Mobile Agent-Based Systems: an Alternative Paradigm for Distributed Systems Development

Paulo César de Oliveira **Eleri Cardozo**

e-mail: paulo@cnptia.embrapa.br eleri@dca.fee.unicamp.br

Departamento de Engenharia de Computação e Automação Industrial
Faculdade de Engenharia Elétrica e de Computação
Universidade Estadual de Campinas – UNICAMP
C.P. 6101 – CEP 13081-970 – Campinas – SP

Resumo

Sistemas baseados em agentes (SBA) se caracterizam como uma abordagem alternativa para o desenvolvimento de sistemas distribuídos. Embora os termos *agente* e *sistema baseado em agentes* sejam amplamente utilizados, não há uma definição de consenso para eles. Este artigo apresenta conceitos e um conjunto de características relacionadas a agentes. A característica de mobilidade, aliada às características fundamentais de agentes, torna *sistemas baseados em agentes móveis* (SBAM) uma abordagem singular para o desenvolvimento de sistemas distribuídos. Aspectos relativos a infraestruturas de suporte a sistemas baseados em agentes móveis são também tratados neste artigo. Finalmente, um conjunto de infraestruturas existentes é examinado tendo como base alguns dos aspectos tratados.

Abstract

Agent-based systems (ABS) constitute an alternative to distributed systems development. The terms *agent* and *agent-based system*, although widely used, do not have yet a consensus definition. This paper presents concepts and a set of characteristics concerning *agenthood*. Mobility, in addition to the fundamental characteristics of agenthood, makes *mobile agent-based systems* (MABS) a unique approach to distributed systems development. Issues regarding frameworks that support mobile agent-based systems are also addressed in this paper. Finally, a set of existing frameworks is assessed based on some of the addressed issues.

1. Introduction

The development of computer networking has provided an underlying basis for the increasing deployment of networked – or distributed – systems. In the early 1990s, corporations relied on networks, distributed information systems and workgroup computing to support the automation of their business processes. Meanwhile, outside the corporate walls, a smashing phenomenon – the Internet – has emerged, providing home users and corporations with a global infrastructure for information sharing and retrieval, and also a common electronic market.

The wide utilization of distributed systems has led to a strong demand for development paradigms which can meet requirements such as:

- *rapid development and maintenance*: the constant evolution in the computer science field and the frequent changes of user's requirements demand timely development and maintenance of computer systems;
- *platform independence*: distributed systems must run on different hardware and operating systems platforms; they must be developed once and deployed in the desired platforms transparently;
- *ease of integration*: systems must adopt open standards in order to be able to easily interact with existing peers;
- *ease of use*: an increasing number of users, with different degrees of expertise, are using networked systems and demand user friendly and personalizable interfaces;
- *flexibility*: systems must rely on flexible architectures with well defined interactions between their components; systems components must not assume only fixed roles, such as clients and servers, producers and consumers;
- *efficiency*: systems must use efficiently the available and probably scarce computer and network resources;

- *support to mobile computing*: as a consequence of technological advances that have made mobile computing available to an increasing number of users.

Since the early 1990s, the Client/Server model has been widely adopted for distributed systems development. In that model, systems are partitioned into components – clients and servers – that assume fixed roles. In most cases, client components must run on different platforms and they are implemented separately for each operating environment, increasing the development and maintenance effort and costs. Furthermore, Remote Procedure Call (RPC) is the most used mechanism for communication in Client/Server systems. Like the local procedure call, it is based on synchronous connections between a client and a server, where the client maintains the entire process state until it receives the return RPC from the server. This leads to an inefficient utilization of computer and network resources and increases networking costs.

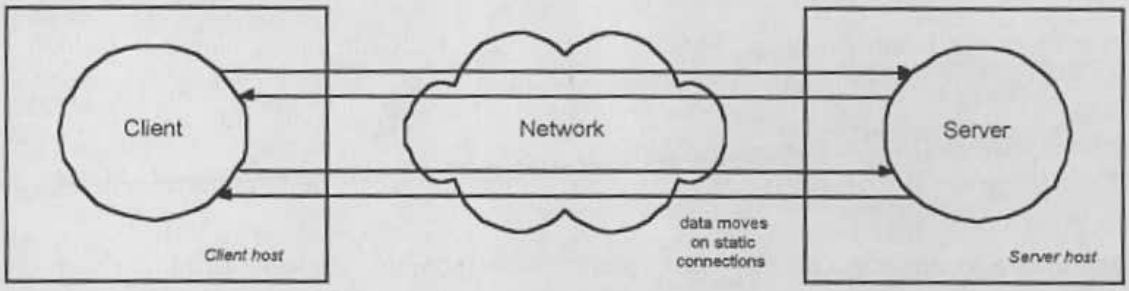

Figure 1: Client/Server components communication via RPC

The agent-based approach constitute an alternative to distributed systems development. Although the terms *agent* and *agent-based system* (ABS) are widely used by researchers in different areas such as distributed systems, software engineering and artificial intelligence, there is no consensus about their definition.

In this paper, agents are considered as *autonomous programs that act on behalf of their owners – users or other programs*. In order to accomplish tasks assigned by their owners, *agents may communicate with other agents, with their environment and with their owners*. This definition focuses on the following fundamental characteristics of *agenthood*:

- *delegation*: users or other programs can delegate tasks to an agent and vest it with authority to act on their behalf;
- *autonomy*: an agent can make its own decisions, based on its owner's statement of goals, preferences and policies;
- *communication*: the ability agents have to interact with their peers, with the environment that hosts them, and with their owner;
- *flexibility*: agents do not assume fixed roles; they may act like clients, servers, observers, etc., depending on their current needs;
- *equity*: agents act like peers, there is no hierarchical relationship among them.

The characteristics pointed out as fundamental reflect the vision that an agent-based system is composed of autonomous entities – agents – that perform tasks assigned – delegated – to them by their owners. Each entity plays flexible roles according to their current needs and is seen as a peer by the other entities that compose the system. Partitioning a system into agents enforces modularity and encapsulation. Parallelism can be also explored.

Although not considered as fundamental, the characteristics described below are strongly related to agents:

- *cooperation*: agents can act in a collaborative manner in order to accomplish common goals;
- *intelligence*: the ability agents have to reason and learn from the interactions with other agents, users and the environment that hosts them;
- *mobility*: agents can move across heterogeneous computer networks, aiming to progressively accomplish tasks that were assigned to them.

Mobility, in addition to the fundamental characteristics of agenthood, makes mobile agent-based systems (MABS) a unique approach to distributed systems development. Unlike RPC, data and programs move over the network.

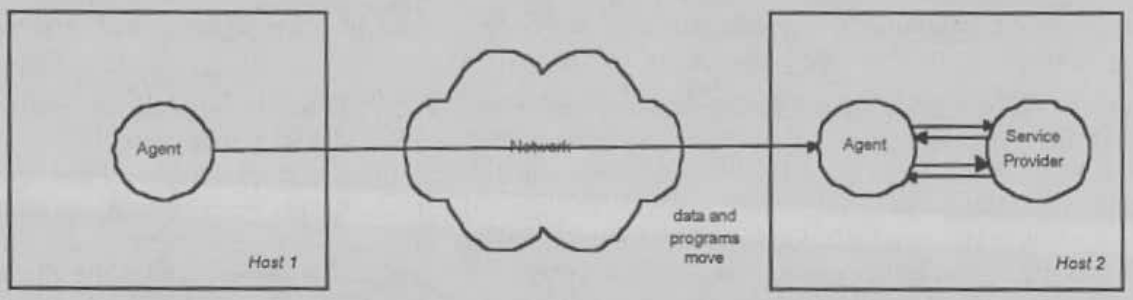

Figure 2: Mobile agent migration for remote resources utilization

Since mobile agents move across heterogeneous networks, their code must run in an identical fashion on every host they can move to [Ling95]. Thus, mobile agents code must be platform independent.

MABS provide the following advantages:

- *intuitive structuring for distributed systems*: systems components are autonomous entities, with their own *"lives"* [ANSA95];
- *flexible distributed computing architecture*: systems components do not assume fixed roles [Nwan96];
- *asynchronous computing*: mobile agents may be launched while their owners do something else [Nwan96];
- *reduced communication costs*: agents move to the host where the desired resources are placed in order to complete tasks, thus it is not necessary to keep static connections between service requesters and providers [ANSA95][Nwan96];
- *support for mobile computers and lightweight devices* (such as Personal Digital Assistants – PDAs) [Ches95];
- *support for disconnected operation*: mobile computers and lightweight devices are intermittently connected to a network. Client applications running on those devices can formulate an agent containing a request for a service, possibly while disconnected, launch the agent during a brief connection session, and immediately disconnect. The response may come in a later connection [ANSA95] [Harr95] [Nwan96];
- *support for applications running on computers with limited local resources* [ANSA95] [Harr95][Nwan96].

The mobile agent-based approach meets the requirements for distributed systems development paradigms mentioned earlier in a more comprehensive manner than the Client/Server model. Platform independence is a basic attribute of mobile agents, since they may be executed on any host on heterogeneous networks. The flexibility requirement, on the contrary to the Client/Server model, is met since agents do not assume fixed roles. Asynchronous computing is supported by agents, leading to an efficient utilization of computer and network resources and reducing communication costs. Support for mobile computing is provided in a broader and more efficient manner through mobile agents.

The requirements of rapid development and maintenance, and ease of integration can be met if agent-based systems development frameworks adopt, respectively, the *Rapid Application Development* (RAD) strategy and open standards. Since modularity and encapsulation are enforced through ABS, maintenance may be easier. Agents, mostly static ones, can be used to help human users to perform tasks, and they can also be used to personalize interfaces based on the user's preferences [Maes94]. In that case, they are also called *personal software assistants* and they introduce ease of use to computer systems.

The paper "Mobile Agents: Are they a good idea?" [Harr95] emphasizes the argument that MABS constitute a unique approach for distributed systems development. Its authors state that currently there is no other approach with the same functionality and aggregate set of advantages.

Agent-based systems is a research and development field that has gained focus recently. A large and increasing number of universities, research centers and companies are investing on it. Lots of researchers, including Guilfoyle [Guil95] and Janca [Janc96] state that, in the next few years, the agent-based approach will cause a tremendous impact over computer systems, through its partial or complete adoption as a generic systems development paradigm.

This section presented the motivation for mobile agent-based systems as an alternative for distributed systems development. The next section contains a brief overview on agent-based systems with focus on MABS. Section 3 addresses issues concerning frameworks that support mobile agent-based systems. Section 4 presents an assessment concerning existing frameworks that support MABS, based on some issues discussed on section 3. The final considerations are described in section 5.

2. Agent-Based Systems Overview

This section presents a brief overview regarding agent-based systems. Firstly, it addresses the origin and research strands of ABS. Secondly, some considerations on intelligent agents are presented. The final part of this section concerns mobile ABS.

2.1. Origin and Research Strands

According to Nwana [Nwan96], the concept of an agent originated from the Concurrent Actor Model proposed by Hewitt [Hewi77] in the early days of research in Distributed Artificial Intelligence (DAI). That model has provided a basis for one of the broad areas of DAI called Multi-Agent Systems [O'Har96]. Multi-Agent Systems (MAS) are computer systems in which several autonomous entities – agents – interact or work together to perform some set of tasks to satisfy some set of goals. Relevant aspects of MAS include behavior coordination and management. Agent-based systems have evolved from MAS, which can currently be seen as a subset of ABS with focus on DAI issues.

Two research strands in the field of ABS are proposed by Nwana [Nwan96]: the first one which spans from 1977 to the current day, and the second from 1990 to the current day too. The first strand has a strong focus on DAI. Initially, its work concentrated on macro issues such as interaction and communication among agents, the decomposition and distribution of tasks, and coordination and cooperation. One of its main goals was to specify, analyze, design and integrate systems comprising of multiple collaborative agents. Those macro issues emphasize the society of agents over individual agents. In addition to the macro issues, the first strand work has also been characterized by research and development into theoretical, architectural and language issues [Wool95]. The second strand reflects the wide utilization of the term agent in different fields of computer science besides DAI, like distributed systems and software engineering. The investigated types of agents is broader.

Although the terms agent and agent-based systems are widely used, there is no consensus definition for them. On the contrary, there is a diversity of understanding about what an agent is and what are the fundamental characteristics of an agent. The following definitions express this diversity. They were taken from the literature concerning the two research strands proposed by Nwana [Nwan96]:

> *"an agent is a program that helps a user perform some task (or a set of tasks), possibly by maintaining persistent state and communicating with its owner, other agents or its environment in general"* [Ling95]
>
> *"an agent is a software which assists people and acts on their behalf. It is delegated to perform some task(s), and given constraints under which it can operate"* [Janc96]
>
> *"software agents are software components that communicate with their peers by exchanging messages in an expressive agent communication language"* [Gene94]
>
> *"An autonomous agent is a system situated within and a part of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda and so as to effect what it senses in the future"* [Fran96]
>
> *"Intelligent agents are software entities that carry out some set of operations on behalf of a user or another program with some degree of independence or autonomy, and in so doing, employ some knowledge or representation of the user's goals or desires"* [IBM96b]

The term agent appears in the literature lots of times together with qualifiers such as: autonomous, personal and intelligent. Furthermore, some synonyms to agents are used, including software robots, personal software assistants, taskbots and userbots. Those qualifiers and synonyms reflect characteristics considered as fundamental by the researchers that proposed the definitions.

In this paper, agents are considered as *autonomous programs that act on behalf of their owners – users or other programs*. In order to accomplish tasks assigned by their owners, *agents may communicate with other agents, with their environment and with their owners*. The definition adopted focuses on the following fundamental characteristics of *agenthood*:

- *delegation*: users or other programs can delegate tasks to an agent and vest it with authority to act on their behalf;
- *autonomy*: an agent can make its own decisions, based on its owner's statement of goals, preferences and policies;
- *communication*: the ability agents have to interact with their peers, with the environment that hosts them, and with their owner;
- *flexibility*: agents do not assume fixed roles; they may act like clients, servers, observers, etc., depending on their current needs;
- *equity*: agents act like peers, there is no hierarchical relationship among them.

The characteristics pointed out as fundamental reflect the vision that an agent-based system is composed of autonomous entities – agents – that perform tasks assigned – delegated – to them by their owners. Each entity plays flexible roles according to their current needs and is seen as a peer by the other entities that compose the system. Partitioning a system into components enforces modularity and encapsulation. Parallelism can be also explored. The agent-based approach is a generic systems development paradigm that can be potentially applied to any domain.

Although not considered as fundamental, the characteristics described below are strongly related to agents:

- *intelligence*: the ability agents have to reason and learn from the interactions with other agents, users and the environment that hosts them;
- *mobility*: agents can move across heterogeneous computer networks, aiming to progressively accomplish tasks that were assigned to them.

Agents can be classified taking into account several dimensions, such as: their characteristics – fundamental or not, the roles they perform, etc. The most accurate way to provide a classification scheme for agents is through a multi-dimensional matrix. Since there is a large number of possible dimensions, such representation would not be easy to read and understand. Thus, classification schemes have been proposed with limited scope. Some of them encompass the work of Nwana [Nwan96], Franklin [Fran96], Stone [Ston96] and the Foundation for Intelligent Physical Agents [FIPA96].

The next subsection presents some considerations regarding intelligent agents. The last subsection addresses mobile agents, the type of agents on which this paper is focused.

2.2. Intelligent Agents

Intelligence is another term with no consensus definition, even among Artificial Intelligence (AI) researchers. This paper does not attempt to define that term either, however, it presents some considerations found in the literature. Intelligence is strongly related to agents. The concept of agents have originated from the DAI field.

Nwana [Nwan96] states that a key attribute of any intelligent being is its ability to *learn*. The learning may also take the form of *increased performance* over time. Weiss and Sandip [Weis96] enforce that agent-based systems must have the ability to *adapt* and *learn*, that is to self-improve their future performance. The IBM Intelligent Agent Center of Competency [IBM96b] defines intelligence as the *degree of reasoning and learned behaviour*: the agent's ability to accept the user's statement of goals and carry out the tasks delegated to it.

Wooldridge and Jennings [Wool95] present a weak notion of agenthood through the following properties: *autonomy*, *social ability*, *reactivity* and *pro-activeness*. Social ability regards the interaction of agents with other agents (and possibly humans) via some kind of agent-communication language. Reactivity means that agents perceive their environment (which may be the physical world, a user via a graphical user interface, a collection of agents, the Internet, or perhaps all of these combined), and respond in a timely fashion to changes that occur in it. Pro-activeness means that agents do not simply act in response to their environment; they are able to exhibit goal-directed behavior by taking the initiative. A stronger notion of agenthood is defined as follows: *"an agent is a computer system that, in addition to having the properties identified earlier, is either conceptualized or implemented using concepts that are more usually applied to humans"*. This definition also emphasizes the AI understanding of agents, were intelligence is a fundamental characteristic of agenthood.

In this paper, intelligence is considered – even though it is a narrow understanding – as the ability agents have to reason and learn from the interactions with other agents, users and environments that host them.

In order to be intelligent, agents must carry with them some kind of knowledge concerning goals, preferences, vocabularies appropriated to various domains, etc. Knowledge representation languages provide

the means to express knowledge through different approaches such as facts, rules, neural networks, etc. [Ches95].

The research efforts in the field of DAI are mainly related to fixed agents that assist human users in performing tasks. Cheong [Cheo96] presents examples of ABS that can teach people, learn about the habits of human users, and derive knowledge from sensing their external environment.

2.3. Mobile Agent-Based Systems

Regarding to the location where they execute, agents can be *stationary* – also called fixed or static; or *mobile* – also called itinerant [Ches95] or transportable [Gray95][Kota94].

Stationary agents stay on the same host – network node – during all of their existence. They may be used to provide services related to resources available on the hosts where they reside. According to Maes [Maes94], they may be also used in a cooperative process with the user, providing personal assistance for a variety of tasks (those agents are also called *interface agents* [Nwan96]).

Mobile agents are capable of moving over a heterogeneous computer network, aiming to progressively accomplish tasks that were assigned to them by their owners.

The idea of dispatching a program for execution on a remote computer is quite old [Ches95]. Dispatching schemes can be exemplified through batch jobs sent to mainframes in the 1960s, and through executable scripts dispatched among networks of mini-computers to permit distributed, real-time processing in the 1970s. Recently, the concept of active mail is another example in which widely available electronic mail services are enabled to deliver executable scripts. The Tabriz product line from general Magic [Gene96], based on the Telescript language, was the first commercial implementation of a mobile agent-based framework released in 1994.

Similarly to the term agent, there is no consensus definition about what a mobile agent is. Although it is agreed that a mobile agent moves across networks carrying its code and data, there is no agreement regarding bearing its execution state. The following definitions were taken from the literature and they show the lack of consensus concerning that aspect:

> *"mobile agents are programs, typically written in a script language, which may be dispatched from a client computer and transported to a remote server computer for execution"* [Harr95]
>
> *"itinerant agents are programs which are dispatched from a source computer and which roam among a set of networked servers until they are able to accomplish their task; they are moving processes which progressively accomplish tasks by moving from place to place"* [Ches95]
>
> *"transportable agents are capable of suspending their execution, transporting themselves to another host on a network, and resuming execution from the point at which they were suspended"* [Kota94]
>
> *"a transportable agent is a named program that can migrate from machine to machine in a heterogeneous network. The program chooses when and where to migrate. It can suspend its execution at an arbitrary point, transport to another machine and resume execution on the new machine"* [Gray95]

MABS can be applied to several distributed systems domains. Some examples encompass information retrieval, network management, electronic commerce and mobile computing [Ling95]. Information retrieval can be supported more efficiently if an agent representing an user's query can move to hosts where the query is most likely to be answered. This can help users a lot since he or she can delegate the query to an agent which is responsible for finding the information in a huge and increasing amount of sources. Network management, mainly in large networks, can be made easier through agents that monitor operations and detect faults. As electronic commerce in the Internet is becoming a reality, mobile agents can be used to locate the cheapest offerings, negotiate deals or even conclude business transactions on behalf of their owners. Finally, mobile computing can be supported broadly through mobile agents. Usually mobile devices and computers are intermittently linked to a network and they have less powerful computing resources. Since mobile agents support asynchronous computing and disconnected operations they may be of great value in the domain of mobile computing.

3. Mobile Agent-Based Frameworks

Frameworks that support mobile agent-based systems – mobile agent-based frameworks – must provide mobile agents with the infrastructure they need to move over the network and offer means for them to accomplish their tasks. This section discusses some relevant issues regarding mobile agent-based frameworks. These issues include delegation, communication, mobility, host resources management, agents management, execution environments, security, fault tolerance and interoperability. A graphical scheme representing some

main functional blocks of mobile agent-based frameworks, based on the discussed issues, is presented at the end of this section.

3.1. Delegation

Agents were primarily thought as entities performing tasks on behalf of their owners – users or other programs. Although agents can be used to perform any kind of tasks, assisting users with routine and time-consuming activities is a wide field of application for agents.

Maes [Maes94] has pointed out an important question concerning trust related to delegation: "How do you ensure that users feel comfortable delegating tasks to agents?" Most of the agent-based frameworks and ABS are not concerned with this question; they assume users trust in their agents. Maes believes that a machine learning approach can answer this question. She has developed a work relying on this approach, where the user is capable to incrementally build up a model of the agent's competencies and limitations, based on the learning and reasoning capabilities the agent presented to accomplish previous tasks. Thus, according to her, associated intelligence can help users trust in their agents.

Since agents act on behalf of their owners, they must carry with them an identification of who they represent. In the Itinerant Agent Framework proposed by Chess [Ches95], an agent has a passport which contains an authentication of the originator – the name or the authority of the owner, and the name or names of other authority sanctioning entities.

3.2. Communication

Agents must communicate in order to perform tasks. Communication encompasses three aspects: interaction with other agents, hosting environments, and users.

Inter-agent communication can be local or remote, synchronous or asynchronous. Agents communicate with their peers, for instance, when an agent representing a user request interacts with a service provider – probably static – agent. The communication can have cooperation purposes, i.e., agents can communicate in order to accomplish common goals; for example, agents representing some people cooperate in order to schedule a meeting.

Genesareth and Ketchpel [Gene94] focus their research on interoperability issues involving agent communication. They introduce the concept of *agent-based software engineering*, where applications are written as *software agents*, i.e., software components that communicate with their peers by exchanging messages in an expressive agent communication language (ACL). According to them, in this approach, agents use a common language for communication, with an agent-independent semantics, facilitating the creation of interoperable software. The following question then emerges: "What is an appropriate agent communication language?" Researchers in the ARPA Knowledge Sharing Effort have defined three components for an ACL: its vocabulary; an inner language called *Knowledge Interchange Format* (KIF) encoding simple data, expressions, rules, constraints [Gene92]; and an outer language called *Knowledge Query and Manipulation Language* (KQML) which is a linguistic layer above KIF that provides context information for efficient communication [Fini93].

Agents communicate with their hosting environment in order to make use of the services and resources it provides. Access to corporate databases and to Object Request Brokers (ORB) based on the CORBA standard [OMG96] are important examples of this kind of interaction.

Agents may need to communicate with their owners and vice-versa. Mechanisms such as electronic mail, for asynchronous interactions, and graphical user interfaces, for example to ask for user confirmation through a pop up window, can be used.

3.3. Mobility

Mobile agents can move across heterogeneous networks, migrating from one host to another, in order to progressively accomplish tasks that were assigned to them. Migration encompasses destination selection, dispatching, transport and receiving of agents.

An agent migrates in order to accomplish tasks. Thus, selecting the host that provides the needed service is a primary issue. Although agents can carry a fixed itinerary with them, in most cases they have to decide where to go. Mechanisms such as keyword search and semantic routing – which tries to find an agent's destination based on what it wants to accomplish – can be used to assist agents in selecting a service provider. These mechanisms may make use of yellow pages or directory services.

Dispatching an agent comprises the following steps: having an agent suspended and encoded for transmission, and releasing the resources used by the agent. Receiving an agent encompasses the following steps: having an

agent decoded and checking if the needed resources for the agent execution are available. If so, than the agent is delivered to its execution environment.

Transport mechanisms transmit an agent from one host to another. Standard protocols such as TCP/IP, HTTP, X.25, and e-mail can be used for transmission purposes.

Intelligence is gaining focus in the mobile agents area [Ches95] [IBM96b]. Mobile agents are usually fine-grained, so they can easily migrate over the network. A key challenge is to meet mobility and intelligence requirements. A mobile agent must carry the minimal, sufficient and necessary information to accomplish the task [Ches95]. If this information is a large knowledge base associated to an execution environment that requires powerful processor resources, the mobility requirements possibly will not be met.

3.4. Host Resources Management

Mobile agent-based frameworks must provide mechanisms for controlling resources available at hosts. This kind of control would allow an efficient utilization of resources, and would avoid them to be inadequately deployed by agents. Agents could contain information regarding their resource needs and their current resource utilization capacity – for example, when they need to pay for using the resources. Such information must be checked by the mobile agent-based framework for resources allocation, before allowing the execution of agents. On the other hand, after the dispatching of an agent the allocated resources must be released.

3.5. Agents Management

Mobile agent-based frameworks must control the life cycle of an agent residing on a host. Each host must have control over the instantiation and the removal of agents. Agents are instantiated before they can be executed on a host. Agents are removed after their dispatching.

3.6. Execution Environments

Execution environments permit the execution of agents written in agent-based programming languages (ABPLs). Compilers and interpreters compose the core of execution environments.

Frameworks that support agent-based systems must not enforce the use of a particular agent-based programming language. They must support various ABPLs and provide interoperability among agents written in different ABPLs. A standard ACL [Gene94] can be used with this purpose. Agent-based programming languages can be examined regarding their *structure* and *execution model*.

ABPLs can be, concerning their structure: *imperative*, *functional* or *declarative*. Imperative ABPLs can be divided into two main categories: *procedural* and *object-oriented*. They are widely used by mobile agent-based frameworks. Functional and declarative languages are mainly used by agent-based frameworks that have a strong focus on intelligence for agents.

Concerning the execution model, ABPLs can be *compiled*, *interpreted* or *compiled to an intermediate portable interpreter-based language*. The first execution model applies mainly for stationary agents, since compiled binary code is platform specific. The other models are widely used by mobile agent-based frameworks, whose code is unique and deployed on different platforms. Platform independence is achieved by the last two execution models, although they introduce performance penalties.

3.7. Security

Security is a relevant issue regarding agent-based frameworks that support mobile agents. In mobile agent-based systems, an agent may freely migrate to any host on the network. Thus, security mechanisms must exist in order to prevent malicious actions from incoming agents. Also, agents must not be tampered with by their hosts [Ches95]. Although security is a relevant issue, some goals related to it cannot be achieved by using the currently available computing technologies. For instance, the verification by a host, with complete certainty, that an arbitrary agent is not a computer virus. However, some security functions can be addressed and provided such as: *authentication* of an agent's sender, checking sender's *authorization* for execution purposes, resources *access control*, actions *logging*, and *integrity checking*.

Authentication means the unambiguous establishment of the sender's identity. This can be accomplished by including a public key certificate – through encryption techniques – of the sender as part of the agent [Ches95]. Access control regards the management of resources residing at a host, avoiding them to be inappropriately used by agents. Agents and information they carry must be protected from tampering through integrity checking mechanisms. Encryption techniques can also be used for such purposes.

3.8. Fault Tolerance

Mobile agent-based frameworks must provide mechanisms for the recovering of agents and the information they carry in case of computer or network failures, so mobile agent-based systems can resume their normal execution as soon as possible. Persistence is a strategy to meet fault tolerance requirements, where agents are stored in non-volatile memory.

3.9. Interoperability

Interoperability regards the ability an MABS has to interact with other systems. In order to interoperate, an MABS must rely on open, non-proprietary standards. Examples of such standards are the Common Object Request Broker Architecture (CORBA) [OMG96], and the TCP/IP family of protocols.

Agent-based systems is an emerging and rapidly growing research and development field , thus the standardization efforts have begun to appear. Some of them include the Foundation for Intelligent Physical Agents[FIPA96], the Agent Society[Agen96], and the OMG's Mobile Agent Facility [OMG96].

The FIPA is an international non-profit association of companies and organizations. It aims to promote the development and specification of agent technologies that are usable across a large number of applications providing a high level of interoperability across applications. Standardization categories currently addressed include: communication, information representation, agent societies, execution environments for mobile agents, and agent management.

The Agent Society is a international industry and professional organization established to assist in the development and emergence of mobile intelligent agent technologies and markets. It supports initiatives for the development and implementation of open intercommunication protocols and interfaces for open intercommunication and interoperability among diverse agents. One of the activities currently in progress encompasses the proposal of a standard agent transfer protocol.

The OMG's Mobile Agent Facility supports the mobility of agents (CORBA objects) and the invocation of their execution environments in the scope of the OMG's Object Management Architecture (OMA).

3.10. Functional Decomposition Scheme for Mobile Agent-Based Frameworks

Figure 3 depicts a graphical scheme representing some main functional blocks of mobile agent-based frameworks, based on the issues addressed in this section. Steps 1 through 5 are taken when agents arrive at a host. Steps 6 through 10 are taken when agents migrate to another host on a network.

Chess [Ches95] presents a model for a mobile agent-based framework where agents can accumulate knowledge and make decisions.

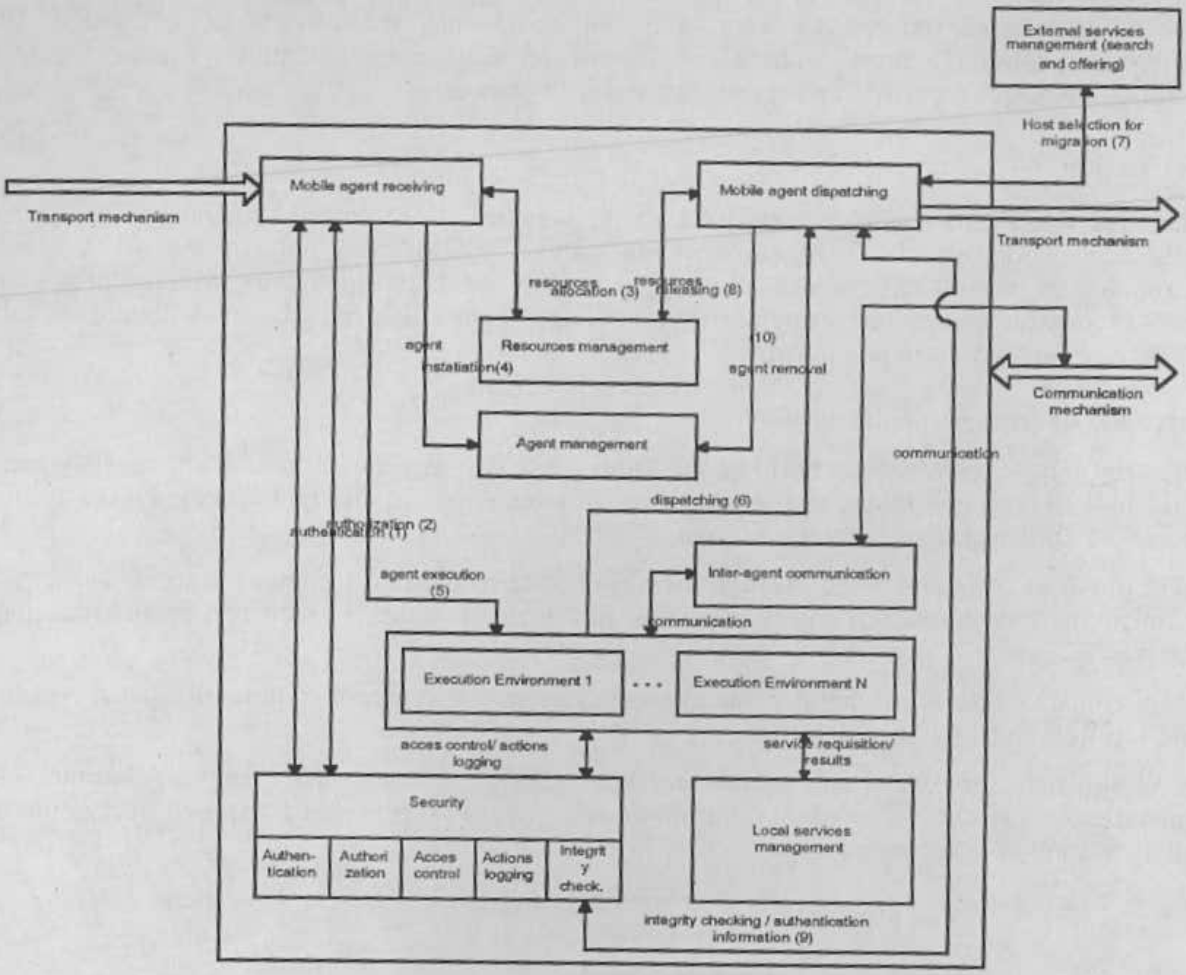

Figure 3: Functional decomposition of mobile agent-based frameworks

4. Existing Mobile Agent-Based Frameworks Assessment

This section presents an assessment on how a set of existing development frameworks that support MABS addresses some of the issues discussed in the previous section. The set of examined frameworks is composed of:

- *Tabriz* 1.0 from General Magic, based on the *Telescript* [Gene96] programming language. Tabriz is available for the following Unix platforms: HP UX, Solaris and SGI Irix. It requires powerful hardware, operating system, and processor resources;
- *Agent Tcl* alpha release 1.1 and *Agent Tk* [Gray95] from Dartmouth College. These are extended versions of Tcl and Tk supporting mobile agents, which are available for the following Unix platforms: Linux, Free BSD, IBM AIX, SGI Irix, DEC OSF/1, SunOs and Solaris;
- *Aglets Workbench* [IBM96a] alpha 4 from IBM's Tokyo Research Laboratory. It is a visual environment for building mobile agent-based applications in Java. Aglets Workbench can be executed on Windows NT/95 and Solaris platforms;
- *TACOMA* 1.2 [Joha95] from University of Tromsø and Cornell University, focuses on operating systems issues in connection to the agent paradigm for distributed computing. TACOMA is based on a language orthogonal model which currently supports C, Tcl, Perl, Phyton and Scheme. Version 1.2 is available for the following Unix platforms: HP UX, Solaris, BSD Unix and Linux;

Other two frameworks were considered, but not selected for the purposes of this paper: *Mole* [Mole96] alpha 1.0 from the University of Stuttgart, and *ARA* (Agents for Remote Actions) [ARA96] from the University of Kaiserslautern. Both had their development effort started recently. The Mole system is based on Java; mechanisms for migration and communication of agents were added to that language. It is available for the Solaris and Windows NT/95 platforms. As stated by its developers, the Mole system is still incomplete and buggy. ARA has its application focus on services for mobile computer systems with a wireless network connection. It supports agents written in C++ and in an extended version of Tcl. ARA is available for the SunOs 4.1 and Linux 1.2 platforms. Although some important results have been achieved by its development team, functional evolution and comprehensive tests still need to be performed on the ARA system.

The mobile agent-based frameworks were examined concerning the aspects of delegation, inter-agent communication, communication with hosting environments, communication with owners, mobility, programming language, security, interoperability and persistence.

4.1. Delegation

The examined frameworks assume users trust in their agents. Regarding the identification of an agent's owner, the frameworks Agent Tcl, Aglets Workbench and TACOMA rely only on the *user's id* related to a process running on multi-user systems. Telescript associates agents to *authorities*, which represent real world individuals or organizations. Telescript agents can discern authorities but neither withhold or falsify their authorities; also anonymity is precluded.

4.2. Inter-agent communication

The Telescript language, which is the basis for Tabriz, has two agent communication mechanisms: *meeting* for interactions in the same *place*, and *connections* for interactions in different *places*. Places are units inside a computer network where services are offered.

Agent Tcl provides communication through *messages* for asynchronous communication, *events* which have similar functionality to messages, and *meetings* for direct connections between two agents enabling efficient data transfer.

TACOMA contains a *meet* mechanism for communication and synchronization of agents. Agents meet at the same location and exchange data.

Aglets Workbench provides an agent *message-passing scheme* that support remote and local communications. Loosely coupled asynchronous as well as synchronous peer-to-peer communication between agents are also supported.

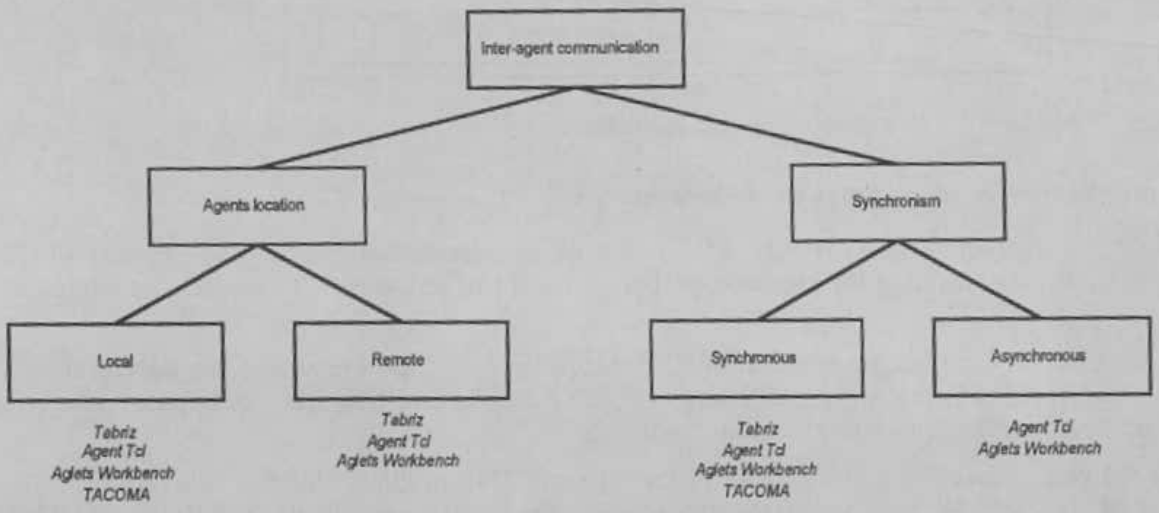

Figure 4: Inter-agent communication

4.3. Communication with hosting environments

Most of the examined frameworks that support MABS – Tabriz, Agent Tcl and Aglets Workbench – were built based on, or have escape mechanisms to, languages like C/C++, Tcl and Java, which provide general purpose means to access services and resources available on hosts. TACOMA is based on a language orthogonal model that supports agents written in various general purpose programming languages.

According to General Magic, Tabriz will provide agents, in a near future, with access to corporate databases through ODBC [Gryp95]. The Aglets Workbench has a data access component called *JoDax*, which provides a high level data access class library in Java.

Regarding the interaction with Object Request Brokers, programs written in Telescript allow C calls, thus a ORB interface may be implemented. Tcl-Dii [CERC95] is a Tcl interface for accessing Iona's ORB implementation called Orbix [Iona96], through the Dynamic Invocation Interface (DII). Tcl-Dii can be attached to Agent Tcl programs. Finally, some ORB implementations provide integration with Java, which is the basis for Aglets Workbench. OrbixWeb from Iona provides Java clients with capabilities to interoperate with back-end Orbix CORBA based services. Visigenic's [Visi96] VisiBroker for Java, allows developers to

deploy application objects in a Java program as client and server-side Java code. Java applets can send and receive messages from distributed objects on the servers via VisiBroker for Java.

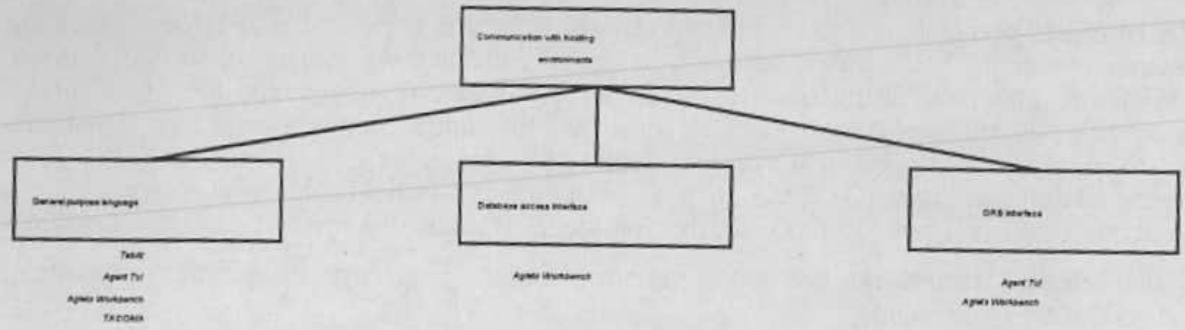

Figure 5: Communication with hosting environments

4.4. Communication with owners

Tabriz allows adding C code for interaction with users into the body of agent-based applications. Thus, graphical user interfaces can be added to Telescript programs. Tabriz also supports electronic mail and paging.

Agent Tcl has a counterpart called Agent Tk, which enables programmers to add graphical user interface capabilities to agent programs.

The Aglets Workbench is based on Java. Thus, the capabilities of the Java's Abstract Windowing Toolkit (AWT) can be used for graphical user interfaces.

TACOMA supports agents written in Tcl/Tk and C, languages that provide programming capabilities for graphical user interfaces.

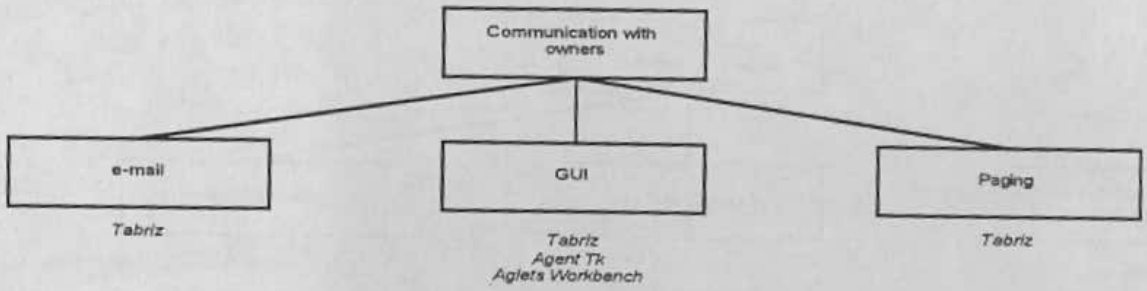

Figure 6: Communication with owners

4.5. Mobility

The Tabriz architecture is composed of the following components: Telescript Language, Telescript Engine and Telescript Protocol Suite. The Telescript language implements concepts such as *agents*, *places* and *travel*, supporting agent mobility. A network of computers is modeled as a collection of places, where a place offers a service. An agent occupies a particular place and travels from one place to another through the *go <ticket>* instruction. The ticket constitutes data that specifies the agent's destination. The Telescript engine is a program that implements the Telescript language, by maintaining and executing places within its purview, as well as the agents that occupy those places. Finally, the Telescript protocol suite enables engines to communicate in order to transport agents between them. The protocol suite can operate over TCP/IP, X.25 or even electronic mail. Tabriz supports migration of the program execution state.

The Agent Tcl architecture has two components: *server* and *interpreter*. The server runs at each network site, accepting incoming agents, messages and meeting requests; and keeping track of the agents running on its hosting machine. The interpreter is a modified Tcl core for agents execution. The *agent_jump <machine>* command migrates an agent to a particular machine transparently. This command captures the internal state – including the program counter – of the agent and sends the state image to the server on the destination machine. Agent Tcl uses the TCP/IP protocol.

Aglets Workbench has, among others, the following components: Aglets Framework, Agent Transfer Protocol (ATP) Framework, and Aglet Server. The Aglets Framework introduces the notion of an *aglet*, a mobile – agile – agent written in Java. The Java Aglet API is a standard for interfacing aglets and their environments, containing methods for aglet creation, disposing, dispatching – through the *dispatch <URL or itinerary >* command, etc. The ATP Framework is the standard protocol used to transfer mobile agents,

based on Uniform Resource Locators (URLs). The Aglet Server receives incoming aglets over the network for execution. Aglets Workbench do not support migration of the program execution state.

TACOMA considers agents as mobile computational units. State is organized into folders which are units of data accessible by agents, and each folder has a name. A collection of folders associated with an agent is called a briefcase. The only abstraction supported by TACOMA to implement the agent model is: *meet <another agent> <briefcase>*. Agents meet at some location and exchange a briefcase. Agents move when they want to meet another agent at a different host – the host name is stored in the HOST folder of the moving agent's briefcase. Since TACOMA is based on a language orthogonal model, it does not support, by default, automatic state capture. TACOMA relies on the TCP protocol suite as its transport mechanism.

None of the selected frameworks has a destination selection mechanism. Itineraries or destinations are defined as part of an agent's code.

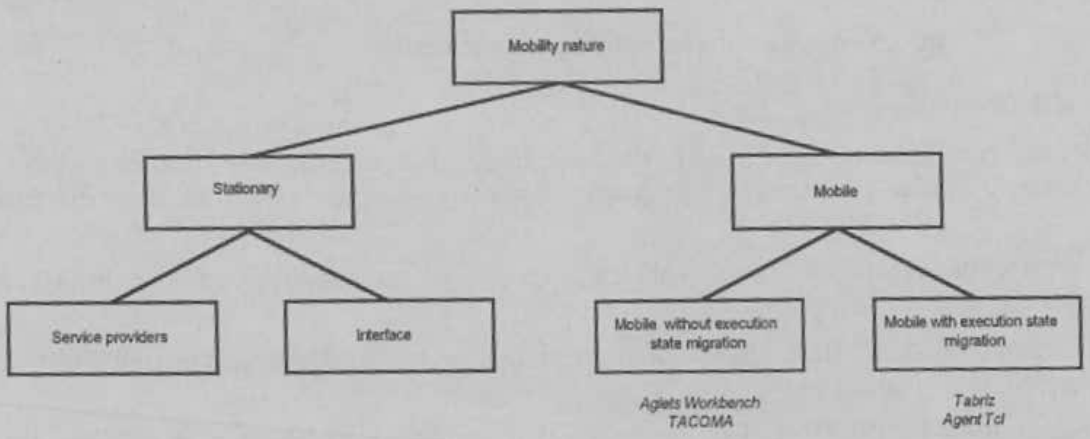

Figure 7: Mobility nature

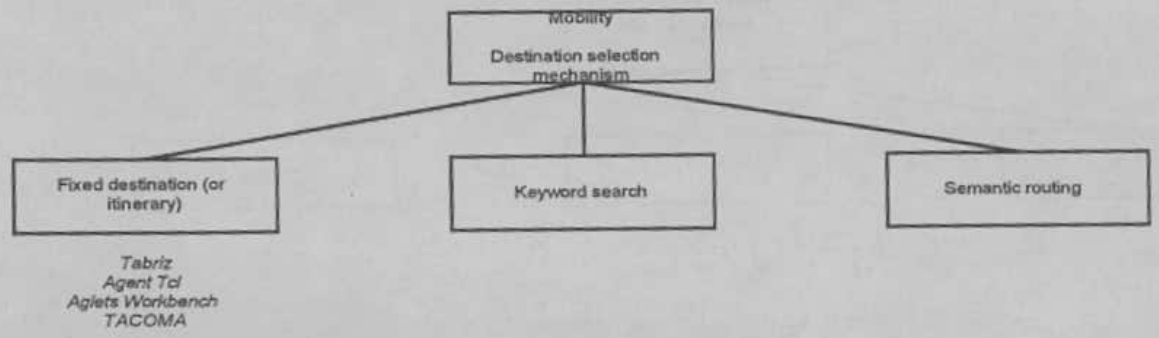

Figure 8: Destination selection mechanisms

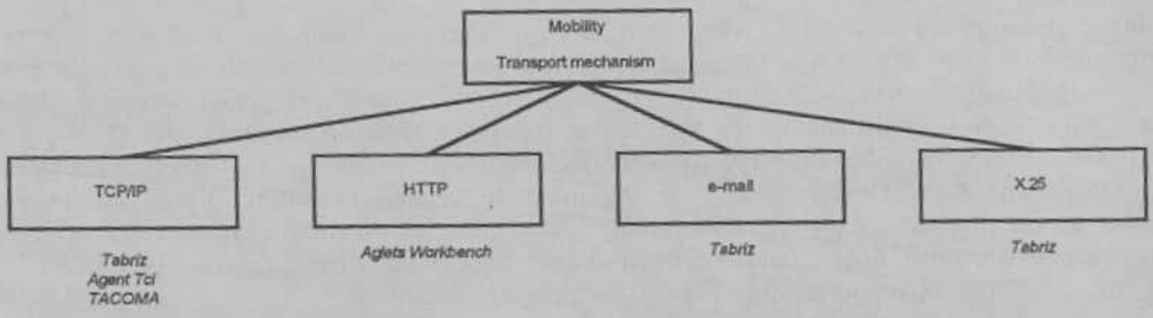

Figure 9: Transport mechanisms

4.6. Programming Languages

Telescript and Java – which is the basis for Aglets Workbench – are object-oriented languages, where programs are compiled to an intermediate portable interpreter-based language. Agent Tcl is a procedural and interpreted language. TACOMA is based on a language orthogonal model supporting languages with different structures and execution models.

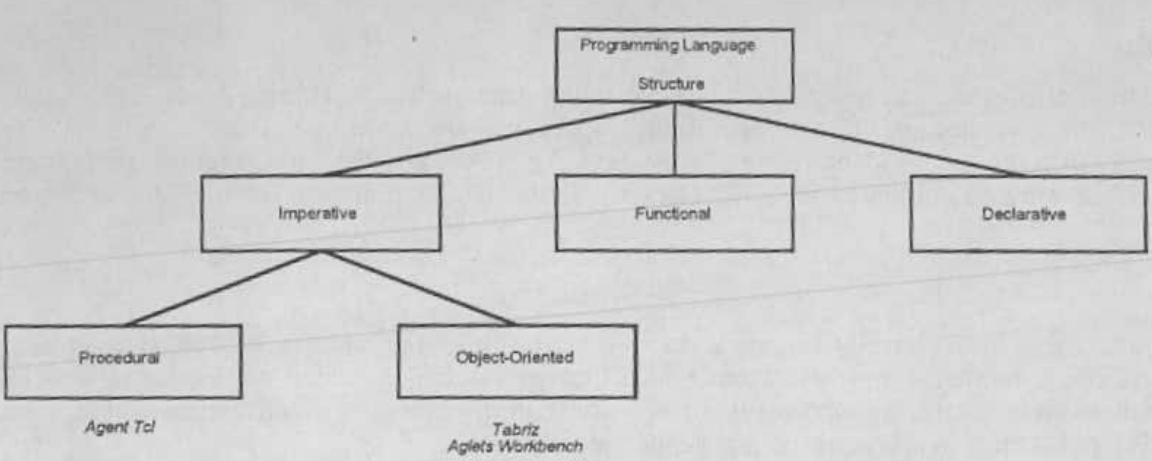

Figure 10: Programming language structure

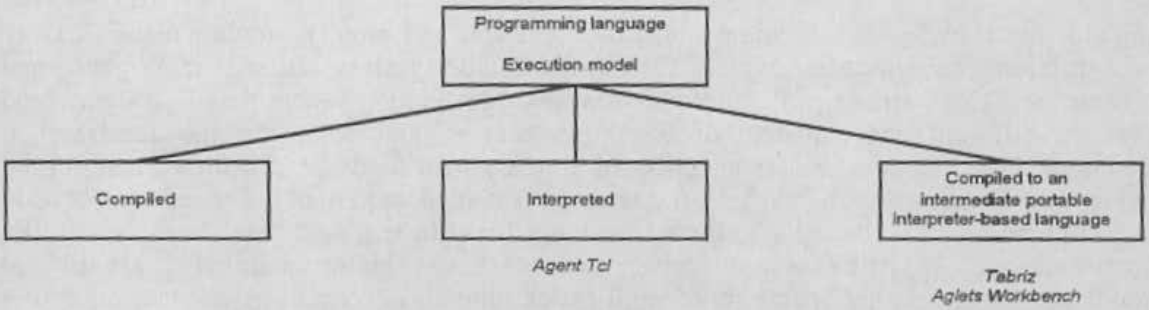

Figure 11: Execution models

4.7. Security

Telescript verifies the authority of an agent when needed. It also lets *authorities* – check for details in the delegation issue – limit what agents and places can do through assigning *permits* to them. A permit is data that grants capabilities of two kinds: the right to execute a certain instruction; and the right to use a certain resource in a certain amount. Also, a Telescript program is executed through a Telescript Engine, where an agent cannot access directly the computer processor, memory, file system or peripheral devices.

Agent Tcl provides rudimentary security. Any agent, message or connection request that does not come from an approved machine is ignored. The system administrator specifies the set of approved machines.

Aglets Workbench supports an extensible layered security model. The first layer comes from the Java language – for example, consistency checks performed by the Java bytecode identifier. The next layer has a security manager which allows developers to implement their own protection mechanisms. The final layer is the Java security API, which provides capabilities to include security functionality in agent programs, such as: authentication, digital signatures and cryptography. The Java Security API is not yet available for the Aglets Workbench Alpha 4 release.

The TACOMA system provides security mechanisms through a firewall agent called *tac_firewall*. The execution of a *meet* instruction results first in the interaction of a mobile agent with the firewall agent, that currently only logs the incoming briefcase to disk. According to the development team, new prototype versions of TACOMA will contain a more comprehensive set of security mechanisms including authentication and access control.

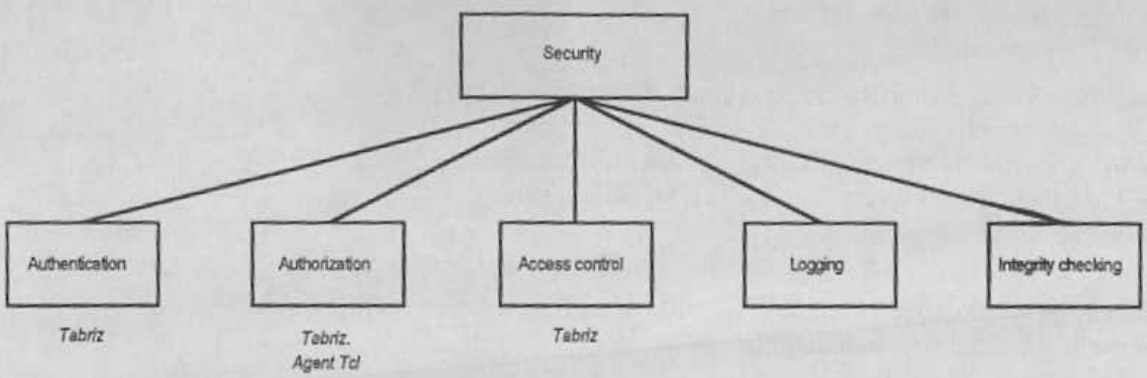

Figure 12: Security

4.8. Interoperability

Since the standardization efforts are emerging, all of the examined frameworks – Tabriz, Agent Tcl, Aglets Workbench and TACOMA – cannot provide interoperability capabilities among MABS. Regarding the Aglets Workbench, recently a language independent version of the Java Aglet API and the Java Agent Transfer and Communication Interface were submitted to the OMG as part of the IBM's proposal for the Mobile Agent Facility.

4.9. Persistence

Tabriz supports persistence at the Telescript Engine level, automatically saving an agent and its state in non-volatile memory. TACOMA maintains a mobile agent's state through briefcases, which are logged to disk by the tac_firewall agent as soon as the mobile agent arrives at its destination, as a consequence of a *meet* instruction. Agent Tcl and Aglets Workbench do not support persistence.

5. Conclusions

The agent-based approach for software development constitute a recent and rapidly growing research field. Although researches of different computer science areas such as distributed systems, artificial intelligence and software engineering are working with agent related issues and developing agent-based systems and frameworks, they have a different understanding of what an agent is, and what are the fundamental characteristics of agenthood. This paper addresses agent-based systems with focus on distributed computing, where mobility is a very significant issue. The existing frameworks that support MABS still need to evolve in order to meet the requirements for distributed systems pointed out here. In the next few years, hopefully, frameworks will provide developers with comprehensive means for developing distributed agent-based systems which can be deployed in a broad spectrum of application domains. As a consequence, computer users will have their life made easier, since many of their tasks would be automated through agents – also called personal software assistants.

Acknowledgments

This work has been supported by *Centro Nacional de Pesquisa Tecnológica em Informática para a Agricultura* (CNPTIA), which is a research center of *Empresa Brasileira de Pesquisa Agropecuária* (EMBRAPA).

References

[Agen96] The Agent Society
The Agent Society Home Page
http://www.agent.org

[ANSA95] Advanced Networked Systems Architecture (ANSA)
Scripts and Mobile Agents
ftp://ftp.ansa.co.uk/phase3-doc-root/bn/APM.1593.01.ps.gz
1995

[ARA96] Distributed Systems Group (DSG) – Computer Science Department
University of Kaiserslautern
The Ara Project
http://www.uni-kl.de/AG-Nehmer/Ara/ara.html
1996

[CERC95] Concurrent Engineering Research Center (CERC)
West Virginia University
TclDii: A Tcl Interface to the Orbix Dynamic Invocation Interface
http://webstar.cerc.wvu.edu/dice/iss/TclDii/TclDii.html
December, 1995

[Cheo96] Cheong, F.;
Internet Agents: Spiders, Wanderers, Brokers and Bots
New Riders Publishing, Indianapolis, 1996

[Ches95] Chess, D.; Grosof, B.; Harrison, C.; Levine, D.; Paris, C.; Tsudik, G.
Itinerant Agents for Mobile Computing – IBM Research Report RC 20010

IBM Research Division, 1995

[Fini93] Finin, T. and others
Specification of the KQML Agent-Communication Language plus example agent policies and architectures
External Interfaces Working Group of the DARPA Knowledge Sharing Effort
Technical Report, Working Paper, June 1993

[FIPA96] The Foundation for Intelligent Physical Agents (FIPA)
FIPA Home Page
http://www.cselt.stet.it/fipa/index.htm
November, 1996

[Fran96] Franklin, S.; Graesser, A.
Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents
Proceedings of the 3rd International Workshop on Agent Theories, Architectures, and Languages
Springer-Verlag, 1996

[Gene92] Genesareth, M. R.; Fikes, R. E.
Knowledge Interchange Format Version 3.0 Reference Manual
Technical Report Logic-92-1, Stanford University, January 1992

[Gene94] Genesareth, M. R.; Ketchpel, S. P.
Software Agents
Communications of the ACM, 37, July, 1994

[Gene96] General Magic, Inc.
Telescript Technology: Mobile Agents
http://www.genmagic.com/Telescript/Whitepapers/wp4/whitepaper-4.html
1996

[Gray95] Gray, R. S.
Agent Tcl: A transportable agent system
Proceedings of the CIKM Workshop on Intelligent Information Agents
CIKM 95, Baltimore, Maryland, 1995

[Gryp95] Gryphon, R.; Charpentier L.; Oelschlaeger, J.; Shoemaker, A.; Cross, J.
Using ODBC 2
Special Edition, QUE, 1995

[Guil95] Guilfoyle, C.
Vendors of Agent Technology
UNICOM Seminar on Intelligent Agents and their Business Applications
London, November, 1995

[Harr95] Harrison, C. G.; Chess, D. M.; Kershenbaum, A.
Mobile Agents: Are they a good idea?
IBM Research Report, IBM Research Division, 1995

[Hewi77] Hewitt, C.
Viewing Control Structures as Patterns of Passing Messages
Artificial Intelligence 8(3), 323-364, 1977

[IBM96a] IBM Corporation – Tokyo Research Laboratory
Programming Mobile Agents in Java – A White Paper
http://www.trl.ibm.co.jp/aglets/whitepaper.htm
September, 1996

[IBM96b] IBM Corporation – IBM Intelligent Agent Center of Competency
The Role of Intelligent Agents in The Information Infrastructure
http://www.raleigh.ibm.com/iag/iagptc2.html

[Iona96] Iona Technologies Ltd.
Iona's Orbix WWW Page
http://www.iona.com/Orbix/index.html

[Janc96] Janca, P. C.
Practical Design of Intelligent Agent Systems
IBM Corporation, Research Triangle Park, June, 1996

[Joha95] Johansen, D., Renesse, R., Schneider, F.B.
An Introduction to the TACOMA Distributed System – Version 1.0
Computer Science Technical Report: 95-23
Department of Computer Science, University of Tromsø, June, 1995

[Kota94] Kotay, K.; Kotz, D.
Transportable Agents
Department of Computer Science – Dartmouth College – 1994

[Ling95] Lingnau, Anselm; Drobnik, Oswald
An Infrastructure for Mobile Agents: Requirements and Architecture
Fachbereich Informatik (Telematik), Johann Wolfgang Goethe-Universität
1995

[Maes94] Maes, P.
Agents that reduce work and information overload
Communications of the ACM, 37, July, 1994

[Mole96] Distributed Systems Group (DSG) – Institute of Parallel and High-Performance Systems (IPVR) – University of Stuttgart
Mole Alpha 1.0 Documentation
ftp://suntrec.informatik.uni-stuttgart.de/pub/MOLE/documentation.html
November, 1996

[Nwan96] Nwana, H. S.
Software Agents: An Overview
Knowledge Engineering Review, Vol. 11, No 3, pp. 1-40, September, 1996

[O'Hare96] O'Hare, G. M. P.; Jennings, N. R.
Foundations of Distributed Artificial Intelligence
John Wiley & Sons, 1996

[OMG96] Object Management Group (OMG)
What is CORBA?
http://www.omg.org/corba.htm

[Ston96] Stone, P.; Veloso, M.
Multiagent Systems: A Survey from a Machine Learning Perspective
Submitted to IEEE Transactions on Knowledge and Data Engineering
June, 1996

[Visi96] Visigenic Software, Inc.
Visigenic Software - Product Overview
http://www.visigenic.com/info/prod.html

[Weis96] Weiss, G., Sandip, S.
Adaptation and Learning in Multi-Agent Systems
Springer-Verlag, 1996

[Wool95] Wooldridge, M., Jennings, N.
Intelligent Agents: Theory and Practice
Knowledge Engineering Review, Volume 10, No 2, June 1995.