Defining Spatial Agents

Armanda Rodrigues
Centro Nacional de Informação Geográfica
Rua Braamcamp 82 5º Esq, 1200 Lisboa, Portugal
armanda@cnig.pt

Jonathan Raper
Department of Geography, Birkbeck College
7/15 Gresse St., London W1P 2LL, UK
j.raper@geography.bbk.ac.uk

ABSTRACT

The concept of agent is becoming increasingly important not only in research (where it has been in use for some time) but also now in commercial applications. However, an agent may represent many d ifferent things according to the people that implement and use them. Based on the concept of agent we define Spatial Agents as agents that make spatial concepts computable. By implementing spatial agents we hope to solve the following problems : Locating and retrieving Spatial Information in large networks (and specifically the Internet), Facilitate the handling of a GIS user interface, Implementing improved spatial tasks and Creating interfaces between GIS and specific software packages. We discuss what are the necessary qualities that a development tool should have to qualify for agent development. We select some of the most prominent tools currently used and try to choose which are best suited for the development of spatial agents. Finally, we reflect on the design of spatial agents that will solve the problems mentioned above and present a prototype of an Interface Agent for the Drawing tool of the Smallworld GIS.

1. INTRODUCTION

The concept of agent is becoming increasingly important not only in research (where it has been in use for some time) but also now in commercial applications. Research on agents emerged initially from Distributed Artificial Intelligence, a branch of Artificial Intelligence that deals with the solution of complex problems by networks of autonomous, cooperating computational processes called agents (Adler and Cottman, 1989). The complexity of agents varies depending on the tasks to perform and their area of expertise. Minsky (1986) argued that very simple agents with very little intelligence are the components of our minds.

Several problems have arisen in the manipulation of spatially-referenced information (and specifically in Geographical Information) where the creation and use of agents may be successful (Rodrigues et al, 1995): Location and retrieval of spatial information in large networks (and specifically the Internet), Facilitating the handling of a GIS user interface, Implementation of improved spatial tasks and Creating interfaces between GIS and specific software packages.

Each of these problems has special constraints and its solution lies with a specific type of agent. Therefore, for each work area we defined the characteristics of the spatial agents to create and its type of agent architecture. By studying current research into agents we have selected the following types of agents as the basis for spatial agent

development: Mobile agents, Interface agents, Deliberative (cognitive) agents and Reactive agents. By combining these type of agents into specific architectures it is possible to create agents to execute spatial searches, to improve GIS user interfaces, to create improved spatial tasks and to connect GIS with specific software packages.

2. DEFINING AGENTS

Many definitions have been given to agents. In fact, the term has been used to "describe everything from a word processor Help system to mobile code that can roam networks to our bidding" (Wayner, 1995). There exist several definitions of agents, given by different researchers, each involving the characteristics most valuable to them (Franklin and Graesser, 1996). We have chosen Wooldridge and Jennings' (1995) because although it is a very general definition it includes key features that will provide an initial classification of agents: "An agent is a self contained problem solving entity implemented in hardware, software or a mixture of the two" that should include the following properties :

- Autonomy : agents should be able to execute their problem solving tasks without direct intervention from humans or other agents. They should, to some degree, control their own actions and internal state.

- Social ability : agents should be able to interact, when they see fit, with other agents and humans, either to complete their problem solving or to help other agents.

- Responsiveness : agents should hold knowledge on their environment and be able to respond to any changes that may happen in this environment.

- Proactiveness : agents should not only act in response to their environment, they should be able to take advantage of fortuitous opportunities to achieve their designated goals. An agent should be able to modify its behaviour in response to stimuli.

Agents can be classified in several ways (read Franklin and Graesser (1996) and Wooldridge and Jennings (1995) for some classifications). Because this paper is concerned with the application of agents to spatial issues, we will discuss only the types of agents that will be useful in this area of expertise: Mobile Agents, Reactive Agents, Cognitive Agents and Interface Agents.

2.1 Mobile Agents

Mobile Agents are programs which may be dispatched from a client computer and transported to a remote server computer for execution (Harrison et al, 1995). It has been suggested that mobile agents offer an important new method of performing transactions and information retrieval in networks. Mobile (Itinerant) agent architectures can be considered an extension to client/server computing in which the client creates additional processes that are sent through the network to search and retrieve the required information. These processes are defined according to the types of interaction they perform (Chess et al, 1995):

- Information Dispersal/Retrieval: Simple interactions based on an ask/receive paradigm between an itinerant agent and a static agent;

- Collaborative: A more complicated type of interaction in which there is a single clearly defined goal. The agents not only ask for and receive information but they also evaluate and compromise according to their preferences;

- Procurement: Very complex interactions governed by an auction protocol in which the agents' goals and resources are hidden from other agents.

The same authors describe in detail one scenario for a Travel Reservation System using itinerant agents. Wayner (1995b) presents mobile agent architectures by defining the underlying technology and the roles given to different players in the architecture: roaming mobile agents, hosts processes that control and enable the agents' activity and resources to be used by the agents. A very important concern in mobile agents is related with security issues: Protecting the host computer from foreign agents attacks or bugs and preventing other users from taking control of arriving agents.

2.2 Deliberative (Cognitive) Agents

Deliberative Agents are agents that contain an explicitly represented, symbolic model of the world and in which decisions are made via symbolic reasoning (Wooldridge, 1995).They have a memory, they build plans of action, they can be selfish or cooperative and they exchange complex messages (Ferrand, 1995).

Wooldridge (1995) presents several efforts within the symbolic AI community to construct deliberative agents. However, these architectures usually come across the following problems:

- The transduction problem: translating the real world into an accurate and adequate symbolic description, in time for that description to be useful;

- The representation/reasoning problem: how to symbolically represent information about complex real-world entities and processes and how to enable agents to reason with this information in time for the results to be useful.

The existing algorithms for symbol manipulation cannot guarantee termination with useful results in an acceptable fixed time bound, essential in agent systems. This led researchers to look into Reactive agent architectures.

2.3 Reactive Agents

Wooldridge (1995) defines reactive agents in a negative fashion (as opposed to deliberative agents). Reactive agents do not include any kind of central symbolic model and do not use complex symbolic reasoning. They are generally very simple structures that act by direct reaction from changes operated in their environment. There is no global control in the system, global behaviour emerges from the local reactive actions of each agent. In section 3, a survey of Multi-reactive-agent systems related to spatial issues is presented. Current developments also show researchers creating hybrid systems using agents that mix cognitive with reactive characteristics (Wooldridge and Jennings, 1995).

2.4 Interface Agents

Interface Agents are semi-intelligent, semi-autonomous systems that assist users when dealing with one or more computer applications (Kozierok and Maes, 1993). The metaphor used is that of a personal assistant collaborating with the user in their work environment. Research into this type of agents has been underway for some time. Maes (1994a) argues that there are several approaches to building Interface agents and defends a Machine Learning approach. She states that under certain conditions, an agent can "program itself". The agent is given some initial knowledge and evolves by learning "behaviour" from the user and from other similar agents. At the MIT Media Lab, several interface agents have been built to help users accomplish tasks of: Electronic mail handling, Meeting scheduling, News filtering and Entertainment selection. For detailed descriptions of characteristics of these agents read Maes (1994a).

3. SPATIAL AGENTS

A Spatial Agent is an autonomous agent that can reason over representations of space. Having a goal with spatial characteristics to fulfil, it must be able to access and/or handle spatial information, reason on it and execute the appropriate tasks. A Spatial Agent will make spatial concepts computable.

Agents with a spatial awareness are a relatively old concept. In Minsky (1986) space is described as "just a society of nearness relations between places" and the global geography of a space as "nothing more than hints about which pairs of points lie near one another". Minsky also states that several layers of agents build the maps inside our brains, each layer being composed of agents that are responsible for regions of the space and whose function is to detect which other agents are the nearest to them. According to Franklin and Graesser (1996), Minsky's Agents may not be considered autonomous agents, concerning the definition we are considering. Minsky's agents are fine grain agents forming a mind as a whole.

The concept of Cellular Automata is also very near to that of agent. "Cellular Automata are mathematical models for systems in which many simple components act together to produce complicated patterns of behaviour" (Wolfram, 1994). Cellular automata are composed of a regular lattice of sites, each site taking on k possible values. The current site is called a cell. The values are updated in discrete time steps according to a rule that depends on the value of the sites situated in some neighbourhood around the cell. Cellular Automata include several of the features that characterise agents although at a simpler scale. The basic unit is the cell. The automaton evolves as the cells change state according to what is happening in their neighbourhood (environment). From the local changes in the state of cells emerges a pattern (goal). Ferrand (1995) quotes work at LAMA-IGA (France) related to the simulation of accessibility of space through the road network from any point. The first attempt to find a solution involved cellular automata but they later moved to Multi-Reactive-Agents Systems (MRAS). This solution can naturally deal with any type of complex diffusion process, like paths crossing above bridges. It can also easily handle a simulation in continuous time, making agents act in different threads of execution. In MRAS, intelligence emerges from the interaction of

multiple simple entities, acting on the base of direct reaction to stimuli (Ferrand 1995). Reactive agents are thus very near the Cellular Automata paradigm, as well as to connectionism. In fact, existing architectures seem to blur the frontier between one concept and the other. Ferrand (1995) also quotes projects related to cartographic generalisation and Multi-criterion Spatial Decision Support where MRAS have been in use. The former aims to "support the creation of maps starting from a set of data, taking into account a thematical focus and graphical constraints linked to the used symbols and the final resolution". The latter concerns single-actor decision making and spatial negotiation support (specifically the choice of a path or an area for a large scale infrastructure like a road or electric line).

At the Santa Fe Institute, the Swarm simulation system was developed in order to provide researchers with a set of standardised simulation tools (Minar et al, 1996). The formalism is that of a collection of independent agents interacting via discrete events. In addition to being containers of agents, swarms can themselves be agents and an agent can also be a swarm. Simulations can, thus, involve several levels of complexity.

Also, Toomey et al (1994) describe a project that aims to implement software agents to help the dissemination of remote sensing data and presents an architecture in which agents communicate with each other in order to locate or, if necessary, produce the information requested by the user. This agent-based Remote Terrestrial Sensing (RTS) data dissemination environment allows the user to specify the desired imagery's geographic region location (by drawing it directly on a world map) and to specify constraints on other image attributes. These constraints are introduced using generic RTS domain terms instead of database specific ones. The Agents in the system are in charge of data sources. A Data Broker Agent leads the communication between agents. These are very different agents from the ones studied by Ferrand. However, they do hold a spatial reasoning component as they have to be able to handle knowledge on the space being portrait in the RTS images as well as on their specific environment of distributed data sources.

3.1 Research Agenda

We have just covered current research on agents with spatial characteristics. However, it is our opinion that the most interesting opportunities for research in spatial agents have yet to be addressed:

- Location and retrieval of Spatial Information in large networks. The growth of the Internet has made huge resources available to everyone. To control and assess these resources, it is necessary to create structures that help locate and retrieve the right information. This problem becomes more intense with spatial information due to the large volumes of data manipulated and the necessity for spatial indexes. There is also an important concern on metadata standards for the publication of spatial data on the Internet. Different agents can be built to locate the necessary information for the user and filter the retrieved information by identifying the users necessities. Spatial Data Mining problems are covered in section 3.1.1;

- Facilitate the handling of a GIS user interface. This thread will use interface agents research to create agents that acquire knowledge on the tasks, habits and preferences of users. These agents should be able to execute tasks on the user's behalf or suggest actions to take. Another possibility of development is the dynamic re-formulating of the GIS user environment according to the user's evolving profile. GIS interface agents are discussed in section 3.1.2;

- Implementation of improved spatial tasks. The use of GIS becomes more difficult as new software releases become available and functionality expands. The large amount of information manipulated is again to be considered. Agent functionality in this area could include the generation of templates for executed tasks, the monitoring of these tasks and the creation of intelligent spatial data that evolves as new information becomes locally/globally available. Spatial Tasks are covered in section 3.1.3;

- Creating interfaces between GIS and specific software packages. This is a consequence of the growing number of fields of application for GIS. The use of spatial models, statistical packages, generalisation algorithms or any other application that adds functionality to GIS is currently very common. Unfortunately the interaction between GIS and these packages is practically non-existent and has to be provided manually by the user. Agents can not only provide for this interface but they can also add power to the packages through personalisation. This issue is discussed in section 3.1.4.

3.1.1 Spatial Data Mining

Agents performing spatial data mining, after receiving the specification of their search, should travel through the network looking for the data the user needs. The best way to implement this kind of agent will, therefore, be to use mobile computing (see section 2.1). In this type of architecture several issues must be considered:

- Server (Host) Implementation - A spatial information server available on the Internet (and/or the World-Wide Web) must publish its information in such a way that agents will be able to find it and recognise its value to their user. Therefore, the structure and publication of the servers' metadata is as important as the data itself. In addition, it is necessary to define concepts common to the agents and the server (Ontologies - Gruber (1993)), creating a common vocabulary that will enable them to communicate and exchange information successfully. Current work into digital libraries is addressing these problems (Communications of the ACM, 1995; IEEE Computer, 1996);

- Client Implementation - The agent should be able to take action on complete as well as incomplete specification of request. A user may know exactly what he wants and where it is held or he may only have a general idea. That is why personalisation is important. The agent can trace back and try to find similar requests on previous situations. Also, if every personal searching agent acquires information on the resources used by its user, then communication and cooperation among agents may provide them valuable information on unknown resources. Using machine learning techniques, it is possible to build knowledge on which agents to ask for help when looking for a specific type of information.

It is also possible to create group profiles so that agents will automatically be able to contact peers that belong to the same group or domain. This type of work has been addressed by the autonomous agents group at MIT Media Lab (Maes, 1994);

- Execution of request - Once the right information or resource has been located there are two ways in which to execute the requested tasks : send an agent to the remote resource location to request the needed service and retrieve the results or have the remote server send the functionality and/or data so that the request can be executed locally. It is obvious that the right choice depends on the request being made (filtering, analysis, queries, etc.). However, the choice lies between the use of mobile agent systems or the development of client applications possibly built using online mapping technology commercially available.

3.1.2 Improving GIS Interfaces

Interface agents will help the user in his/her daily work with the GIS. They will constantly learn the user preferences and profile and build knowledge on the way in which he/she works. As it learns, the agent will suggest automation of tasks to the user. The evolution of the rate of right suggestions will help the user trust the agent. Slowly, the agent will start to execute tasks, on the user's behalf. The agent can also modify the user environment according to the his/her preferred tools or the commands that are executed most often. The user environment will constantly and dynamically evolve. The Open Sesame! Assistant for the Macintosh provides this kind of functionality on the Finder application. It "listens" to the user's commands and builds a profile based on what it has gathered. if a pattern is identified the agent will offer to automate the task the pattern refers to. The user's confidence is taken slowly, as the percentage of "right" suggestions increases. The agent can also build knowledge through Programming by Demonstration (PBD) (Cypher, 1993). An experienced GIS user may be asked to "show" a learning interface agent how it should execute sets of tasks in order to achieve a specific goal. The agent will be told when and where it should start to learn and when the learning process is complete. With the gathered information it should then be capable of repeating the process when the right situation presents itself. One related experience is that of Campos et al (1996), who describe a knowledge-based interface agent for ARC/INFO that receives and processes user's requests in plain English. The agent takes this information and generates sequences of commands that ARC/INFO can understand. If the concepts known to the user are not confirmed by the ARC/INFO database, it interacts with him/her to clarify the misconception. After execution, the agent delivers and presents the results to the user.

3.1.3 Facilitating spatial tasks

This type of agent will be developed in order to integrate proactive spatial processes into complex GIS applications. Simulation environments or Spatial Decision support systems will be the best areas for application of this kind of agent. Depending on the complexity of these processes, these agents will either be reactive or deliberative (cognitive). The need for the manipulation of symbolic contextual information will be a major concern. The information available to the user will autonomously evolve as new data becomes available to the application or as new simulation cycles are

initiated. One example of this type of application is the prototype of the MEGAAOT projects (Rodrigues at al, 1998). An environmental planning application for spatial decision making at the local level is based on a multi-agent control system. The spatial processes involved evolve dynamically as changes are issued into a land use map. Dependencies on processes and data are dynamic links that may or may not be effective at one moment t. The environmental performance of a specific change at a specific time is stored as added experience for later use.

3.1.4 Connecting spatial systems

These agents will serve as Interfaces between GIS and external packages like spatial models, statistical packages, generalisation algorithms, etc. These agents will provide for an input interface for external commands, they will be responsible for communicating with the external package and will finally integrate the results into the GIS data model. It is our belief that this type of agent can range from a very simple process that sends some input to an application, to a (maybe distributed) complex application transparently integrated with the GIS.

4. TOOLS FOR SPATIAL AGENT DEVELOPMENT

Several development tools have become available for the implementation of these agents. Next, we will evaluate some of these tools according to the properties for agent development and the necessities of spatial agents.

4.1 Properties for Agent Development

With the increasing popularity of agents several development tools have emerged with the intention of making agent development easy and natural. There are also several existing tools that have been taken in by researchers and developers for the implementation of agents.

According to the various problem areas in which agents can be useful, and taking into account agent characteristics mentioned in section 2, we can say that tools can be classified according to the following properties:

- Modularity : The ability to create agents that are capable of handling simple, well defined tasks. This provides agent reusability and enables the use of divide and conquer strategies[1] when dealing with complex problems, thus resulting in simpler programming and better overall system architecture. This is a fundamental property associated not only with agent development but with good code in general;

- Interaction among agents[2] : The interface provided for agent communication should be flexible and robust, as run-time communication is not known at design

[1] The use of the term divide-and-conquer does not imply the use of static problem solving modules. These modules are obviously agents themselves subject to the properties of proactiveness and therefore capable of evolving.

[2] The issue of security of information across a network is not considered here as it is not relevant for the discussion.

time. An agent must be prepared to communicate with any other agents, in a location and time-independent fashion;

- Access to distributed resources[2] : An agent's access to data and knowledge, despite their location. In some cases the agent may even be able to change its location in order to better achieve its goals; This is related to the next property

- Mobility[2] : In some domains, agents will have improved efficiency if they are capable of moving in a network. If this is a heterogeneous network, transportability of code is an issue. Agents must be able to change their location and continue to run without needing to recompile;

- Knowledge manipulation: The agent's ability to store knowledge, update the knowledge it holds ("learn") and plan the actions to take considering the changes. The chosen tool should have structures and mechanisms that provide for knowledge representation, Machine Learning and Planning. In the perfect situation the tool would integrate these characteristics has fundamental concepts;

Based on this assessment of agent development characteristics we will now review several tools currently used for agent development. We will be mainly concerned in classifying these tools for their adequacy for the development of the type of spatial agents mentioned above.

4.2 Agent Development Tools

In this section we present a study of the characteristics of several widely distributed programming languages and development tools which are currently being used to build Agents. It is not a comprehensive list. The aim is to identify what makes a tool adequate for (spatial) agent development and not to provide a complete list of tools and languages.

4.2.1 New Tools

Java

The Java language was designed to meet the challenges of application development in the context of network-wide distributed environments (Gosling and McGilton, 1995). It is a very robust object-oriented language (with the exception of primitive data types, everything in Java is an object) as all memory management is executed by the interpreter including automatic garbage collection. Being an object-oriented language, it satisfies the modularity requisite, therefore providing for inheritance, encapsulation and dynamic binding. There is no special feature for communication among agents. However, the messaging mechanisms typical of OO languages and the built-in capacity for multi-threading will facilitate the development of agent communication structures. This is an architecturally neutral language : this means that a program written in Java can be sent to and run on any system where the Java interpreter and run-time system have been installed. The language is the same on any platform. However, it provides no knowledge handling features or machine learning and planning functionality. Recently, several Java-based agent development libraries have been made available, providing the necessary functionality to add knowledge and machine learning capabilities to agents systems. Communication structures between

agents are also widely provided (e.g.: Bits & Pixels Intelligent Agent Library, <http://www.bitpix.com/business/main/bitpix.htm>). The creation of development tools for the creation of mobile Java agents is also becoming commercially available. One example is IBM's Aglets Workbench (Lange and Chang, 1996).

KSE

The Knowledge Sharing Effort (KSE - <http://www.cs.umbc.edu/kse/>) aims to "develop techniques and a methodology for building large-scale knowledge bases which are shareable and reusable" (Finin et al, 1994). KSE aims to provide building blocks for interaction and interoperation, which are the two characteristics they find most desirable in agent systems (Finin et al, 1997). Several tools have been developed in order to achieve this goal:

- KQML (Knowledge Query and Manipulation Language) - This is a message format and a message handling protocol to support run-time knowledge sharing among agents. It is therefore an interface for communication among agents. KQML focuses on an extensible set of performatives, the operations that agents use to communicate.

- KIF (Knowledge Interchange Format) - This "is a formal language for the interchange of knowledge among disparate programs (written by different programmers, at different times, in different languages and so forth)" (ARPA KSE, 1995). KIF has not been built to be an internal representation for knowledge, but to serve as a common format for the interchange of knowledge between programs.

- Ontolingua - "An ontology is an explicit specification of a conceptualisation" (Gruber, 1993). It describes the concepts and relationships that an agent or community of agents have in common. Ontolingua provides descriptions of Ontologies in a form that is compatible with multiple representation languages. The syntax and semantics of Ontolingua definitions are based on KIF. Each ontology defines a set of classes, functions and object constants for a specific domain including contraints for interpretation (Finin et al, 1997).

The KSE tools provide for robustness and flexibility in the communication among agents, even if written in different languages, by different people. They also enable representations of knowledge to be shared by several different systems. However, these are not tools to build agents but to help them communicate and cooperate when trying to achieve their goals.

TCL (and Agent TCL)

TCL stands for "Tool Control Language" and is a simple scripting language for controlling and extending applications. Its popularity comes from being embeddable and extendible. "Its interpreter is implemented as a library of C procedures that can easily be incorporated into applications, and each application can extend the core TCL features with additional commands specific to that application" (Ousterhout, 1993). Although TCL was created for a very different purpose, its internal structure, cost and reputation make it a good candidate for agent programming (Wayner, 1995b): TCL is an interpreted scripting language that encourages modular programs

with many small tools; Tools and their metaprograms pass around TCL scripts in the same way that computers may want to dispatch mobile agents. TCL is not object-oriented but being interpreted and extendible makes it quite easy to add functionality to applications and to execute them in different machines.

Safe-TCL is an extension to TCL that allows for "foreign" agents to run safely on a machine. These "unsafe" agents will only be able to execute specific actions, those that will not endanger the environment in which they are running. Wayner (1995b) states that Safe-TCL contains all the hooks that allow it to read incoming mail and evaluate it as a script in a safe manner. This incoming script will execute without being able to thrash the host computer.

The qualities of TCL for agent programming have led to the development of an (mobile) agent system (Agent TCL-Gray et al, 1997). Agent TCL includes migration, message passing and graphical interface (through Tk) facilities. Security issues are considered through the use of Pretty Good Privacy for encryptation and authentication. Safe-TCL enforces access restriction in the host computer.

Telescript

"Telescript is a set of technologies that provide foundation for electronic messaging, distributed processing and remote programming with communicators, computers, telephones and the networks that link them together" (Knaster, 1995). Included is a programming language with which it is possible to implement and customise powerful messaging systems. Messages in Telescript include agents that can execute tasks in a Telescript-aware network. According to Wayner (1994), Telescript Agents include "built-in intelligence about how to interact with other systems" which is a key advantage if we realise that the language is as computationally powerful as C or BASIC. Of course, in order to have Telescript agents running on your machine and communicating among themselves you must have installed a Telescript engine. Telescript is an object-oriented language enabling mobile agents in a telescript network to interact in a robust way and access data and resources at different locations. As Wayner (1994) says, they hold intelligence on the systems they interface with. However, there are no mechanisms for knowledge handling, machine learning or planning. Recently, General Magic, the company responsible for the creation of Telescript, seems to be redirecting its agent development to the Java and ActiveX platforms (see <http://www.genmagic.com>).

4.2.2 Existing tools

Some of the already existing languages have been adopted for agent development thanks to their inherent capabilities. Some of these were even improved to better support agent development.

Lisp

Lisp is one of the oldest higher-order languages in the computer world that survives through several dialects, each maintaining a part of the initial characteristics of the language. Recently, a committee of Lisp programmers approved the ANSI CLOS (Common Lisp Object System) standard. This standard defines a Dynamic Object-Oriented Language that allows for all the normal OO features and some more. It provides automatic and efficient use of multiple inheritance, automatic memory

management, method dispatching and unlimited scalability. There are even particular implementations that include extensions to the development environment for constructing complex knowledge-based systems. These extensions have been created to help construct applications but they are not part of the language itself. Lisp has been considered as a good language for developing agents because programs and data are stored in trees built from nested lists. A program can build a structure and then execute it. This becomes important in the implementation of learning agents that need to build new algorithms for automation of tasks. Also, some of the implementations of Lisp are interpreted, allowing for the development of transportable code (and maybe mobile agents). At the MIT Media Lab, the Autonomous Agents group used Lisp to build their learning agents.

It is clear that the modularity property of our list is fulfilled by ANSI CLOS. The knowledge handling property is partly provided by a body of extensions developed in specific implementations. There is, however, no support of network based functionality.

Smalltalk Agents

Being another object-oriented programming language (where literally everything is an object) Smalltalk supports inheritance, class and instance behaviour, dynamic binding, messaging and garbage collection. Its OO nature provides the modularity property and the messaging for writing agents that communicate with each other. To improve its capacity for agent development, Quasar Knowledge Systems has developed SmaltalkAgents, a full-feature object-oriented authoring environment which includes tools for "designing, developing and managing frameworks, classes and objects for use in the development of sophisticated components and agents" (Quasar Knowledge Systems, 1995). SmalltalkAgents includes: Visual, drag-and-drop rapid application development environment for user-interface layout and component design; Native windows components, multi-threading, and exception handling; Easy integration with external (C, C++, Java, Visual Basic) code and data structures and and an extensive library of Smalltalk classes. According to Wayner(1995b) SmalltalkAgents is intended to make it simple for programmers to develop applications that run on a distributed network of machines with different CPU's. Quasar has created a low-level device independent language and all of SmalltalkAgents programs are effectively compiled into this language. In this way, their programs can be run on any machine that supports this low-level language without needing to recompile.

CLIPS

CLIPS "is a productive development and delivery expert system tool which provides a complete environment for the construction of rule and/or object based expert systems" (NASA, 1995). CLIPS supports three programming paradigms : rule-based (knowledge is represented as heuristics), object-oriented (complex systems are modelled as modular components) and procedural (capabilities similar to those provided by C, Pascal, Lisp). There are some current efforts to create extensions to CLIPS adding functionality for producing agent systems. DYNACLIPS (Cengeloglu et al, 1994) is a set of blackboard, dynamic knowledge exchange and agent tools for CLIPS. Agents communicate through the blackboard sending and receiving facts,

rules and commands. Another example of an agent tool based on CLIPS is AGENT_CLIPS (<http://users.aimnet.com/~yilsoft/software/agentclips/agentclips.html>). The agents created in AGENT_CLIPS are rule-based agents which can scan newsgroups and web pages looking for information. They can also exchange knowledge with other agents (facts and rules) at runtime.

These are cases of development tools that integrate at its root the possibility of knowledge representation with object-oriented and rule-based programming. The OO features provide for modularity and the fact that it was built primarily as an expert system development tool make it easier to solve the Knowledge Engineering problems.

4.2.3 Why use only one language?

Because of the wide use that the term agent is given today there is no one language that can on its own take the burden of agent development. As we have seen each tool has its own qualities and the most complex agents may be built from getting all of these strong features together. However, some tools are quite complete for developing one specific type of agent. Some agent tools developers try to provide an environment as complete as possible for the kind of agent they want to specialise in. The problem is always making a complete specification of the requirements for the system we aim to build.

	JAVA	Telescript	Tcl	SmallTalk Agents	KSE	Lisp	CLIPS
Modularity	*	*	+	*	-	+	*
Robustness and flexibility in communication among agents	#	*	#	*	*	-	#
Mobility	+	*	#	-	-	+	-
Distributed data and resources	*	*	*	*	*	+	*
Knowledge	#	-	-	-	*	+	*

Note : *- The tool supports the property completely; +- The tool has some support to the property; # - provided through extensions to the tool;

Table 1 : Classification of development tools according to the relevant properties

Table 1 tries to classify the studied development tools according to the properties specified in section 4.1. Satisfying the modularity property are all the Object-Oriented languages : Java, Telescript, SmalltalkAgents and CLIPS. TCL is classified as having some support of the property because although it is not OO, it encourages the development of modular code. Only some implementations of Lisp are OO so we classify it, too, as only providing some support of modularity. Robust and flexible communication among agents is provided by Telescript, SmalltalkAgents and the KSE tools. In fact, the major aim for the development of the KSE tools was providing sharing of information between agents. Extensions to Java, Tcl, and CLIPS also enable communication among agents. The only language that explicitly implements functions for mobile agents is Telescript (but only inside a Telescript-aware network)

although TCL extensions (Safe-TCL and Agent TCL) also provide some facilities for mobility. Access to distributed data and resources is given by any interpreted language. Any program written in an interpreted language (as some implementations of Lisp) will be able to access information in hosts where the its interpreter resides. SmalltalkAgents and Java are not interpreted but their code is generated into low-level device independent languages so their programs may run wherever those languages are understood. The KSE tools are not used to create agents but they make heterogeneous agents understand each other.

We conclude that Telescript and TCL (especially Agent TCL) are the best choices for implementing mobile Agents. However, this can also be done with Java. Interface agents will be best implemented when using Lisp, as this language enables the creation of new algorithms resulting from the learning of the agent. Lisp will also be a good choice for Deliberative agents, as lists can be used to store symbolic knowledge. Extensions to CLIPS can be used for the development of either Interface and Deliberative Agents. Reactive agents do not have major constraints on languages so we will not consider any best choices. The KSE tools are fundamental in making heterogeneous agents communicate.

4.2.4 How to implement Spatial Agents ?

To implement Spatial data mining agents is more than creating mobile agents. It is necessary to choose tools for implementing the server, for creating metadata knowledge, for the implementation of the mobile searching agents, for interface agents (specification of requests) and for the communication among agents. Therefore, this architecture will probably need mobile agents, interface agents and deliberative agents.

When Implementing GIS Interface Agents there is a need for integrating GIS code with a learning algorithm probably implemented in Lisp or another language with the similar properties. Monitoring events in the GIS and creating new commands for automating tasks will have to be transparently executed by the GIS but controlled by the agent code.

Agents facilitating Spatial tasks will either be deliberative or reactive. In the deliberative case the concern with the manipulation of symbolic knowledge suggests the use of Lisp or DYNACLIPS, although intelligent agent libraries for Java can also be used. Reactive agents leave most possibilities open.

Finally, connection agents are an open issue. Because of the amazing amount of possibilities for their implementation, we will not suggest any agent architectures or tools. This will have to be decided according to the problems to solve.

This analysis of spatial architectures and tools does not mean to be complete. The objective was to look into some possibilities and choose possible paths to take. In the next section we will present a prototype of a GIS Interface agent that has been developed for the Smallworld GIS drawing tool. It is a very simple prototype but its implementation was very useful for the authors because it raised some additional questions in the development of Spatial Agents.

5. A PROTOTYPE OF A GIS INTERFACE AGENT

5.1 Agent Architecture

In collaboration with SmallworldWide, Cambridge, an Interface Agent Architecture for the drawing and plotting tool of the Smallworld GIS was developed. Smallworld GIS is an Object-Oriented GIS which was developed in an OO language called Magik. This language is part of Smallworld and most of the code that forms the GIS is available for the use to make changes. It is, therefore, very easy and natural to include monitoring structures, and add additional features to the code. Because this initial agent does not include any reasoning, it was decided that Magik was sufficient to build it. The Agent Architecture is composed of two types of agent (fig. 1):

- The Agent Controller (fig. 2) - the agent that monitors the GIS;

- The Task Agents - Each task agent automates a specific tool of the GIS and/or helps the user learn how to use this tool. In this prototype the Task agent is only one (the drawing agent) but the final objective is to have several task agents, each for every GIS tool.

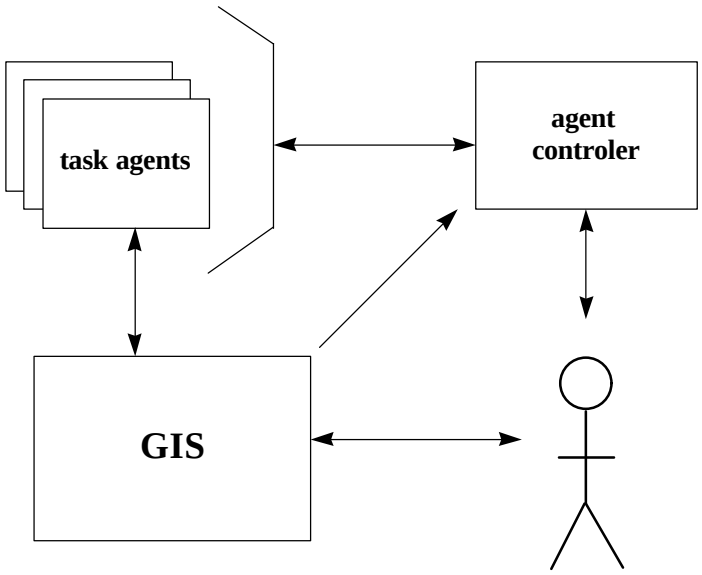

Fig. 1 - Agent Architecture

The Agent Controller monitors every event that occurs in the GIS and channels the relevant information to the appropriate task agent (currently the appropriate agent is the only one that has been developed - the drawing agent).

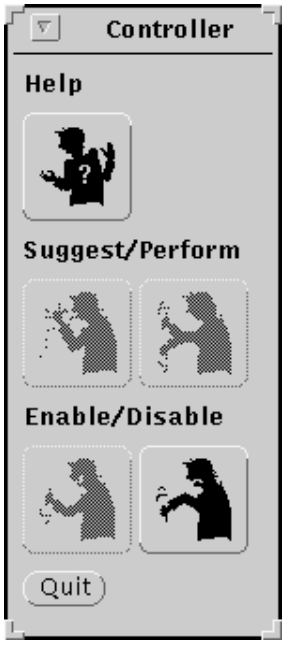

Fig. 2 - The Agent Controller user interface

The task (drawing) agent receives event information from the controller and uses that information to review its state and re-assess the possibilities ahead. Afterwards, it will communicate to the controller its availability either to suggest or perform further actions. The Agent Controller includes a menu with six possible functions. These functions are presented to the user through icons that may be enabled or disabled, depending on the last communication with the drawing agent. The controller functions are the following:

- Help - The Smallworld GIS manuals have all been converted into HTML form. Therefore, it is possible, at any moment, to access the specific HTML page related to the part of the drawing tool that is being used. This icon is always enabled during the life of the controller;

- Suggest - The enabling of this icon by the controller means that the drawing agent knows which is the state of the work and that it can suggest further actions to the user. If the user decides to click on this icon, directions will be provided;

- Perform - The enabling of this icon means, not only that the drawing agents knows the state of the work but also that it holds all the information to carry the execution of the function through to the end. If the icon is used, the current drawing function will be executed using the parameters that have already been entered;

- Enable - Enables the controller to listen to events occurring in the GIS;

- Disable - Disables the controller. After the use of this icon the controller is “asleep” and will not communicate either with the GIS or with the drawing agent. The Enable icon will “awake” it and put it to work again;

- Quit - kills the controller.

5.2 Further developments

This prototype architecture is hardwired to the GIS and the agent has been specifically created for the Smallworld drawing tool. There is no learning mechanism and no intelligence attached to the system. It works quite well because this is a part of the system that, although very repetitive, needs very little personalisation. We would like to create more abstract task agents that would learn from the user how to automate different tasks (preferably spatial tasks). The aim is to create spatial task agents that can reason over spatial processes or data. This type of agent will need knowledge based structures including spatial knowledge as well as machine learning mechanisms.

6. ACKNOWLEDGEMENTS

This research is funded by PRAXIS XXI and JNICT through the fellowship PRAXIS XXI/BD/2920/94.

The authors would like to thank:

- Miguel Capitão of OBLOG Software as one of the authors of the initial version of this paper;

- Winton Davies of the Department of Computer Science, University of Aberdeen, Scotland, for reading earlier drafts of this paper and supplying helpful comments.

- Gillian Kendrick and all the Corporate Development and Product Marketing staff at SmallworldWide, Cambridge for their help during the development of the drawing tool agent prototype.

7. BIBLIOGRAPHY

Adler, R. M., Cottman, B. H.(1989), A development Framework for Distributed Artifi cial Intelligence, Proceedings of The Fifth Conference on Artificial Intelligence Applications, IEEE, 1989.

ARPA KSE (1995), KIF Manual, ARPA Knowledge Sharing Effort, 1995, <http://www.cs.umbc.edu/kse/kif/ >.

Batty, M, Xie, Y. (1994), From cells to cities, Environment and Planning B: Planning and Design, vol 21, pp s31-s48, 1994.

Cengeloglu, Y., Khajenoori, S., Linton, D. (1994), A Framework for Dynamic Knowledge Exchange Among Intelligent Agents, American Association for Artificial Intelligence (AAAI) Fall Symposium, November 1994.

Chess, D., Grosof, B., Harrison, C., Levine, D., Parris, C., Tsudik, G. (1995), Itinerant Agents for Mobile Computing, IEEE Personal Communications, Volume 2, Number 5, pp. 34-49, October 1995.

Communications of the ACM (1995), *Special Issue on Digital Libraries*, 1995.

Cypher, A. (1993), Watch What I Do : Programming by Demonstration, The MIT Press, Cambridge, Massachusetts, 1993.

Ferrand, N. (1995), Multi-Reactive-Agents Paradigm for Spatial Modelling, Proceedings of the GISDATA Workshop on Spatial Modelling, Stockholm, June 1995.

Finin, T., Fritzson, R., McKay, D., McEnitr, R. (1994), KQML - A Language and Protocol for Knowledge and Information Exchange, in Fuchi, K., Yokoi, T. (Eds.), Knowledge Building and Knowledge Sharing, Ohmsha and IOS Press, 1994.

Finin, T., Labrou, Y., Mayfield, J. (1997), KQML as an agent communication language, in Bradshaw, J. (Ed.), "Software Agents", MIT Press, Cambridge, USA, 1997.

Franklin, S., Graesser, A. (1996), Is it an Agent, or just a Program ?: A Taxonomy for Autonomous Agents, <http://www.msci.memphis.edu/~franklin/AgentProg.html>, Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages, Springer-Verlag, 1996.

Gosling, J., McGilton, H. (1995), The Java Language Environment: A White Paper, 1995, <http://java.sun.com/docs/white/langenv/>.

Gray, R., Kotz, D., Cybenko, G. and Rus, D. (1997), Agent Tcl, Cockayne, W. and Zyda, M. (Eds.), Itinerant Agents: Explanations and Examples with {CD-ROM}. Manning Publishing, 1997, Imprints by Manning Publishing and Prentice Hall. To appear.

Gray, R.S. (1995), Agent TCL: A Transportable agent system, Proceedings of the CIKM Workshop on Intelligent Information Agents, Fourth International Conference on Information and Knowledge Management (CIKM'95), Baltimore, Maryland, December 1995.

Gruber, T. R. (1993), A Translation Approach to Portable Ontology Specifications, Technical Report KSL 92-71, Knowledge Systems Laboratory, Computer Science Department, Stanford University, September 1992, Revised April 1993.

Harrison, C.G., Chess, D.M., Kershenbaum, A. (1995), Mobile Agents: Are they a good idea ?, Research Report, IBM Research Division, T. J. Watson Research Center, Yorktown Heights, NY 10598, 1995.

Huhns, M. N. (1987), Ed., Distributed Artificial Intelligence, Morgan Kaufmann, Los Altos, CA, 1987.

IEEE Computer (1996), *Special Issue on the Digital Library Initiative*, May 1996.

Jennings, N. R. (1995), Agent Software, Proceedings of The Agent Software Seminar, UNICOM Seminars, London, England, 25-26 April, 1995.

Knaster, S. (1995), Magic Cap Concepts, Development Release 1, <http:// www.genmagic.com>, 10th September 1995.

Kozierok, R., Maes, P. (1993), A learning interface agent for scheduling meetings, Proceedings of the ACM-SIGCHI International Workshop on Intelligent User Interfaces, Florida, January 1993.

Lange, D. and Chang, D. (1996), Programming Mobile Agents in Java: A White Paper, September, 1996, < http://www.trl.ibm.co.jp/aglets/whitepaper.htm >.

Maes, P. (1994), Agents that Reduce Work and Information Overload, Communications of the ACM, Volume 37, Number 7, July 1994.

Maes, P. (1994a), Modelling Adaptive Autonomous Agents, Artificial Life, Volume 1, Number 1/2, pp. 135-162, 1994.

Minar, N., Burkhart, R., Langton, C., Askenazi, M. (1996), The Swarm Simulation System: A Toolkit for Building Multi-Agent Simulations, 1996, <http://www.santafe.edu/projects/swarm/overview.ps>.

Minsky, M. (1986), The Society of Mind, Touchstone, Simon & Schuster, Inc., New York, 1986.

NASA, CLIPS (1995), A Tool for building expert systems, NASA Johnson Space Center, 1995, <http://www.jsc.nasa.gov/~clips/CLIPS.html>

Ousterhout, J. K. (1993), Tcl and Tk Toolkit, Addison-Wesley Publishing Company, 1993.

Quasar Knowledge Systems (1995), Smalltalk Agents, 1995, <http://www.qks.com>.

Rodrigues, A., Raper, J., Capitão (1995), M., Implementing Intelligent Agents for Spatial Information, Proceedings of The Joint European Conference on Geographical Information 1995, The Hague, The Netherlands, 26-31 March, 1995.

Rodrigues, A., Grueau, C., Raper, J., Neves, N. (1998), Environmental Planning using Spatial Agents, Innovations in GIS 5, Taylor & Francis, to appear.

Toomey, C. N., Simoudis, E., Johnson, R. W., Mark, W. S. (1994), Software Agents for the Dissemination of Remote Terrestrial Sensing Data, Proceedings of the Third International Symposium on Artificial Intelligence, Robotics and Automation for Space (i-SAIRAS 94), NASA Jet Propulsion Laboratory, Pasadena, California, 18-20 April, 1994.

Wayner, P. (1994), Agents Away, Byte, May 1994, pp 113-118.

Wayner, P. (1995), Agents of Change, Byte, March 1995, pg.95.

Wayner, P. (1995a), Free Agents, Byte, March 1995, pp 105-114.

Wayner, P. (1995b), Agents Unleashed: A Public Domain look at Agent Technology, AP Professional, MA 02167, Boston, 1995.

Wolfram, S. (1994), Two-Dimensional Cellular Automata, Cellular Automata and Complexity: Collected Papers, Addison-Wesley Publishing Company, 1994.

Wooldridge, M. (1995), Conceptualising and Developing Agents, Proceedings of The Agent Software Seminar, UNICOM Seminars, London, 25-26th April, 1995.

Wooldridge, M., Jennings, N. R. (1995), Intelligent Agents: Theory and Practice, Knowledge Engineering Review, 1995, Volume 10, Number 2, 1995.