To Appear in IEEE Computer Magazine, 1999. Copyright IEEE CS Press

Coordination Models for Internet Applications based on Mobile Agents

Giacomo Cabri, Letizia Leonardi, Franco Zambonelli (*)
Dipartimento di Scienze dell'Ingegneria
Università di Modena
Via Campi 213/b – 41100 Modena – ITALY
Phone: +39-059-376735
Fax: +39-059-376799
E-mail: {giacomo.cabri, letizia.leonardi, franco.zambonelli}@unimo.it

Abstract

Internet applications can take advantage of a design paradigm based on mobile agents. However, mobile agents introduce peculiar problems in the coordination of the application components. This paper surveys a set of possible coordination models for mobile agent applications and adopts a sample application in the area of WWW information retrieval to clarify the concepts expressed and to outline the advantages and the drawbacks of each model. This analysis shows that reactive tuple space models are the most suited to mobile agent applications, due to their flexibility and adaptability.

***Keywords**: Mobile Agents, Coordination, Reactive Tuple Spaces, WWW Information Retrieval*

1 Introduction

The shift of several application areas from intra-net environments to the Internet, along with the rise of several new Internet-oriented applications, requires new design and programming paradigms. Traditional distributed applications are designed as a set of processes assigned to a given execution environment (acting as managers of local resources) that cooperate in a network-unaware fashion. The *mobile agent* paradigm, instead, defines applications made up of network-aware entities (*agents*) capable of actively changing their *execution environment* by transferring themselves while executing (*mobility*) [KarT98, Fug98].

The current interest in the mobile agent paradigm is widely justified by the advantages it provides over traditional approaches in Internet applications: *(i)* mobile agents can significantly *save bandwidth*, by moving locally to the resources they need; *(ii)* mobile agents can carry the *code to manage remote resources* and they do not need the remote availability of a specific server; *(iii)* mobile agent applications do not require *continuous network connections* to proceed, because interacting entities can be moved to the same site when connections are available and interact without requiring further network connections; consequently *(iv)* mobile agents suit *mobile computing* systems.

In the last few years, several systems and programming environments have appeared or have been proposed to support the development of Internet applications based on mobile agents [KarT98]. Nevertheless, several issues still have to be solved to make the mobile agent technology widely accepted: secure and efficient execution supports, standardization, appropriate programming languages and coordination models [Fug98].

In particular, one of the fundamental activities in mobile agent applications is coordination between agents and the entities they encounter during execution, these entities being *other agents* or *resources* in the hosting environments. However, mobility of application components and extension of the distribution area imply problems that are different from the ones concerning traditional distributed systems. This paper proposes a taxonomy of the possible coordination models for mobile agent applications and uses it as a basis to survey and analyze some recent proposals.

An application in the area of WWW information retrieval is assumed as a case study to illustrate the approaches presented and to outline their possible advantages and drawbacks. In particular, the application helps to point out that coordination models that fully uncouple the interacting entities (like Linda [GelC92, PapA98]) – possibly enhanced with the capability of programming specific reactions in response to interaction events [CorLZ98] – have the necessary adaptability to suit the wide and heterogeneous Internet scenario, implying a simple application design.

2 The Case Study Application

The application we adopt to illustrate the impact of different coordination models in the design of mobile agent applications relates to WWW information retrieval.

Let information be stored in HTML pages spread over the Internet and let the user be interested in all HTML pages that include one specific *keyword*. The user knows one site, to be used as a search starting point, where to find interesting (i.e., matching the keyword) documents. Thence, other sites containing further interesting information can be found by following the links contained in the HTML pages.

A *traditional* design could be based on the definition of a user process that asks the starting WWW server for a page, retrieves and analyses it and repeats the same task for each WWW server referred by the links found in the interesting documents. The intrinsic inefficiency of this approach mainly stems from the need to transfer large amounts of data over the network, with a great waste of bandwidth and time. In addition, the lack of reliability of network connections is likely to make a large number of accesses to some sites fail, thus requiring several attempts before completing the analysis of the documents on those sites.

The *mobile agent* design could be based on *searcher agents* that move locally to a site, analyze the stored documents, and follow the interesting links by cloning themselves. In particular, the user creates a first searcher agent and sends it to the starting site, with the goal of analyzing the HTML pages there allocated and yielding back the URLs of the interesting pages. If the first agent finds interesting documents containing links to other sites, the agent clones itself (i.e., it creates a new agent with the same code) for any link to follow (see figure 1) and eventually returns to the home site with the information found. The higher efficiency and reliability of this approach if compared to the traditional (*non-mobile*) one is evident: *(i)* it saves bandwidth by moving only aggregated data over the Internet and avoiding the transfers of large amounts of data; *(ii)* it does not require continuous network connections, because the analysis of the documents on a site can proceed even in the absence of network connections.

The mobile agent design has to be completed by identifying the coordination needs of the application components and by solving them with the exploitation of an appropriate coordination model.

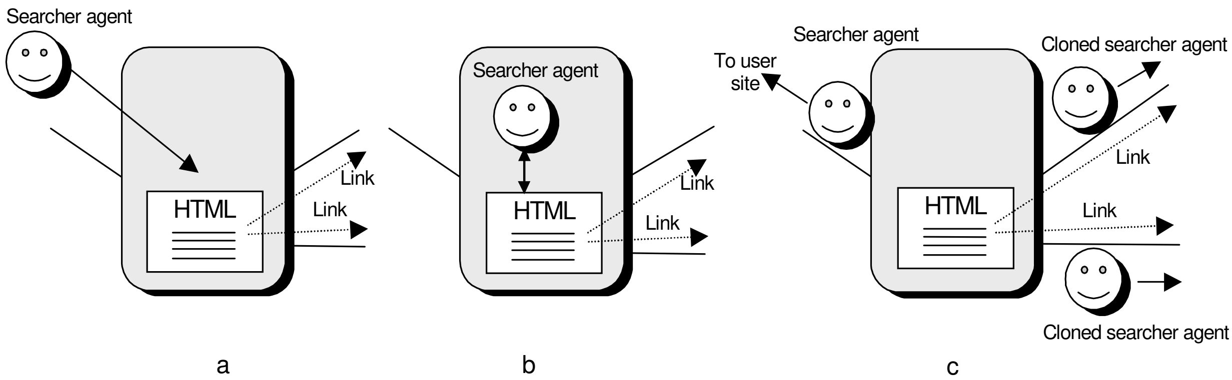

Figure 1. An agent arrives at a site (a), accesses the local HTML pages (b) and creates a new agent for any link to be followed before returning to the user site (c).

3 A Taxonomy of Coordination Models for Mobile Agent Applications

In its nomadic life, an agent needs to coordinate its activities with other entities, both other mobile agents and resources in the hosting environments. In particular:

- an application may be made up of several mobile agents that cooperatively perform a task and, then, are in need of exchanging data and knowledge;
- a mobile agent usually needs to roam through the Internet to access remote resources and services allocated on some nodes of the network.

In the case study application, inter-agent coordination is needed to avoid *multiple visits* in the same site. In fact, cross-references in HTML pages may cause searcher agents to visit the same sites several times, thus duplicating searches and wasting time and resources. Agent-to- execution environment coordination is aimed at defining a precise protocol to *retrieve information* on a site. Letting an agent directly access the local HTML pages is not viable, because agents may have no knowledge of the local file system structure and because this solution lacks security.

Coordination issues have been extensively studied in the past [GelC92], leading to the definition of different taxonomies for coordination models [PapA98]. To analyze the impact of different coordination models in mobile agent applications, we propose a new taxonomy built on two characteristics that are of particular relevance when mobile agents are involved, i.e., the degrees of *spatial* coupling and *temporal* coupling induced by a coordination model. In particular:

- spatially coupled coordination models require the interacting entities to share a common name space; conversely, spatially uncoupled models enforce anonymous interactions;
- temporally coupled coordination models imply synchronization of the entities involved; conversely, temporally uncoupled coordination models achieve asynchronous interactions.

Four main categories of coordination models can derive from the combination of the abovementioned characteristics, as shown in figure 2: direct, meeting-oriented, blackboard-based and Linda-like.

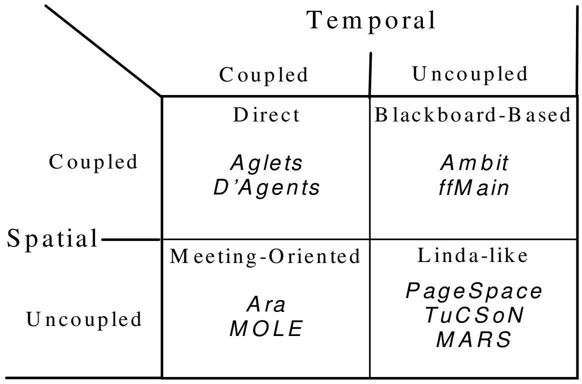

Figure 2. Coordination models for mobile agent applications-

3.1 Direct Coordination

In *direct* coordination models, agents start a communication by explicitly naming the partners involved (spatial coupling). In the case of inter-agent coordination, two agents are to agree on a communication protocol, typically *peer-to-peer*. The access to the local resources generally uses *client-server* coordination, since a hosting environment usually provides local servers for the management of its resources. Direct coordination usually implies synchronization of the entities involved (temporal coupling).

Most of the Java-based agent-systems – like *Aglets* and *D'Agents* [KarT98] – adopt the direct coordination model. At a high level, they can exploit the client-server coordination model typical of the object-oriented paradigm, even between remote objects via Java *RMI*. At a low level, they can directly exploit TCP/IP, which is very flexible but requires the precise definition of a message exchange protocol.

The general adoption of direct coordination models in Internet applications is not suitable. In fact, if two mobile agents communicate directly on a wide scale, they need to be localized by means of complex and highly informed routing protocols. In addition, repeated interactions require stable network connections; this makes communication highly dependent on network reliability and possibly leads to failure or unpredictable delays. Finally, several mobile agent applications are intrinsically dynamic, because of dynamic agent creation, making it difficult to adopt a spatially coupled model that requires the identification of the communication partners.

Direct coordination can be supported by middleware systems, such as CORBA and DCOM, to solve the problems related to agent-location and naming. However, middleware systems are mostly oriented to allow interoperability among heterogeneous software components rather than to define a globally distributed computing environment, and still rely on direct (RPC-based) communications for their services. Therefore, while middleware systems can facilitate the use of independently developed components, their exploitation to enable direct mobile agent coordination would suffer from the above-identified latency and reliability problems.

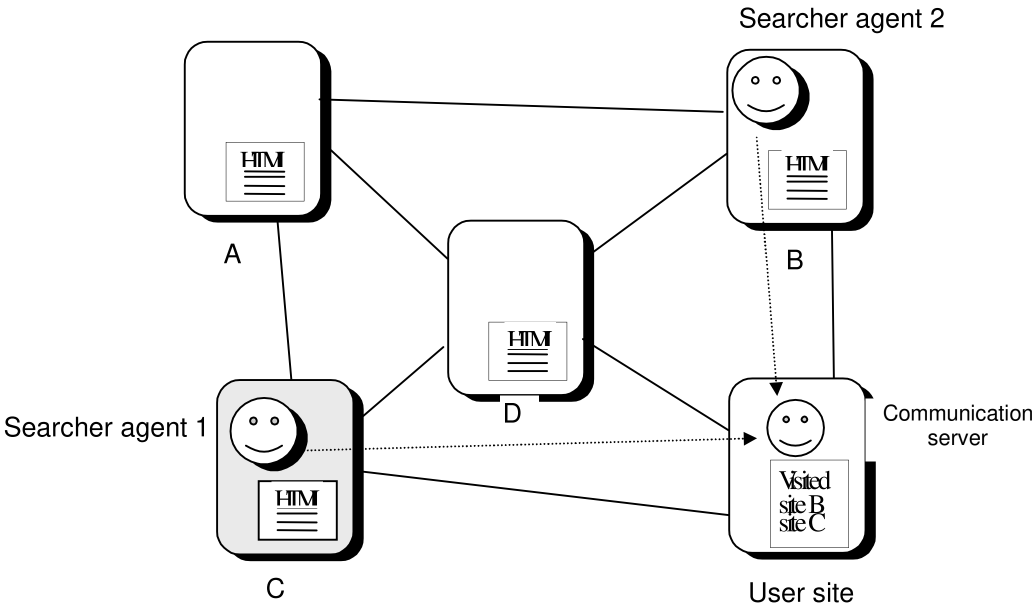

Fig. 3. Searcher agent 1 and searcher agent 2 communicate to the server on the user site that sites B and C have been visited.

In the case study application, agents cannot know how many other agents are in the application, because searcher agents are dynamically created depending on the links found. Thus, choosing a direct coordination model forces odd design choices: "ad hoc" application entities are to be introduced to avoid multiple visits to the same site. As an example, a fixed entity on a given site – let it be the user site – can assume the role of communication server (figure 3). Whenever a searcher agent arrives at a site, it inquires of the communication server whether the site has already been visited. In the positive case, the searcher agent can simply terminate without further investigating in the site. Otherwise, it informs the communication server to mark the site as visited. This solution obviously produces high traffic on the network, makes the communication server a bottleneck and requires stable network connections, thus nullifying most of the advantages deriving from the adoption of the mobile agent paradigm.

In the presence of mobile agents, direct coordination models can be effectively exploited only for agent-to-local environment interactions, by providing local servers on each site (WWW servers in the application example).

3.2 Meeting-Oriented Coordination

Meeting-oriented coordination aims at defining spatially uncoupled models, where agents can interact in the context of meetings without needing to explicitly name the partners involved. Agents join (either explicitly or implicitly) known meeting points; afterwards, they can communicate and synchronize with the other agents participating in such meetings. Ever-open meetings abstract the role of servers in an execution environment; application agents can open further meeting points on need. Meetings are often locally constrained: to avoid the problems related to non-local communication – i.e., unpredictable delay and unreliability – a meeting takes place in a given execution environment and only local agents can participate in it.

The *Ara* mobile agent system implements a typical example of meeting-oriented coordination

[Pei97]. The concept of event-based communication and synchronization, defined by the OMG and implemented in the *MOLE* mobile agent system [Bau98] can be considered as a sophisticated form of meeting-oriented coordination. Specific synchronization objects, which agents must share the reference to, assume the role of meetings. The "access events" to one of these synchronization objects imply agents to implicitly join the meeting.

Meeting-oriented coordination models partially solve the problem of exactly identifying the partners involved, though they cannot achieve fully spatial uncoupling: agents have to share at least the common knowledge of the meeting names. Their major drawback, however, consists in usually enforcing synchronization among interacting agents. Since the schedule and the position of agents cannot be predicted in many applications, the risk of missing interactions is very high. In addition, if the meetings are not locally constrained, they have to be implemented by message-passing, thus inheriting the efficiency and reliability problems of direct coordination models.

In the case study application, the easiest way to exploit this coordination model is via a locally constrained meeting, and by introducing an additional agent for each site visited. When a searcher agent has explored a site, it creates a "meeting" agent before terminating its task. The meeting agent is forced to reside on the creation site and to enter a predefined meeting point: every time any other searcher agent arrives at this site, it locally enters the meeting to check the presence of the meeting agent and then to detect whether the site has already been visited or not (see figure 4). With regard to agent-to-local environment interactions, ever-open meetings can assume the role of local WWW servers to retrieve information on a site.

The meeting-oriented design solution produces less network traffic than the direct coordination one and it is fully distributed. Nevertheless, the solution forces one meeting agent to remain on each site visited, and this leads to several drawbacks: *(i)* a malicious agent can exploit the time it is on a site to send private information to the outside; *(ii)* there is no possibility for the meeting agent to know when to die, unless global garbage collection mechanisms are provided.

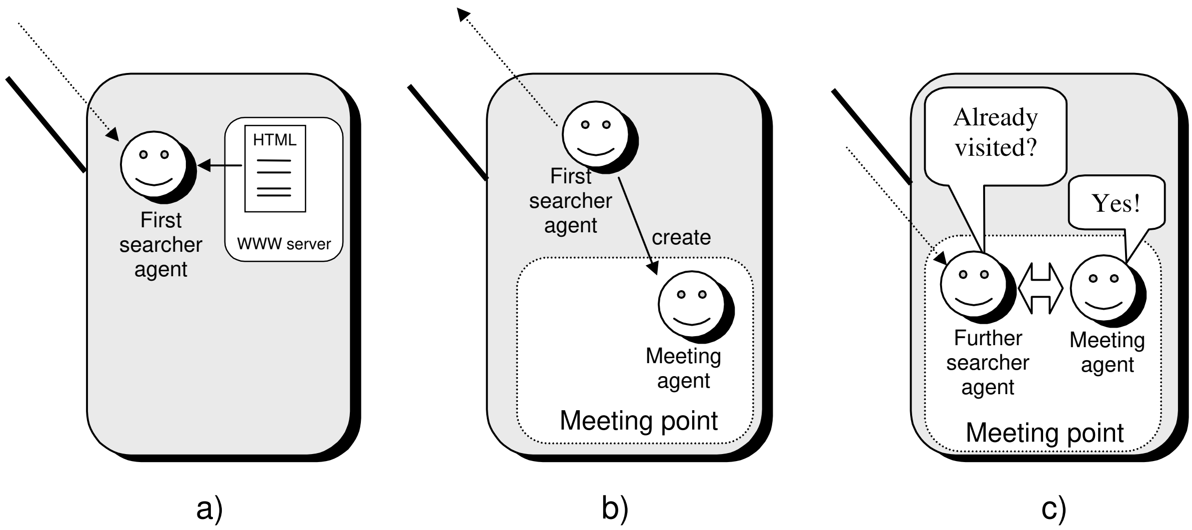

Fig. 4. The first searcher agent arrives at a site and retrieves the documents of interest (a), leaves a meeting agent (b) that will notify further agents (c) by entering a predefined meeting point.

3.3 Blackboard-based Coordination

In *blackboard-based* coordination models, agents interact via shared data spaces, used as common repositories to store and retrieve messages. In this sense, interactions are fully temporally uncoupled but, since agents have to agree on a common message identifier to communicate and exchange data via a blackboard, they are spatially coupled. A local blackboard can be associated with each hosting environment to overcome scalability problems.

Several systems propose and implement blackboard-based coordination models for mobile agent applications. In *Ambit* [CarG98], a formal model for mobile computations, agents can "attach" a named message to a blackboard on a given site; another agent can retrieve and read this message when arriving at the same site. The *ffMAIN* agent system [DomLD97] defines mobile agents that interact – both with each other and with local resources – via an information space accessed through the HTTP protocol, where data – identified via URLs – can be stored, read and extracted.

Blackboard-based coordination achieves temporal uncoupling: messages can be left on blackboards, no matter where the corresponding receivers are or when they are going to read the messages. This suits a mobile scenario in which the position and the scheduling of the agents cannot be either easily monitored or granted. In addition, since all inter-agent interactions are forced to be performed via a blackboard, hosting environments can easily monitor and control them, thus leading to a more secure execution model.

Our case study application can exploit the blackboard model to avoid multiple visits. When a searcher agent arrives on a site, it checks in the blackboard for a “marker” message, to detect whether another agent of the same application (and looking for the same keyword) has already visited the site. Such a message could have the form: (application_id, search_keyword). If the agent finds the message it notices that the site has already been visited; otherwise, it has to leave a marker message in the blackboard (see figure 5).

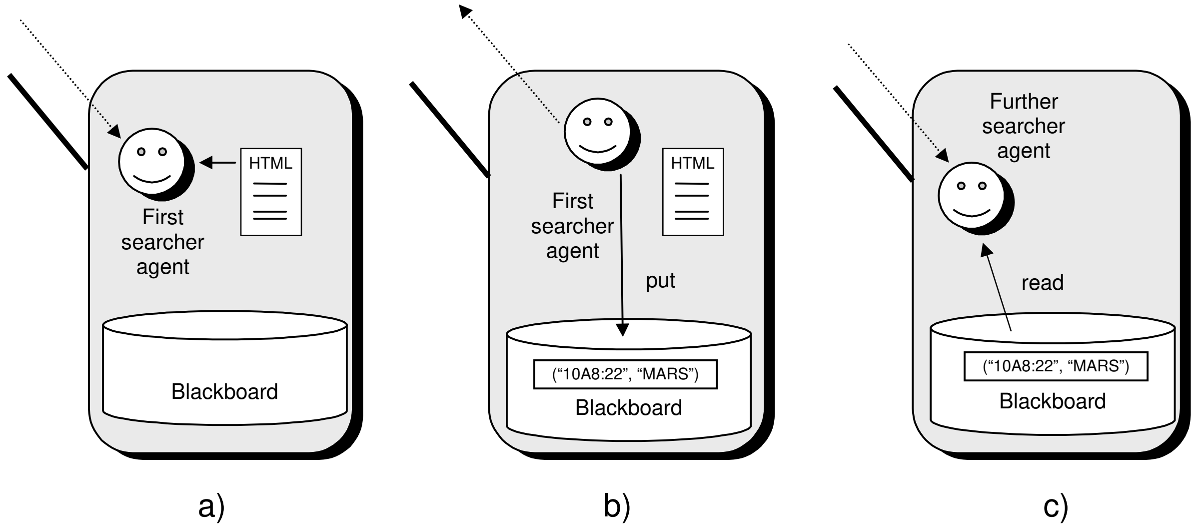

Fig. 5. The first searcher agent arrives at a site and retrieves the documents of interest (a), leaves a marker message in the blackboard (b), which is read by further searcher agents (c).

3.4 Linda-like Coordination

Linda-like coordination models use local tuple spaces as containers of messages. A tuple space is similar to a blackboard, but in addition the accesses are based on *associative mechanisms* [GelC92]. Information is organized in tuples and retrieved in an associative way via a pattern-matching mechanism. This enforces full uncoupling, requiring neither temporal nor spatial agreement.

The concept of an associative blackboard is adopted in the *PageSpace* coordination architecture for interactive Web applications [Cia98]. A multiplicity of distributed object-based tuple spaces can be used by both mobile and fixed agents to store and retrieve object references. Other systems, such as *TuCSoN* and *MARS*, adopt a Linda-like coordination model and go further by defining a reactive tuple space model, as described in the rest of the paper.

Associative coordination suits mobile agent applications. In such a wide and dynamic environment as the Internet, a complete and updated knowledge of the hosting environments and of the other application agents may be difficult or even impossible to achieve. Agents are somehow likely to require pattern-matching mechanisms to deal adaptively with uncertainty, dynamicity and heterogeneity. Consequently, it is worthwhile integrating these mechanisms directly in the information space itself to simplify agent programming and to reduce application complexity.

In our case study application, associative mechanisms may not be necessary for inter-agent interactions: as shown previously, agents know exactly which message to retrieve. However, if the application is made up of several kinds of agents searching for different keywords, an associative mechanism permits an agent to check the tuple space for the presence of marker messages whose keyword field matches the agent's own keyword. Conversely, the associative mechanism makes the agent ignore those messages that have been left by agents searching for different keywords.

With regard to agent-to-local environment interactions, if the pathnames of all public readable files are available in the tuple space, for example in the form (Name, Extension, Date, FilePointer), agents can simply look for tuples corresponding to pathnames matching the "html" extension (see figure 6). The actual content of the document can be retrieved by the field of the tuple that refers to the corresponding file.

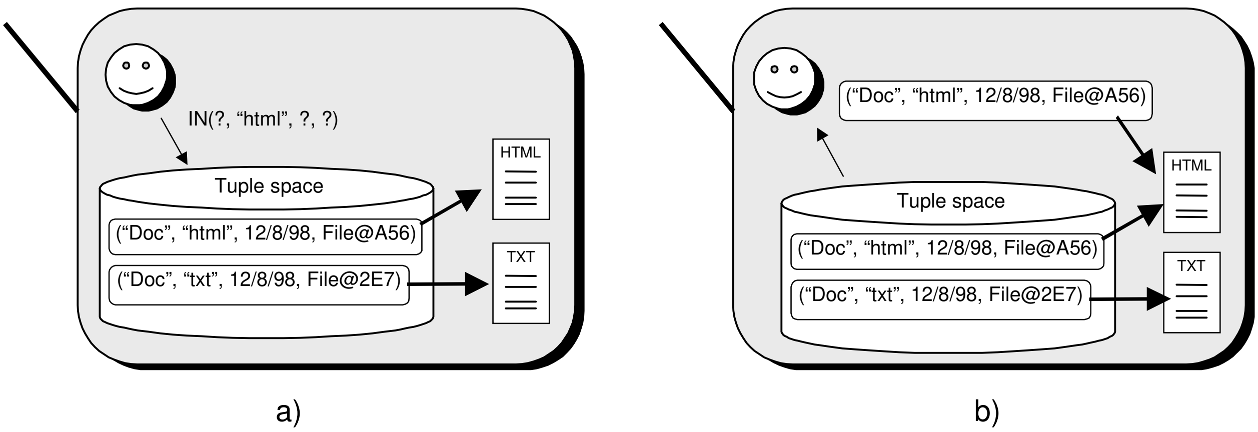

Fig. 6. The searcher agent asks for a tuple matching the given template (a), the tuple space returns a matching tuple with the reference to an HTML file (b).

With regard to agent-to-local environment interactions, a blackboard can be exploited to let agents retrieve the information needed without requiring the presence of specialized resource managers and to let the local environment provide all the data it wants to publish while protecting private data. In the application example, the local environment can provide the pathnames of all its publicly accessible files in the form of messages, without the need of a WWW server. However, because of spatial coupling, the blackboard cannot be effectively exploited to have access to the information about the local HTML pages. In fact, there is no way for an agent to retrieve only the pathnames of the HTML files, but it is bound to read all the messages and successively to select only the ones corresponding to HTML files.

4 Adding Reactivity to Coordination Models

All the above described coordination models can be enriched with reactivity, which stems from the capacity of embodying computational capacity within the coordination media to let it issue specific programmable reactions that can influence the behavior of the agent interactions.

Different proposals in different contexts aim at integrating some degree of reactivity in coordination models. In the area of active networks and active messages [Ten97], the possibility for messages to embody the code needed to handle them can be somehow categorized as an effort toward dynamic and programmable direct coordination models. In the event-based meeting-oriented coordination model [Bau98], synchronization objects can embody specific policies to influence the interactions among the agents involved in a meeting. For example, synchronization objects can be programmed to asynchronously notify events to agents, thus achieving a partial form of temporal uncoupling.

In a *reactive tuple space* model, the tuple space is no longer a mere tuple repository with a built-in and stateless associative mechanism. Instead, tuple spaces can also have their own state and can react with specific actions to the accesses performed by mobile agents. Reactions can access the tuple spaces, change their content and influence the semantics of the agents' accesses.

In this context, the *TuCSoN* model [OmiZ98] defines programmable logic tuple spaces for the coordination of knowledge-oriented mobile agents. Reactions are programmed as first-order logic tuples. The *MARS* system implements a portable reactive tuple space for the coordination of Java-based agents [CabLZ98]. The tuple space interface is defined according to the *JavaSpaces* specification; reactions are programmed as Java methods associated to tuples.

Reactivity of the tuple spaces can provide several advantages in mobile agent applications. It can be used to implement specific local policies for the interactions between the agents and the hosting environments, to achieve better control and to defend the integrity of the environments from malicious agents. In addition, reactions can adapt the semantics of the interactions to the specific characteristics of the hosting environments, thus simplifying the task of agent programming much more than the fixed pattern-matching mechanism of Linda-like coordination models. Even more generally, the capability of adapting the behavior of the tuple spaces to specific accesses adds distributed intelligence to the whole system. As intelligence and adaptive behavior are widely recognized as fundamental characteristics for Internet agents, we claim that the same properties can enrich Internet sites. For example, in the same way that an intelligent agent can dynamically evaluate the characteristics of a hosting environment to plan appropriate patterns of access to its resources, a reactive tuple space can dynamically change both its content and the structure of its tuples, to adapt it to the agent accesses.

In our case study application, a reactive tuple space can be exploited in several interesting ways, even if one takes into account only simple reactions without peculiar intelligent behavior. Let us suppose that one hosting environment represents a local WWW server whose HTML pages are replicated in one or several mirror sites. In this case, when a searcher agent visits this site, not only must the other searcher agents of the same application avoid further visits on the same site, but they should also avoid visiting the mirror sites. A reactive tuple space can be programmed by the local administrator to react to the local insertion of marker messages in order to coordinate itself with mirror sites and there replicate the agents' marker messages. The administrator of a site could also decide to exploit the reactivity to delete the marker messages locally left by searcher agents on the basis of an allowed lifetime. This could be very useful because of the intrinsic difficulty for searcher agents to know whether a marker message on a visited site is further needed by other searcher agents. As a further example, we can recall that the Linda model defines a disruptive "in" operation that extracts matching tuples from the space. Suppose there is a malicious attempt to extract HTML tuples from one site: in this case, the local administrator can decide not to raise any exception, but to program its local tuple space in order to transform a disruptive operation performed by a foreign agent into a non-disruptive operation (read). In this way, the malicious agent proceeds unaware of the innocuous effect of its access and the administrator has preserved the safety of the local environment.

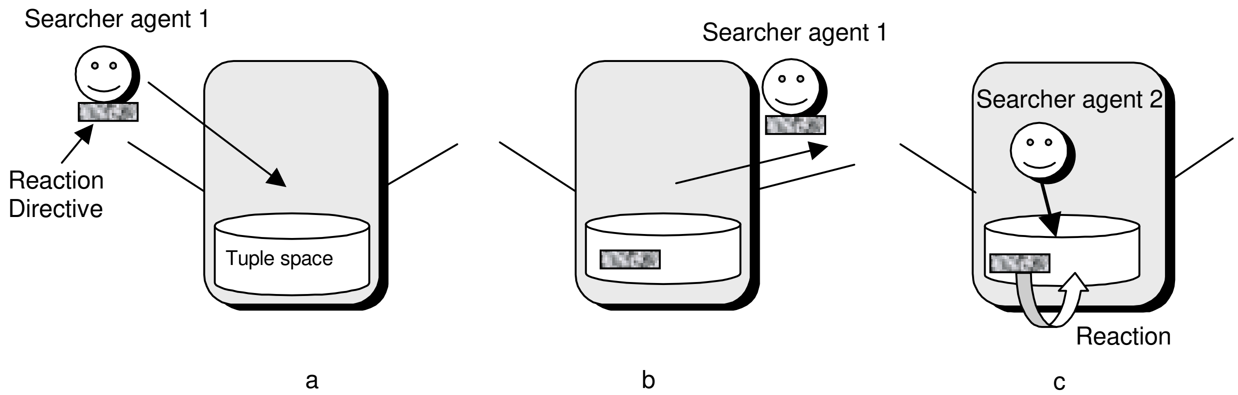

Fig. 7. When an agent arrives to a site (a), it installs to the local blackboard the application specific reaction (b). Further incoming accessing agents will raise the reaction (c).

Further advantages could be provided by giving agents the capability of deciding their own coordination rules, by installing reactions in the tuple spaces of the visited sites (figure 7). For example, in the case study application, a searcher agent could install in the visited tuple space a reaction that denies further incoming searcher agents access those HTML files whose corresponding pages have not been modified since the last visit. In general, the capability of an application to install its own reactions into the tuple spaces permits a clear separation of concerns between algorithmic issues and coordination issues [GelC92]: the agents are in charge of embodying the algorithms to solve the problems; the reactions represent the specific coordination rules for the application. This is likely to notably simplify both application design and agent coding. In our example, the programmer can separate the code for the basic searching algorithm – embodied in the agent code – from, for instance, the code needed to avoid multiple visits – programmed in the form of reactions in the tuple spaces.

5 Conclusions and Open Directions

The choice of the coordination model has a great impact on the design of mobile agent applications. In particular, we argue that a Linda-like coordination model – possibly enriched with programmable reactive capabilities – can lead to a clean, flexible and scalable application design, and we therefore suggest its adoption as the primary coordination model for mobile agent applications.

Of course, we do not claim that reactive tuple spaces have to be the exclusive model for any kind of application: for example, performance reasons may sometimes suggest using direct communication among distant agents instead of forcing agents to move to the same site to interact via a tuple space. In this context, a more quantitative analysis – either constructed via simulation tools or on the field, by measuring the performances of different design solutions – would help designers to identify these situations and select the most suitable approach. Additional issues, neglected by this paper, have to be faced before both mobile agents and the presented coordination technologies can gain wide acceptance. In particular:

- effective security policies must be defined to rule agent accesses to tuple space and, in the case of reactive tuple spaces, to rule the spread of application-specific reactions over foreign Internet sites;
- letting mobile agents access a node and its local resources implies resource consumption and, therefore, should be properly monitored and regulated, possibly via the definition of coordination models enriched with contracting and currency-exchange capabilities.
- any proposal must be well integrated with the current Web infrastructure, made up of CORBA applications, WWW services and browsers, and made compliant with the existing standards, such as HTTP and XML, to facilitate interoperability with existing components.

All the above issues represent interesting starting points for future study.

Acknowledgements

We thank the anonymous referees for their helpful suggestions. This work has been supported by the Italian MURST in the framework of the project “MOSAICO – Design Methodologies and Tools of High-Performance Systems for Distributed Applications”.

References

[Bau98] J. Baumann, F. Hohl, K. Rothermel, M. Straßer, “Mole - Concepts of a Mobile Agent System”, *The World Wide Web Journal,* Vol. 1, No. 3, pp. 123-137, 1998.

[CabLZ98] G. Cabri, L. Leonardi, F. Zambonelli, “Reactive Tuple Spaces for Mobile Agent Coordination”, *Proceedings of the 2nd International Workshop on Mobile Agents*, Stuttgart (D), Lecture Notes in Computer Science, No. 1477, pp. 237-248, September 1998.

[CarG98] L. Cardelli, D. Gordon, “Mobile Ambients”, *Foundations of Software Science and Computational Structures*, Lecture Notes in Computer Science, No. 1378, Springer-Verlag (D), pp. 140-155, 1998.

[Cia98] P. Ciancarini, R. Tolksdorf, F. Vitali, D. Rossi, A. Knoche., "Coordinating Multi-Agents Applications on the WWW: a Reference Architecture", *IEEE Transactions on Software Engineering*, Vol. 24, No. 8, pp. 362-375, May 1998.

[DomLD97] P. Domel, A. Lingnau, O. Drobnik, "Mobile Agent Interaction in Heterogeneous Environment", *Proceedings of the 1st International Workshop on Mobile Agents*, Lecture Notes in Computer Science, Springer-Verlag (D), No. 1219, pp. 136-148, April 1997.

[Fug98] A. Fuggetta, G. Picco, G. Vigna, "Understanding Code Mobility", *IEEE Transactions on Software Engineering*, Vol. 24, No. 5, pp. 342-361, May 1998.

[GelC92] D. Gelernter, N. Carriero, "Coordination Languages and Their Significance", *Communications of the ACM*, Vol. 35, No. 2, pp. 96-107, February 1992.

[KarT98] N. M. Karnik, A. R. Tripathi, "Design Issues in Mobile-Agent Programming Systems", *IEEE Concurrency*, Vol. 6, No. 3, pp. 52-61, July-September 1998.

[OmiZ98] A. Omicini, F. Zambonelli, "TuCSoN: a Coordination Model for Mobile Agents", *Journal of Internet Research*, Vol. 8, No. 5, pp. 400-413, 1998.

[PapA98] G. A. Papadopoulos and F. Arbab, "Coordination Models and Languages", *Advances in Computers*, Vol. 46, pp. 329-400, Academic Press, Aug. 1998.

[Pei97] H. Peine, "Ara - Agents for Remote Action", in William R. Cockayne and Michael Zyda: *Mobile Agents: Explanations and Examples*, Manning/Prentice Hall, 1997.

[Ten97] D. Tennenhouse M. Smith, W. D. Sincoskie, D. J. Wetherall, G. J. Minden, "A Survey of Active Network Research", *IEEE Communications*, Vol. 35, No. 1, pp. 80-86, January 1997.

Selected URLs on Mobile Agents

Ambit	http://www.luca.demon.co.uk/Ambit/Ambit.html
ARA	http://www.uni-kl.de/AG-Nehmer/index_e.html
D'Agents	http://www.cs.dartmouth.edu/~agent
General Magic	http://www.genmagic.com
IBM Aglets	http://aglets.trl.ibm.co.jp
MARS	http://sirio.dsi.unimo.it/MOON
MOLE	http://www.informatik.uni-stuttgart.de/ipvr/vs/projekte/mole.html
PageSpace	http://flp.cs.tu-berlin.de/~pagespc/
TuCSoN	http://www-lia.deis.unibo.it/Research