# DESIGN OF DISTRIBUTED SYSTEM BASED ON TECHNOLOGY OF MOBILE AGENTS[1]

Martin TOMÁŠEK
Department of Computers and Informatics, Faculty of Electrical Engineering and Informatics, Technical University of Košice, Letná 9, 042 00 Košice, Slovakia, phone: (+421 55) 602 25 50, fax: (+421 55) 633 01 15,
e-mail: martin.tomasek@tuke.sk

**SUMMARY**

*An information system operating in environment that is highly distributed, heterogeneous, extremely dynamic and comprising many autonomous nodes must handle several emerging problems. Concept of high-level communication is the most important problem within the flexible distributed system. Our research on distributed systems in heterogeneous computational environment tries to combine technologies as software agents and mobile code. The paper presents principles and an architecture where mobile agents could move across distributed environments, integrate with local resources and other mobile agents, and communicate with the users. There are also two concepts of inter-agent communication defined: session-oriented communication scheme and anonymous communication scheme.*

***Keywords:*** *distributed system, mobile code, agent, agent technology, software agent, mobile agent, agent-oriented programming, agent-based system.*

## 1. INTRODUCTION

Computational environment as Internet is highly distributed, heterogeneous, extremely dynamic, and comprising a large number of autonomous nodes [11]. An information system operating in such an environment must handle several emerging problems. The most traditional architecture on the Internet is the client-server model, but it is limited and restrictive. Some nodes will want to take both roles (client and server) depending on which node they are interacting with. There are several forms of heterogeneity in this environment, e.g. different platforms, different data formats, the capabilities of different information services, and the implementation technologies. The information system must handle these problems to be able to work effectively.

Many software technologies have matured to participate in and contribute to the Internet environment. The technologies as event simulation, applied natural language processing, knowledge-based reasoning, advanced information retrieval, speech processing, etc. are already prepared for this. However, there is a lack of tools and techniques for constructing intelligent clients and servers or for building agent-based software in general.

Agent technology [4] can address each of the problems mentioned above. When we describe these agents as intelligent, we refer to their ability to: communicate with each other using an expressive communication language, work together cooperatively to accomplish complex goals, act on their own initiative, and use local information and knowledge to manage resources and handle requests from other agents. Agents has attracted a lot of attention in artificial intelligence as well as distributed systems circles.

Mobile agents [5] may be able to move between computers in a network while carrying along their internal state in order to resume work at a new location. Though first prototype systems and even products exist, the architecture of mobile agents systems is not well understood today and hence needs more investigation.

## 2. SOFTWARE AGENTS INTERACTION

For the purpose of software agents interaction they must establish communication relationship from time to time. We call it *session-oriented communication* and agents must make an appointment, then they can communicate using message mechanisms.

In the general case, a group of agents performing a common task may be arbitrarily structured and highly dynamic. In those environments, one cannot assume that an agent that wants to synchronize on an event knows a prior which agent is responsible for generating this event. Therefor, it is needed to design the concept of *anonymous communication*, allowing agents to generate events and register for the events they are interested in.

We could address many of the difficulties of communication between software agents by giving them a common language. In linguistic terms, this means that they would share a common syntax, semantics and communication protocol.

Getting information processes to share a common syntax is a major problem. There is no universally accepted language in which to represent information and queries. We found many languages

[1] This paper was supported by grant Nr. 1/6278/99 of the Slovak Grant Agency.

such as KIF [3], extended SQL, LOOM [6] and XML [13] together with RDF [12] or Topic Maps [10] which have their supporters, and they will become or already are standardized and recommended to represent and to exchange knowledge. As a result, it is currently necessary to say that two agents can communicate with each other if they have a common representation language or use languages that are inter-translatable.

Assuming a common or translatable language, it is still necessary for communicating agents to share a framework of knowledge in order to interpret message they exchange. This is not really a shared semantics, but a *shared ontology* [7]. Shared ontology is under development in many important application domains such as planning, scheduling, data mining, biology and medicine.

KQML [1] is a language that is designed to support interactions among intelligent software agents. It was developed by the ARPA supported Knowledge Sharing Effort [7] and separately implemented by several research groups. It has been successfully used to implement a variety of information systems using different software architectures. It is concerned primarily with communication protocols (and secondarily with semantics). It is a language and a set of protocols that support computer programs in identifying, connecting with and exchanging information with other programs. Current KQML implementations use standard communication and messaging protocols as a transport layer, including TCP/IP, e-mail, HTTP and CORBA. In designing KQML, the main goal was to build in the primitives necessary to support all of the agent architectures in use.

## 3. DESIGN OF DISTRIBUTED SYSTEM ARCHITECTURE

The architecture of the distributed system (Fig. 1) is in principle multi-agent system and comprises both static and mobile agents [9]. We do not use the traditional design of the distributed systems, where an environment is strictly homogeneous and implemented by standards as DCOM/COM or CORBA. Our design follows principles of mobile code systems [2]. The environment is heterogeneous and can differ in each place of the distributed systems. Components of the system are presented as autonomous processes – software agents. Such a system then can be easily nested in the heterogeneous environment as Internet and fulfil the distributed tasks there (e.g. information retrieval, e-commerce tasks, etc.).

The next chapters take each type of agent from the architecture and describe its activities and purpose within the infrastructure.

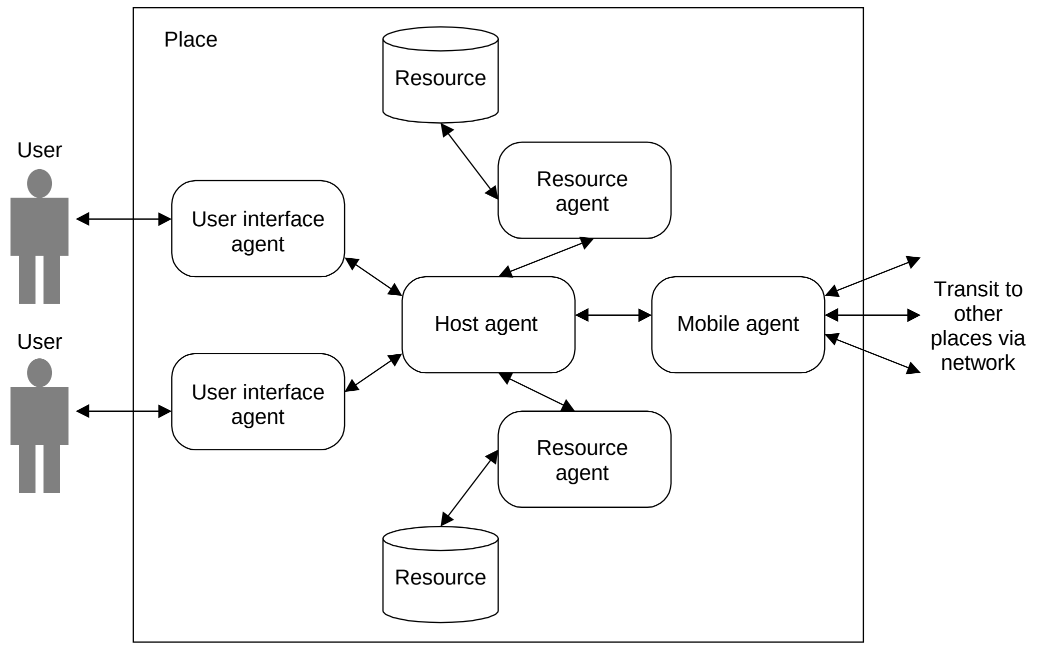

**Fig. 1:** Architecture of the distributed system based on software agents

### 3.1 Host Agent

The host agent is a static agent that coordinates the activities that occur within a place. The host agent can offer a number of services related to information resources, users and agents within a given place.

It provides a migration service to mobile agents wishing to leave the place. The host agent is responsible for negotiation passage to the recipient place and for ensuring that the mobile agent is transferred successfully. If the migration of a mobile agent is rejected, then it is the duty of the host agent to restart the agent to allow it to choose a new destination.

It authenticates and performs a validation check on mobile agents wishing to enter the place. Agents that cannot be authenticated or fail the validation check are rejected.

It launches received mobile agents in a suitable run-time environment, which will be related to the amount of trust given to the mobile agent.

It mediates access to information resources at a place level. When a mobile agent enters a place, the host agent performs a security check upon it and allocates the mobile agent a permission. This permission is used by resource agents to determine whether to allow access to a resource.

It provides a central registration area where static and mobile agents can register and advertise their resources and interests. A place is also a meeting point which allows agents to gather and share information to resolve their goals.

It is the initial point of contact within the place. All messages between static and mobile agents are initially routed through the host agent, which allows agents to communicate in asynchronous fashion.

It advertises public information resources and the presence of mobile agents to ask agents from both inside and outside of the place. In this way, mobile agents can interrogate the contents of a particular place before moving. It also provides a mechanism for agents to locate each other.

## 3.2 Resource Agent

The purpose of a resource agent is to mediate access to a particular information resource for a mobile agent. The resource agent understands how to access the resource and also understands the permission structures associated with the resource.

It is fully conversant with the protocols of the information resource. The information resource should be completely accessible by mobile agents, so that user intervention directly on the resource is not required.

It provides an ontological description for each of the services offered by the resource. These services are the methods by which mobile agents interrogate, update and manage resource.

It advertises the presence of the information resource by registering the services available with the host agent. In this way, mobile agents can be aware of what information resources are present within a place before they migrate to a particular place.

The information resources are systems, which present an external interface through which they communicate or can be accessed. This allows resource agents to be developed for any type of resource (electronic mail, USENET news, databases, the WWW and so on).

## 3.3 Mobile Agent

Mobile agents are the components within the architecture, which can migrate between places. Initially, a user launches a mobile agent within a given place, which is called *home place*.

Mobile agents determine where to migrate to next by initially querying the local host agent for a list of places of which it is aware. The mobile agent can then use this information to contact each host agent and determine which one offers a set of information resources which are compatible with its own goal set.

Mobile agents are authenticated by an electronic signature that they carry. This signature may be encrypted but must certainly be verifiable with the home place of the mobile agent or a third-party place.

Mobile agents possess the characteristic of persistence. They use migration as a mechanism to achieve longevity. In this way, they are not reliant on the home place that launched them. This is particularly useful in mobile computing, where the user is only connected to the network for short periods of time.

Mobile agents may communicate not only with resource agents, but also with other mobile agents to achieve their goals.

Mobile agents should transit the results of their findings and actions to their users on the home place.

## 3.4 User Interface Agent

The user interface agent is a “personal assistant” who is collaborating with the user in the mobile agent architecture. It is essentially an interface agent that perform some tasks.

It launches mobile agents on behalf of the user and tracks their progress and location. It provides mobile agents with communication point through which they can return the results of their tasks.

It organizes and pre-processes information returned from mobile agents into a form that is suitable for the user. This may involve filtering out replicated information, presenting urgent information to the user quickly, rendering information using tools that the user is most familiar with.

It provides host agents with the authentication credentials of a user’s mobile agents. It is the task of the user interface agent to ensure that the requesting entity is itself and authentic host agent, to make certain that authentication credentials are not given to unauthorized systems.

## 4. DESIGN OF MOBILE AGENT COMMUNICATION

Considering interaction of mobile agents within the architecture, we have to distinguish among following types of communication [8]:

*Mobile agent – static agent interaction.* Since the static agents are the representatives of services in the agent world, the style of interaction is typically client-server. Consequently, static agents are requested by issuing requests, results are reported by responses.

*Mobile agent – mobile agent interaction.* This type of communication significantly differs from the previous one. The role of the communication partners is peer-to-peer rather then client-server. The communication patterns that occur in this type of interaction might not be limited to request-response only. The required degree of flexibility is provided by message passing scheme.

*Anonymous mobile agent interaction.* There are situations, where sender does not know the identities of the agents that are interested in the sent message. Anonymous type of communication is supported by group of communication protocols. Senders send out event message anonymously, and receivers explicitly register for those events they are interested.

### 4.1 Session-Oriented Communication Scheme

A session defines a communication relationship between a pair of agents. Agents that want to communicate with each other, must establish a session before the actual communication can be started. After session setup, the agents can interact by message passing. Sessions have the following characteristics:

Two agents participating in a session are not required to reside at the same place.

In order to preserve the autonomy of agents, each session peer must explicitly agree to participate in the session.

While an agent is involved in a session, it is not supposed to move to another place, to avoid the need for message forwarding, etc.

#### *Establishment*

To set up session, two functions should be offered: *PassiveSession* and *ActiveSession*. The first function is used by agents to express, that they are willing to participate in a session. The second one is used to issue synchronous setup request. Using this function, caller is blocked until either the session is successfully established or a timeout occurs.

Function *ActiveSession* returns reference to the newly created session object to the caller. Input parameters are identifier of the place, where the desired session peer is expected, identifier of the peer mobile agent or group of mobile agents, and timeout interval.

Function *PassiveSession* does not return anything. The input parameters are optional and they are identifier of the place and identifier of the mobile agent or group of mobile agents. If the parameters are not specified, the caller expresses its willingness to establish a session with any mobile agent residing at any place.

The session is established if two session requests $R_A$ and $R_B$ of two mobile agents $A$ and $B$ match. They match if the first parameter (place identifier) in $R_A$ and $R_B$ identifies location of $B$ and $A$, and if the second parameter (mobile agent or group identifier) qualifies $B$ and $A$.

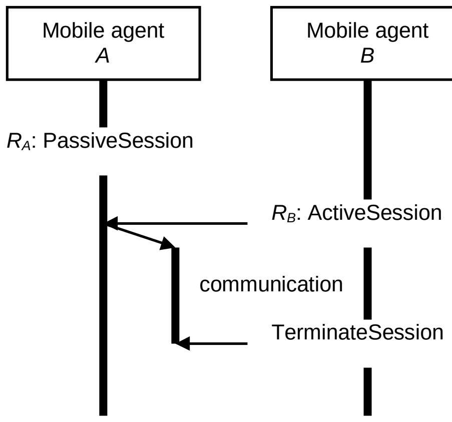

**Fig. 2:** Client-server communication

A combination of these two functions allows a client-server style of communication (Fig. 2). The agent playing server role issues *PassiveSession* when it is ready to receive requests. The agent playing client role invokes *ActiveSession*, this causes the server agent to establish a session with the caller and assigns a thread for handling this session. The server agent can handle more sessions in parallel and they are established purely by client agents.

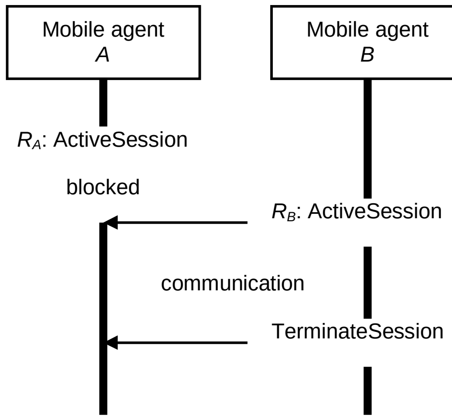

**Fig. 3:** Peer-to-peer communication

If both mobile agents invoke *ActiveSession* requests this corresponds to rendezvous (Fig. 3). Both requestors are blocked until the session is established or timeout occurs. This type of session establishment is suited for agents that want to use peer-to-peer communication.

***Communication***

The most appropriate communication paradigm for session-oriented interaction is message passing. The available communication mechanism can be realized by messaging objects.

*Messaging Objects* can be conveyed asynchronously between the participants of a session. Messages are sent by calling the *SendMessage* method. For receiving messages the *ReceiveMessage* method is provided. The *ReceiveMessage* method blocks until a message is received or timeout occurs, whatever happens first.

***Termination***

At any time, a session can be terminated unilaterally by each of the both session participants, either explicitly or implicitly. A session is terminated explicitly by calling *TerminateSession* function, and implicitly when a session participant moves to another place. All resources associated with the terminated session are released.

### 4.2 Anonymous Communication Scheme

This scheme is used for cooperation of mobile agents in a group to perform a user task. For anonymous communication we model an application as a sequence of reactions to events that in turn generate new events. Events may be defined by user, application or system. An event-based view maps very closely onto real life, and any programming primitives that support event-based concepts tend to be more flexible in modeling a given problem.

Our anonymous communication scheme also follows the specification of event service defined by OMG.

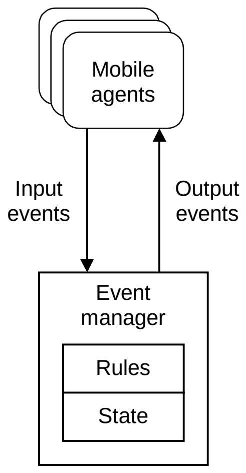

**Fig. 4:** Event handling by Event Manager

*Events* are objects of a specific type, containing some information. Events are generated by “producers” and transferred to the “consumers”. Mobile agents take both roles: producers and consumers. Both know which events to produce or consume, they share common knowledge of used event types in an interaction group. Event types are negotiated at startup time and mobile agents are configured before a migration.

*Event Managers* (Fig. 4) are objects that could be defined as active components responsible for synchronization of an entire application. The manager monitors input events and generates output events according defined internal rules and internal state information.

Rules consist of a condition and an action part. The condition part is a logical expression composed of event types and state information of the manager. If the logical condition becomes true, the action part is triggered. The action part consists of commands (send output event, change internal state information). The internal state information is a set of variables.

Event manager is well-suited to model dependencies within mobile agents groups. Relationships between agents are expressed by the manager’s internal rules and can be defined in terms of success. Mobile agents participating in such groups send success events after they have accomplished their task.

## 5. CONCLUSION

Usage of autonomous software agents is also efficient way to develop large heterogeneous distributed agent-oriented information systems. The problem of higher-level communication is very important to have a flexible distributed system. In this paper we discussed main problems of the intelligent communication and knowledge exchange and e.g. KQML is a language and associated protocol by which intelligent agents can communicate to share information and knowledge.

Our work is focused on distributed systems nested in a big heterogeneous environment as Internet. We designed architecture of the distributed system based on the paradigms of agent technology and mobile code. It is not specific distributed system and it can handle any distributed tasks within the environment. To handle mobile agent interactions we defined two types of communication schemes: session-oriented interaction and anonymous interaction.

## REFERENCES

[1] Finin, T., Fritzson, R., McKay, D., McEntire, R.: KQML as an Agent Communication Language, Proceedings of the Third International Conference on Information and

knowledge Management (CIKM ’94), ACM Press, November 1994.

[2] Fuggetta, A., Picco, G. P., Vigna, G.: Understanding Code Mobility, Software Engineering, 24(5), May 1998, pp. 342 – 361.

[3] Genesereth, M., Fikes, R., et al.: Knowledge Interchange Format, Version 3.0 Reference Manual, Technical Report, Computer Science Department, Stanford University, 1992.

[4] Jennings, N. R., Wooldridge, M. J.: Applications of Intelligent Agents, In: Jennings, N. R., Wooldridge, M. J. (eds.): Agent Technology: Foundations, Applications, and Markets, Springer, Berlin, Heidelberg, New York, 1998, pp. 3 – 28.

[5] Karnik, N. M., Tripathi, A. R.: System Support for Mobile Agents, Proceedings of the 3rd ECOOP Workshop on Mobile Object Systems, MOS‘97, Jyväskylä, Finland, 1997.

[6] MacGregor, R., Bates, R.: The LOOM Knowledge Representation Language, technical Report ISI/RS-87-188, USC/ISI, 1987.

[7] Patil, R., Fikes, R., Patel-Schneider, P., McKay, D., Finin, T., Gruber, T., Neches, R.: The DARPA Knowledge Sharing Effort: Progress Report, In: Nebel, B., Rich, C., Swartout, W. (eds.): Principle of Knowledge Representation and Reasoning: Proceedings of the Third International Conference (KR’92), San Mateo, USA, November 1992.

[8] Tomášek, M.: Concepts for Mobile Agents Interaction, In: Proceedings of Scientific Conference with International Participation Computer Engeneering and Informatics (CEI’99), Herľany, Slovakia, 14 – 15 October 1999, pp. 259 – 264.

[9] Tomášek, M.: Designing mobile agent architecture, In: Proceedings of the EMES’99 International Conference, (Z. Maghiar Ed.), Oradea, Romania, 27 – 30 May 1999, pp. 194 – 200.

[10] Topicmaps.net, http://www.topicmaps.net

[11] Vokorokos, L., Kubiš, Ľ.: Information systems for administration of scientific-research projects resolved at academic glebe. AT&P Journal, 12/2000, HMH, pp. 58-59, 71. ISSN 1335-2237, Slovakia.

[12] W3C Semantic Web Activity: Resource Description Framework (RDF), http://www.w3.org/RDF

[13] XML.ORG – The XML Industry Portal, http://www.xml.org

## BIOGRAPHY

**Martin Tomášek** was born 1975 in Košice, Slovakia. He received the master degree in computer science in 1998 at the Faculty of Electrical Engineering and Informatics of the Technical University of Košice, Slovakia. He is presently a PhD student at the Department of Computers and Informatics of the Faculty of Electrical Engineering and Informatics of the Technical University of Košice, Slovakia and his study field is software and information systems. The subject of his PhD dissertation focuses on distributed systems based on agent technology and mobile code. He authored several publications on mobile agents and he participates in research and educational projects of the university. His research interests include mobile code, software agents, distributive programming and design of information systems.