has branches within the knowledge base for storing temporary data, processed data, history information and outstanding message information. Additionally, the agent can create its own branches to represent and store custom information.

Communication messages are sent between agents as a tuple of the form `(messagetype, hierarchicalmessage)`, where `messagetype` puts `hierarchicalmessage` into a specific context for processing by the receiving agent. A core vocabulary of message types is defined for asking agents to perform standard functions, such examples are `MIGRATE`, `SUSPEND`, `REGISTER` and `QUERY`. Additionally, each message type sent to an agent will evoke a response from the recipient. For example, a `MIGRATE` message will return a message of the type `MIGRATEOK` if the migration of an agent occured successfully or `MIGRATEFAIL` if the migration was unsuccessful.

In this way, all message interactions between agents (apart from direct data streaming) occurs through typed message passing. To make agents flexible and dynamically extensible in the message types that they can receive, typed messages are handled by *message modules*. A message module describes the input and output interface for a particular message type as well as its behaviour, and an agent can load or unload a message module on demand. Each message module must be authenticated and supplied only by the domain agent to prevent agents loading malicious or unauthorised message modules. As a further security measure, agents can have static message modules and dynamic message modules. Static message modules must be loaded when the agent first executes and can neither be overridden by new modules of the same type nor be unloaded. Examples of such static message types are `LOADMODULE` and `UNLOADMODULE`, since these need to be incorruptible to ensure the validity of newly loaded modules and to ensure that the incorrect modules are not unloaded. Dynamic message modules can, as their name implies, be loaded and unloaded dynamically and can be overridden by message modules of the same type. They form the extensible behaviour set of the agent.

Message modules are useful for three reasons. Firstly, they prevent mobile agents from becoming too large in terms of their code size, since they can unload all non-essential modules before they migrate and then reload them once they have migrated and restarted successfully at their destination. Secondly, it gives the agent flexibility over extending its own message set to incorporate new functionality at run-time. And finally, message types with the same signature can be implemented differently in different domains; a fact which need not concern the agent as long as the input and output interface and the functionality of the message module remains the same.

Brokering is a function of the domain agent and is based around the concepts of *service registrations* and *request registrations*. A service registration is an offer of an undertaking that one agent will perform for another agent, perhaps for some consideration of money, information or resource access. A request registration is an offer an undertaking that one agent wishes to be performed by another agent, again perhaps for some consideration. The registrations are made with regard to a particular resource (for example, a WWW server, an FTP server, a database front-end, etc.) and agents wishing to invoke services or fulfil requests can query the domain agents to find compatible agents within the local domain.

4.2. Primitives for DIM Tasks

The DIM tasks that we have identified in section 2 are close to the tasks that a user would wish to use to manage their distributed information resources. For example, they want to discover new WWW resources, they want to manage hypermedia links between documents, they want to be assisted in navigating documents, etc. However, in trying to decompose these high-level tasks into manageable actions for our DIM agents, we have had to consider the objects which can be manipulated within our DIM environment. From this, we have identified two major facets of DIM that we are considering within the current implementation; documents[1] and hypermedia links[2], represented through *document management* and *hypermedia link management*[3].

Document Management Primitives		
	`CREATE_DOC`	*Create a new document*
	`DESTROY_DOC`	*Destroy an existing document*
	`RETRIEVE_DOC`	*Retrieve the contents of an existing document*
	`STORE_DOC`	*Overwrite the contents of an existing document*
	`QUERY_DOC`	*Query the existence of a document*
	`MODIFY_DOC`	*Modify the contents of an existing document*
	`FIND_DOC`	*Find within the contents of an existing document*
Hypermedia Link Management Primitives		
	`CREATE_LINK`	*Create a new link*
	`DESTROY_LINK`	*Destroy an existing link*
	`RETRIEVE_LINK`	*Retrieve existing links from within a link database or document*
	`QUERY_LINK`	*Query the existence of a link*
	`MODIFY_LINK`	*Modify an existing link*
	`ASSERT_LINK`	*Assert the validity of an existing link*
	`FIND_LINK`	*Find existing documents that contain a given link*

Table 1 : Simple Document and Hypermedia Link Management Primitives

Table 1 gives a list of simple primitives that we have identified for these two facets. These primitives are the atomic actions that a DIM agent can perform and provide a consistent interface across all resource agents. If a resource agent is abstracting a resource that can support simple document management, such as FTP, then it will register all of the primitives for document management with the domain agent. Similarly, if the resource can support a hypermedia linking model, such as the WWW, then it will register all of the hypermedia link management primitives. These primitives are registered with the domain agent as *service registrations*, that is, they are public services that any agent can use. An agent wishing to take advantage of a particular service formulates a registration query and sends this to the domain agent; the domain agent interrogates its local registration database and returns a list of the agents that match the query criteria.

---

1 A document is considered to be a collection of data that can be handled by a document management system, such as a word-processed report, a financial spreadsheet, etc.

2 A hypermedia link is considered to be an association between two documents that can be handled by a hypermedia link service, such as the WWW or TNG.

3 There are others that we have considered, such as agent management, presentation management, versioning management, and single and collaborative user management, but we detail the most relevant here.

The registration model requires a core set of attributes (such as resource type, primitive name, agent location, input interface, output interface, etc.) to be present within the service registration message that is sent to the domain agent, but the registering agent can also customise its registration by adding arbitrary information (such as cost of the service, assurances of quality, etc.). Subsequent communication, such as negotiation and data transfer, occurs directly between the agents themselves and not through the domain agent.

The primitives are expressed as message modules that can be loaded by a resource agent that supports a particular facet and DIM agents communicate their intention to take advantage of a service through message passing. Figure 2 shows a sample single-domain environment of core agent and sample DIM agent interaction to achieve DIM tasks set by a user. In this example, there are two resources being abstracted; a WWW server and associated documents, and an FTP server and associated documents. The resource agent protecting the FTP server has only registered the document management primitives with the domain agent, whereas the resource agent protecting the WWW server has registered both the document and hypermedia link management primitives. This has the result that no hypermedia link management message modules are loaded into the FTP resource agent, so it cannot process any messages of those types.

There are three DIM agents operating currently in this domain and they have the following tasks:

- *Resource Discovery Agent*. This agent can search across any type of resource for a given search term within a set of documents. This is useful in discovering new documents that are related to the search term.
- *Link Assertion Agent*. This agent can be given a root hypermedia link to a hypermedia structure and will assert which links point to valid or invalid destinations. This is useful in ensuring that a given set of documents contain valid hypermedia links.
- *Link Update Agent*. This agent can be given a hypermedia link that is to be substituted for another hypermedia link across a set of documents. This is useful in updating hypermedia links that have moved.

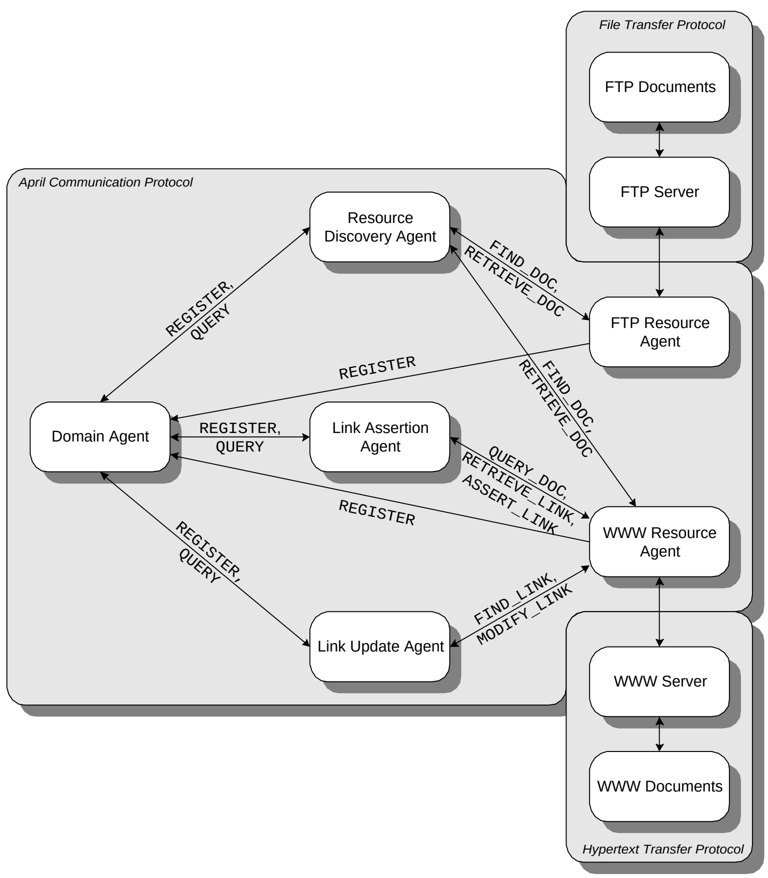

Figure 2 : Interaction between Sample DIM Agents and Sample Resource Agents

As each DIM agent is launched by the domain agent, they register their interests with the domain agent as *request registrations* in the same way that the resource agents registered their services as *service registrations*. This allows agents to be as lazy or proactive as they desire. Proactive resource agents will try to find agent's which have compatible requests that it can fulfil; lazy resource agents will wait until a proactive DIM agent determines that the resource agent has compatible services that it wants to use. For the purposes of this example, we will assume a lazy resource agent but a proactive DIM agent environment.

After registering, the Resource Discovery agent will query the domain agent to determine which document management resources are available within the domain. According to its local registration database, the domain agent will inform the Resource Discovery agent of the locations of the FTP resource agent and the WWW resource agent. The Resource Discovery agent then sends a `FIND_DOC` message to the FTP resource agent and WWW resource agent; this `FIND_DOC` message contains enough information to allow the resource agents to search the contents of the documents that they contain and return a list of files to the Resource Discovery agent. Upon receipt of this list from both resource agents, the Resource Discovery agent sends a `RETRIEVE_DOC` message to each, asking them to fetch the contents of each matched document. The consistent `FIND_DOC` interface across both resources allows the Resource Discovery agent to be unconcerned about the find mechanism employed by the resource agent and the access protocols involved.

The second agent, the Link Assertion agent, queries the domain agent to find any resource agents that support hypermedia linking management and the domain agent only returns the location of the WWW resource agent. Next, the Link Assertion agent sends a `QUERY_DOC` message to determine the existence of the root document that it has been given by the user; if it is present, then the Link Assertion agent sends a `RETRIEVE_LINK` message to the WWW resource agent, indicating that it wants all hypermedia links pertaining to a particular document. The WWW resource agent retrieves these either from a local link database or by consulting the raw HTML of the document and returns a list of all of the links applicable. The Link Assertion agent then sends a `ASSERT_LINK` to the resource agent, asking it to check that each link has a valid destination; valid and invalid links are noted. Once all of the links for a particular document have been tested, the process is recursively repeated upon all local files until all the links have been reached. The consistent `RETRIEVE_LINK` and `ASSERT_LINK` interface presented by the WWW resource agent allows the Link Assertion agent to be unconcerned about how links are stored or resolved.

Finally, the Link Update agent queries the domain agent to find any agents that support hypermedia linking management and the domain agent returns the location of the WWW resource agent. From here, it sends a `FIND_LINK` message to the WWW resource agent, asking for all of the documents which contain the hypermedia link specified by the user to be replaced. The resource agent either consults the local link database or searches through *all* local documents to return a list of documents that contain the link. The Link Update agent then sends a `MODIFY_LINK` message to the WWW resource agent for each document in the list, which updates the links accordingly. The consistent `FIND_LINK` and `MODIFY_LINK` interface presented by the WWW resource agent allows the agent to be unconcerned about how links are represented or modified.

This example scenario shows DIM agents and core architecture agents at work in a single domain, which may be spread over a number of networked machines. In a multi-domain environment, DIM agents can decide to migrate to other domains by asking the domain agent of the local domain for a list of known remote domains. The domain agent can obtain this list from the local gateway agent, which obtains it from the gateway agent above it in the hierarchy, in much the same way as DNS works. A DIM agent equipped with this list can then query the domain agent of each remote domain and determine the set of registered resources that are available; from this the agent can make an informed decision about where to jump to next.

Not all DIM agents will either have the desire or capacity to move; an agent can be restricted in a number of ways. For example, the user that owns the agent may confine it to a particular host or domain, a remote domain might not allow agents to migrate into the domain for security reasons or the agent might not be able to migrate due to execution reasons, for example, the agent is written in an unsupported language. In all of these cases, the DIM agent must resort to a client-server approach to achieving its tasks, but the discovery of remote resources is still through the domain agent.

The architecture that we have developed presents a number of advantages for developing DIM agents.

The DIM primitives provide a dynamic and extensible set of atomic actions which can be implemented according to the functionality level offered by the resource; a `FIND_DOC` document management primitive could be as simple as a text-based search engine or as complex as a content-based retrieval engine which indexes a wide variety of media types. The difference in offered complexity between these two is expressed through the service registration that a particular resource agent makes with the domain agent; more information is made available to the DIM agent for more complex services. For example, the service registration of an FTP text-based search engine would consist of:

```
(REGISTER, [(Resource_name, "FTP"), (Service name, "FIND_DOC"), (Mime_types,
["text/*"]), (Modifiers, ["substring", "regular_expression"])])
```

Similarly, for a WWW content-based retrieval engine that supports sound and images, the two service registrations would be:

```
(REGISTER, [(Resource_name, "WWW"), (Service_name, "FIND_DOC"), (Mime_types,
["audio/*"]), (Modifiers, ["query_by_humming", "sounds_like"])])

(REGISTER, [(Resource_name, "WWW"), (Service_name, "FIND_DOC"), (Mime_types,
["image/*"]), (Modifiers, ["coloured_like", "shaped_like"])])
```

Block registrations can be made where MIME types and supported modifiers overlap; in the text example above, the search modifiers can be applied to all `text` MIME subtypes. Moreover, if multiple MIME types are made in one registration, then the search modifiers can be applied *across* the MIME types specified.

5. Related Work

Many of the agents that we are proposing to undertake DIM tasks already have parallels within the agent community. For example, we envisage the user interface agent to be an expression of the work undertaken by Maes [17] and Lieberman [16] in developing agents that learn from the user and adapt to their needs. Furthermore, Microcosm [6] has shown how an open hypermedia system can integrate with third-party applications in a variety of ways to allow users to associate their preferred browsers and editors with different media types. Microcosm has a concept of *awareness* to represent the level of integration an application can achieve; the more aware an application is, the tighter it can integrate with the open hypermedia system.

Information retrieval agents are being employed to utilise the information that they receive from many different resources. Examples in this field are the Information Retrieval Agent [25] which can search for loosely specified articles from differing document repositories and NetAgent [23] which explores the idea of distributed agents that assist in retrieving relevant and contextual information for the user based upon past searching patterns and experiences.

Telescript [26] is an architecture to support mobile agents for electronic commerce. It is oriented around providing high levels of security, places for agents to meet and mechanisms for allowing agents to communicate. There is a close analogy between places and resource agents, as well as between engines and domains, but the purpose of each in Telescript is not as well defined. Moreover, the Telescript architecture is tied to its particular implementation and platform (to ensure conformity and to enforce security) and has been criticised for not being open and modular. However, Telescript's architecture has begun to tackle the questions of security and resource protection across public access networks by enforcing very stringent restrictions upon mobile agents.

The Frankfurt Mobile Agent Infrastructure (ffMAIN) [20] is a mobile agent architecture based around the Hypertext Transfer Protocol (HTTP) which allows mobile agents to migrate between sites that have been augmented with their agent server. However, as yet, the project has not considered how agents would interact and integrate with their local environment, but there is provision for users to launch agents and monitor their progress.

The itinerant agents system [3] is a conceptual framework for implementing secure, remote applications in situations such as the Internet. There are a number of similarities between the architecture that they describe for supporting mobile agents and the one detailed in this paper; the need for ontologies, high security features, resource access, etc. Unfortunately, the ideas have not yet been developed further.

6. Conclusion

We have described the requirements of Distributed Information Management applications in internet and intranet environments, and presented a solution based upon an agent metaphor. We have characterised the agents and presented an architecture which we believe to be of general application in the DIM context. The design has been prototyped and demonstrated with agents which operate within an internet environment.

The dynamic and modular nature of the DIM agents allows them to be flexible to adapting to new situations and environments; the dynamic and modular nature of the architecture allows it to incorporate new abstractions. For example, to offer a media conversion service for converting between data formats, a conversion resource agent could be written and added dynamically to the system; its service registration would be a document management primitive of the type `CONVERT`.

The approach taken for the design of the architecture, the primitives and the agents is object-oriented in nature; the primitives are a set of methods that can be invoked on one object (agent) by another. The management facets of each primitive set are sub classes of a particular super class, for example, the document management facet relates to a *document class* and the hypermedia link management facet relates to a *hypermedia link class* and both are part of the super class *information object*. In this way, the architecture could be implemented in an object-oriented system.

The DIM primitives themselves help to provide functionality that is not generally available across such publicly accessed networked systems, especially the Internet. The sample DIM agents that we have developed are targeted specifically at areas where Internet-based tools are lacking or are simply not available. Since users are not currently able to exploit the distributed and diverse nature of their information, the DIM agents highlight the need for such similar tools to be made generally available. As more and more information becomes available electronically, users will require the ability to create, modify and manage their documents and hypermedia links wherever they are stored and whatever format.

Finally, by making the hypermedia link a first-class object within the architecture this means that the services of an Open Hypermedia Linking Service (OHLS) could be employed to manage hypermedia links in much the same way that a document management system could be used to manage documents. This has the result that for link-embedding systems such as the WWW, different link management solutions (other than the intrinsic model supported by HTML) could be used; for example, the Distributed Link Server [2]. This would give users a powerful management abstraction over their links and documents.

Future directions of the research presented in this paper lie in developing a high-level scripting language in which users can develop and combine DIM agents built from hierarchies of primitive sets. We are also investigating the use of Knowledge Query and Manipulation Language (KQML) [10] to represent the specification of the communication protocol between agents and Knowledge Interchange Format (KIF) [11] to ease information conversion between different agent systems.

Acknowledgements

This research has been funded by an EPSRC Research Studentship and partially supported by EPSRC Research Grant number GR/K73060. The authors would particularly like to thank the members of the 'Voyager' group for their general contribution to this research and Frank McCabe for his tireless support and help with April.

References

[1] BROOKS, R. A., Intelligence without Reason. *In:* Proceedings of the Twelfth International Joint Conference on Artificial Intelligence, Sydney, Australia, pages 569-595, 1991.

[2] CARR, L. A., DeROURE, D. C., HALL, W. and HILL, G. J., The Distributed Link Service: A Tool for Publishers, Authors and Readers. *In:* Fourth International World Wide Web Conference: The Web Revolution, Boston, Massachusetts, USA, December 1995.

[3] CHESS, D., GROSOF, B., HARRISON, C., LEVINE, D., PARRIS, C. and TSUDIK, G., Itinerant Agents for Mobile Computing, IBM Research Report RC 20010, 1995.

[4] DALE, J., A Mobile Agent Architecture to Support Distributed Resource Information Management, PhD Mini-Thesis, University of Southampton, UK, May 1996.

[5] DAVIS, H. C., HALL, W., HEATH, I., HILL, G. and WILKINS, R. J., Towards an Integrated Information Environment with Open Hypermedia Systems. *In:* Lucarella, D., Nanard, J., Nanard, M. and Paolini, P., *Eds*. ECHT '92, Proceedings of the Fourth ACM Conference on Hypertext, Milan, Italy, ACM Press, pages 181-190, November 1992.

[6] DAVIS, H. C., KNIGHT, S. J. and HALL, W., Light Hypermedia Link Services: A Study of Third Party Application Integration. *In:* Proceedings of the European Conference on Hypermedia Technology, Edinburgh, UK, September 1994.

[7] DAVIS, H. C., LEWIS, A. J. and RIZK, A., A Draft Proposal for a Standard Open Hypermedia Protocol: Revision 1.2. *In:* Proceedings of the 2nd Workshop on Open Hypermedia Systems, California, USA, April 1996.

[8] DAVIS, H. C., Data Integrity Problems in an Open Hypermedia Link Service, PhD Thesis, University of Southampton, UK, 1996.

[9] DeROURE, D. C., HALL, W., DAVIS, H. C. and DALE, J., Agents for Distributed Multimedia Information Management. *In:* Proceedings of the First International Conference on the Practical Application of Agents and Multi-Agent Technology, London, UK, pages 91-102, 1996.

[10] FININ, T., LABROU, Y. and MAYFIELD, J., KQML as an Agent Communication Language. *In:* Software Agents, Bradshaw, J., *Ed.*, MIT Press, 1995.

[11] GENESERETH, M. R. and FIKES, R. E., Knowledge Interchange Format Version 3.0 Reference Manual, Technical Report Logic 92-1, Stanford University, USA, January 1992.

[12] GOOSE, S., DALE, J., HILL, G. J., DeROURE, D. C. and HALL, W., An Open Framework for Integrating Widely Distributed Hypermedia Resources. *In:* Proceedings of the IEEE International Conference on Multimedia Computing and Systems, Hiroshima, Japan, 1996.

[13] GRAY, R. S., Agent TCL: A Transportable Agent System. *In:* Proceedings of the CIKM Workshop on Intelligent Information Agents, Fourth International Conference on Knowledge Management, Mayfield, J. and Finin, T., *Eds.*, Baltimore, USA, 1995.

[14] HARRISON, C. G., CHESS, D. M. and KERSHENBAUM, A., Mobile Agents: Are they a Good Idea?, IBM Research Report, IBM Research Division, 1995.

[15] JENNINGS, N. R., Agent Software, Keynote Speech. *Presented at:* The First International Conference on the Practical Application of Agents and Multi-Agent Technology, London, UK, 1996.

[16] LIEBERMAN, H., Letizia: An Agent that Assists Web Browsing. *In:* International Joint Conference on Artificial Intelligence, Montreal, Canada, August 1995.

[17] MAES, P., Social Interface Agents: Acquiring Competence by Learning from Users and Other Agents. *In:* Software Agents—Papers from the 1994 Spring Symposium, Technical Report SS-94-03, Etzioni, O., *Ed.*, pages 71-78, 1994.

[18] MAES, P., Agents that Reduce Work and Information Overload. *In:* Communications of the ACM, 37(7), pages 31-40, 1995.

[19] McCABE, F. G. and CLARK, K. L., April—Agent Process Interaction Language. *In:* Intelligent Agents, Lecture Notes in Artificial Intelligence, 890, Wooldridge, M. J. and Jennings, N. R., *Eds.*, Springer-Verlag, pages 324-340, 1995.

[20] LINGNAU, A. and DROBNIK, O., An Infrastructure for Mobile Agents: Requirements and Architecture. *In:* Proceedings of the 13th DIS Workshop, Florida, USA, September 1995.

[21] Office of Science and Technology, Technology Foresight Panel on IT and Electronics, Volume 8, HMSO Publications, 1996.

[22] OREN, T., The Architecture of Static Hypertexts. *In:* Proceedings of Hypertext'87, Association for Computing Machinery, pages 291-306, 1987.

[23] PARK, T. and CHON, K., Collaborative Indexing over Networked Information Resources by Distributed Agents. *In*: Distributed Systems Engineering Journal, December, 1995.

[24] Sun Microsystems, Inc., Java Remote Method Invocation Specification, Prebeta Draft revision 1.1, 1996.

[25] VOORHEES, E. M., Software Agents for Information Retrieval. *In:* Software Agents—Papers from the 1994 Spring Symposium, Technical Report SS-94-03, Etzioni, O., *Ed.*, pages 126-129, 1994.

[26] WHITE, J. E., Mobile Agents. *In:* Software Agents, Bradshaw, J., *Ed.*, AAAI Press/MIT Press, 1996.

[27] WILKINS, R. J., The Advisor Agent: A Model for the Dynamic Integration of Navigation Information within an Open Hypermedia System, PhD Thesis, University of Southampton, UK, 1994.

[28] WOOLDRIDGE, M. J. and JENNINGS, N. R., Intelligent Agents: Theory and Practice. *In:* Knowledge Engineering Review, 10(2), June 1995.