UDC:007.52
Original scientific paper:

MULTI-AGENT APPLICATIONS DESIGN SUPPORT

Marek Paralič, Milan Krokavec
Technical University of Košice,
Faculty of Electrical Engineering and Informatics, Košice
{Marek.Paralic, Milan.krokavec}@tuke.sk

Abstract: *Nowadays distributed applications have to fulfil challenging criterions in respect of flexible utilization. Availability of different network services should be spread through a number of mobile devices like PDAs and mobile phones. For such a context the design and implementation of distributed applications grew over classical client-server architecture. In this paper the utilization of the mobile agent paradigm for solving these kinds of problems is described. In this paper we start with a brief description of the ESMA system for support of mobile agent based applications designed and implemented at the Department of Computers and Informatics, TU of Košice. Then we continue with the methodology for design and implementation of mobile agent based applications based on this system. Finally, the multi-agent dimension of new solution is emphasized and illustrated on a simple example of on-line registration system.*

Keywords: *multi-agent systems and applications, mobile programming, distributed programming.*

1. INTRODUCTION

The term multi-agent system is now used for all types of systems composed of multiple autonomous components showing the following characteristics [6]: (i) each agent has incomplete capabilities to solve a problem, (ii) there is no global system control, (iii) data is decentralized and (iv) computation is asynchronous. Multi-agent architectures are distributed systems in which agent is: able to act autonomously, able to reason and make decisions about the environment in which it is embedded, and able to interact with other agents. The distributed and autonomous nature of agent systems potentially supports flexibility and robustness in system operation and organization, particularly when these are coupled with dynamic system behaviour [12].

Basic dimensions to characterize agent software are: reactivity, autonomy, proactivity, continuity, interactivity, adaptability, cooperativity and robustness [2]. There are of course other properties that are not critical to a general notion of agent-hood, but that can be used as a basis for classification of agents. One example is agent mobility - mobile agents can move between locations in a network. Mobile agents are capable to decide what locations in a computer network they visit and what actions they take once being there. This ability is given in the source code of the mobile agent (implicit behaviour) or by the agent's itinerary

set dynamically (explicit order).

Mobility however becomes a necessary condition of agent-hood in certain environments, where hard real-time constraints exist at remote servers, where large volumes of data must be processed, where processing resources of agent servers are over-utilized, and where the communication channel is costly or hostile. From the distributed programming point of view mobile agents are a promising paradigm that is meanwhile recognized as a powerful tool for design and implementation of sophisticated distributed systems with dynamic changeable contextual physical and software environment. Usual benefits of mobile agents are (i) reduced network load, (ii) overcome of the network latency, (iii) encapsulation of different protocols, (iv) asynchronous and autonomous execution, and (v) natural heterogeneity.

Despite of increased interest of the last decade, the research in area of the multi-agent systems (MASs) is over 30 years old. MAS researchers have developed some sophisticated theories and technologies for multi-agent interaction and coordination, coalition formation, conflict resolution, dynamic organization and reorganization, robustness, multi-agent learning and real-time multi-agent behaviour [6]. But from practical point of view a number of scientific issues – such as managing dynamic heterogeneity [9], exception handling [7] and agent-oriented methodologies [1] – are under-explored. Recently the advantages of agent-oriented design have had a significant impact: the identification of interaction and coordination as a central focus of multi-agent system design. That is interaction and coordination play a central role in the analysis and design of multi-agent systems and makes the multi-agent approach significantly different from other approaches toward building distributed or intelligent systems. This realization lead to several new methodologies for building multi-agent systems that focused on the interaction between agents as the critical design aspect (e.g. agent-oriented methodologies *MaSE* [1] and *Gaia* [16]).

In this paper we describe a method, how the *Experimental System for Mobile Agents* (ESMA) [11] - designed and implemented at the Department of Computers and Informatics (FEI TU of Košice) - can be used for multi-agent applications creation. The paper is structured as follows. In section 2 the ESMA system is briefly characterized, section 3 introduces the methodology for design of multi-agent applications that utilizes mobile agents, section 4 describes a simple example of reservation system designed and implemented as a multi-agent application, section 5 concludes the paper and outlines our current and future research directions.

2. ESMA – EXPERIMENATAL SYSTEM FOR MOBILE AGENTS

The main goal when designing of ESMA was to combine the power of high-level distributed programming support with the mobile agent paradigm. It utilizes expressiveness and formal foundation of concurrent constraint programming to solve the problem of system support for dynamic re-binding of not transferable resources and inter-agent collaboration based on logic variables. Proposed solutions make the agent-based programming easier and more straightforward and on the other hand offer a basis for building more sophisticated multi-agent applications. The framework was implemented in the *Distributed Oz* (*Oz 3*) programming language [13] (in the Mozart programming system [15]) that appears to the programmer as concurrent, object-oriented language with data-flow synchronization. Mozart offers simultaneously the advantage of a *true distributed system* and the means for building a m*obile code system* [3]. In the following subsections the experimental framework will be introduced: mobile agent environment built from servers, its services and key properties of mobile agent as a programming entity.

2.1. MOBILE AGENT ENVIRONMENT (MAE)

The basic functions offered by a mobile agent environment are transport and management of agents. In today's agent systems these services are offered by servers, which must be installed and running on every host computer that should be available for the mobile agents. Similarly our experimental framework contains two common modules: *MAE* and *MaeAdmin*. Running the MAE module on a host computer makes a basic environment for mobile agents available, whereby MaeAdmin is an optional administration tool. MAE offers the following functionality:

- Creation of a mobile agent with a unique identity,
- Transport of an agent,
- Sending a message to an agent (possibly on another host),
- Getting the status information about an agent,
- System recovery after crash and
- Foreign agents acceptance in case of running as docking server.

Every agent created on a local MAE is a home agent for this MAE. On all other sites it will visit, it gains the status of a foreign agent. The MAE stores information about all its home agents and currently available foreign agents in local database. The information about foreign agents is stored in the system only during time between successful receiving from the previous host and successful sending an agent to the next one.

For the programmer of mobile agent based application, the mobile agent environment is represented by the MAE module, which must be imported. Importing a MAE module causes either the launching of a new environment, recovering with initialisation values from persistent database for home and foreign agents and already connected other MAE servers after crash, or getting a reference to an already started MAE local server. The last process is realized also by resuming every incoming mobile agent. Thus, an agent gets access to the key services of the mobile agent environment. The possibility of dynamic loading and installation of first-class modules is thereby very important. Beside the loading and connecting of the local MAE module the process of resuming of a mobile agent includes:

- Updating the agent's information about the current location,
- Setting appropriate task for the current location,
- Creating a new agent port for the high-level Oz communication and starting the service on it,
- Adding information into the database of foreign agents,
- Setting the appropriate application interface,
- Sending a commitment to the previous site (if all steps were successful) and
- Sending a log message to the owner of the agent (information about current location).

From the implementation point of view the agent is an object, for which by each move the agent's class and current data state are separately saved and transported. This adopts the persistent procedure mechanism to enable independence from home server and system recovery after crash without the need for destroying all foreign and home agents present on the crashed location. The process of resuming an agent after move is extended with creating a new object of agent class and restoring the last data state from transported persistent data.

The communication between MAEs is realized in two layers: the first layer uses TCP sockets for exchanging of tickets for Oz ports. Oz ports then build a second, high-level communication layer, which can take advantage of Oz space data transparency. The Oz

space offers the possibility to transparent copying of stateless entities and creating references for worldwide unique stateful entities. These possibilities can be fully utilized especially by the inter-agent communication.

2.2 MOBILE AGENT (MA)

Mobile agents in our framework are objects derived from a special class *MobileAgent* that offers the basic facilities expected by the MAE from agents. There are several features, attributes and methods of this class. The most important are:

- *Name*: attribute with the locally unique name at the agent's home MAE,
- *Itinerary*: attribute with a list of host/message pairs with hosts that should be visited by the mobile agent and the first message, that it will receive there,
- *AgentTicket*: attribute with a ticket to an agent specific Oz port for incoming messages from other agents/applications,
- *Owner*, *OwnerTicket*: attributes with information about the address and communication port of the agent owner site,
- *Task*: attribute with the first message sent to the agent at the current location,
- *Environment*: attribute represents access to the local, application-specific resources after they are successfully loaded by resuming of an agent at a new location,
- *mae*: attribute with the current locally accessible mobile agent environment,
- *commonInterface*: attribute that represents the path to dynamic loadable resources for the agent,
- *runServeStream*: a method that starts to serve current agent stream connected to the agent communication port,
- *addIItem*, *removeIItemL*, *removeIItemTask*: methods for dynamic modification of the agents itinerary (add an item at the end, remove item according to a location or according to a task).

3. DESIGN AND IMPLEMENTATION OF MULTI-AGENT APPLICATIONS

Creating an mobile agent based application that could be possibly multi-agent application in our framework is straightforward and requires the following steps:

1. Identifying all fixed, not transferable resources needed by the application (i.e. their type) by means of abstract names, and identifying parts of the transferable agent state.
2. Design and implementation of application-specific classes, which are derived from the *MobileAgent* class and deal with agent state and other resources through their abstract names and communicates with the environment and possibly other agents.
3. Design and implementation of an application that creates one or more instances of mobile agents, specifies their itinerary, sends them away, waits until they finish their jobs (or until the owner stops their work), and processes the results.
4. Design and installation of special environment modules (functors) in a compiled form, which map the abstract sources of MA to the real local resources of the host computer, that should enable the execution of the MA based application.

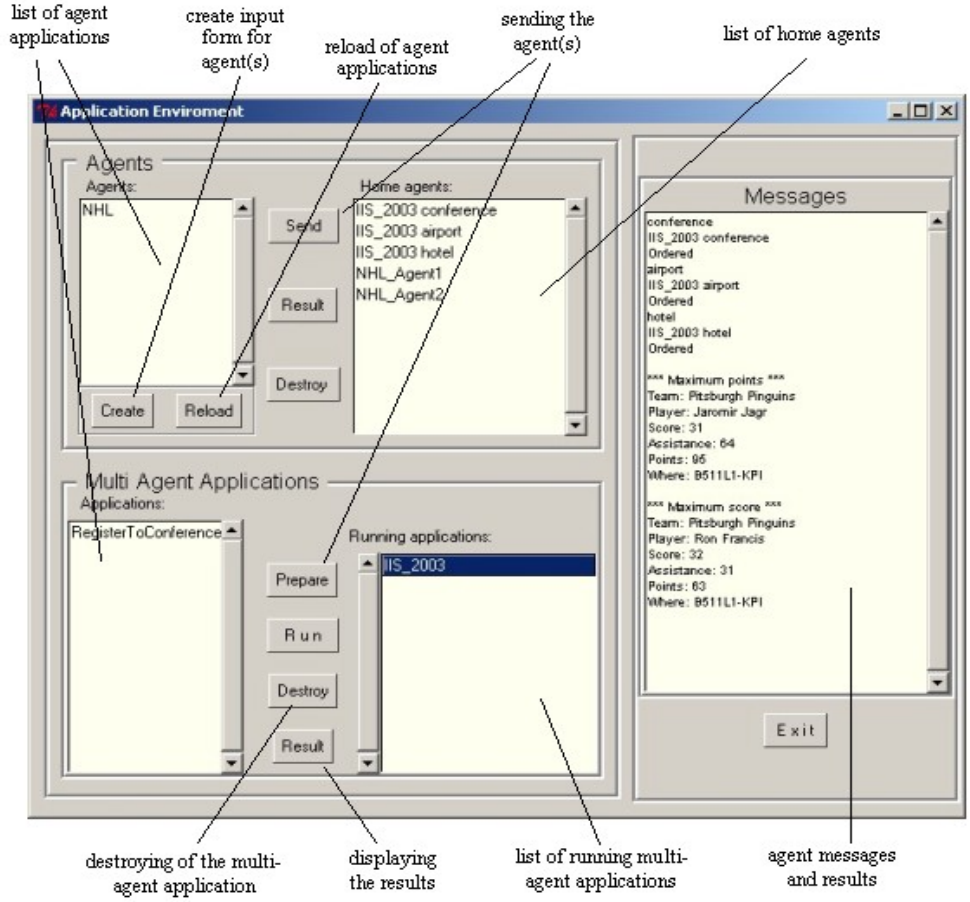

Figure 1: Basic environment for the agent-based applications

To make the usage of agent-based applications easier, a simple graphical interface was designed and implemented (see Figure 1). It supports third step from the previous methodology and enables the running of agent-based applications to users without knowledge of the implementation programming language. The detailed characteristics of the design and implementation process of mobile agent based applications can be found in [11], where an example using mobile agents for information retrieval about players of NHL regarding the chosen criterion is described.

But not only applications using one type of mobile agent can be supported. The multi-agent (MA) applications, which use more than one type of agent class, could also be created. The MA application is identified by a list of subagents, which are able to communicate and coordinate their activities (if needed). First the agent programmer prepares the needed agent classes according to steps 1 and 2. Then the graphical user interface offers the possibility to create an instance of a multi-agent application. A special form has to be prepared, whereby not only the itinerary (i.e. list of locations together with tasks to be done on each location) is specified, but also the set of subagents.

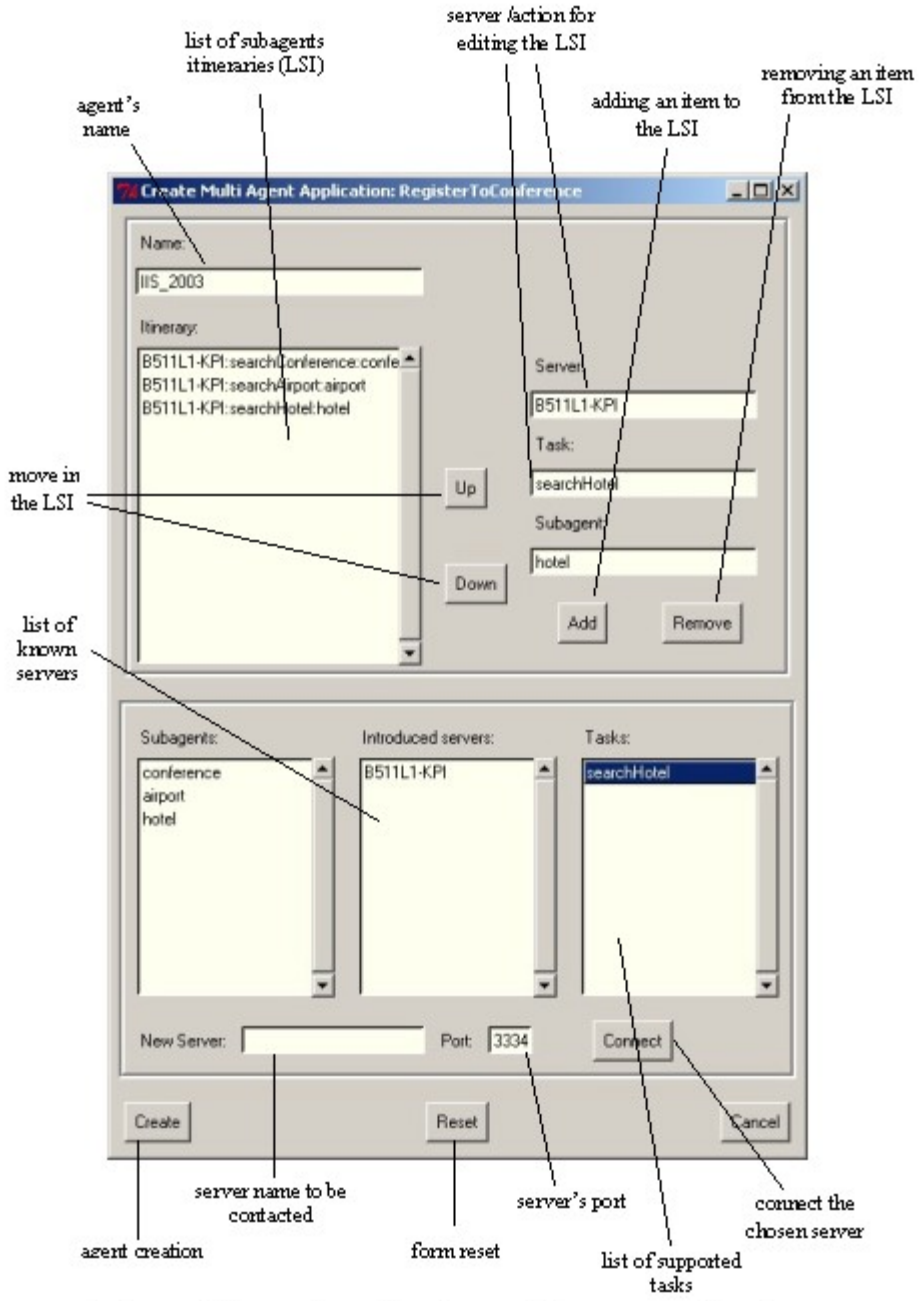

Figure 2: Input form for the multi-agent applications

The necessary information for running a MA application is required according to a control file - *DbTasks.src* – that includes the list of all subagents and their tasks. Using this information together with information from the databases of known MAE servers, the appropriate fields in the multi-agent form (see Figure 2) are filled. After selection of an agent,

the list of its possible tasks is automatically refilled by tasks offered by this agent. On click of the *create* button, the "list of subagents itineraries" (LSI) is used to create all needed instances of mobile agents with appropriate itinerary. All agents are put into the list of home agents on the running MAE server. At this point exists the multi-agent application as instance and all subagents as home agents in created state, whereby every subagent stores the references of all other subagents in a special list called *AgentBook*. On click of the *run* button, the procedure *GoAgent* is called on all agents that belong to the same multi-agent application. Similarly all mentioned agents are destroyed by clicking of the *destroy* button. After return of the agent, the (partial) results could be presented to the user after click on the *result* button.

4. RESERVATION SYSTEM EXAMPLE

The proposed structure for support of the MA applications was verified on a simple MA application of conference reservation system *RegisterToConference*. This application enables registration to a conference (subagent *conference*) together with reservation of an accommodation (subagent *hotel*) and air transport to the place of the conference (subagent *airport*). Supported tasks for all agents are search tasks for finding the information they are looking for and registration/ordering in case of success. The information about requested conference is specified in the *envirc* file that connects conference agent to the local resource. In similar way the local databases of flight connections and accommodation possibilities are searched, whereby the information regarding destination and time schedule is available from the conference agent. Agents move between locations from the itinerary up to some location, where the requested flight/accommodation could be found. If agent fails to find the requested service in every location in the itinerary, sends a failure message to other running agents of the *RegisterToConference* application, the registration process is aborted and all agents return to the home location. If the agent was successful, sends a message to other agents, waits for the confirm messages and success messages from other agents. Only if all agents are successful, the registration/ordering process is continued and agents return home with success.

5. CONCLUSIONS AND FUTURE WORK

In this paper we presented a toolkit for support of the design, implementation and execution of multi-agent applications by means of the ESMA system, which was also briefly introduced. Mobile agent based applications offer high flexibility in distributed environment. More sophisticated applications include more than one type of mobile agents and the support of running such applications requires management of all instances of all subagents involved in the multi-agent application. Presented graphical user interface helps after implementation of the multi-agent applications in the execution phase. So users, who do not have knowledge regarding agent toolkit used for design and implementation, can run the multi-agent application under different circumstances.

In the present we are working on an implementation of presented framework by means of standardized FIPA compliant agent framework JADE [5] that also offers direct support of mobile devices like PDAs and mobile phones. We want to apply it in the area of knowledge management support for e-government [10]. Parallel to that we are also working on a natural method for modelling such applications, whereby a tool for formal description of distributed systems based on agent technology and technology of mobile code is our goal [14].

In our future work we want to concentrate on design of a set of basic multi-agent schemes, which could support also the phase 2 in presented methodology and so overcome

a gap between programmers without skills in design and implementation of the multi-agent applications and those with these skills. For this purpose we plan to use configurable IPSE (Integrated Project Support Environment) InCASE [4] developed at DCI FEI in Košice.

REFERENCES

[1] DeLoach, S.A.; Wood, M.F.; Sparkman, C.H. (2001): *Multiagent System Engineering*, The International Journal of Software Engineering and Knowledge Engineering, Vol. 11(3)

[2] Franklin, S.; Graesser, A. (1996): *Is It Agent, or Just a Program? : A Taxonomy for Autonomous Agents*, Proceedings of Agent Theories, Architectures, and Languages (ATAL 96), Budapest

[3] Fuggetta A.; Picco, G.P.; Vigna, G. (1998): *Understanding Code Mobility*, IEEE Transactions on Software Engineering, Vol. 24(5)

[4] Chladný, V., Havlice, Z., Szaniszlo, P. (2000): *Architecture of the InCASE*, Proceedings of the International Scientific Conference Electronic Computers and Informatics 2000 (ECI 2000), Košice - Herľany

[5] *JADE - Java Agent DEvelopment Framework*, Telecom Lab Italia, The Agent and Object Technology Lab (University of Parma), http://jade.cselt.it (2003)

[6] Jennings, N.R.; Sycara, K.; Wooldridge, M. (1998): *A Roadmap of Agent Research and Development*, Journal of Autonomous Agents and Multi-Agent Systems, Vol. 1 (1)

[7] Klein, M. (1999): *Exception Handling in Agent Systems*, Proceedings of the Third International Conference on Autonomous Systems, Seattle, Washington

[8] Nwana, H.S. (1996): *Software Agents: An Overview*, The Knowledge Engineering Review, Vol. 11 (3)

[9] Ouksel, A.; Sheth, A. (1999): *Semantic Interoperability in Global Information Systems*, Special Issue of ACM SIGMOD Record, Vol. 28(1)

[10] Paralič, J.; Sabol T.; Mach, M. (2003): *Knowledge Enhanced E-government Portal*, Proceedings of the 4th IFIP International Working Conference Knowledge Management in Electronic Government (KMGov 2003), Rhodes

[11] Paralič, M. (2002): *Contribution to the mobile agent based programming*, PhD thesis, Technical University of Košice

[12] Posland, S; Charlton, P. (2001): *Standardizing Agent Interoperability: The FIPA Approach,* Proceedings of The Advanced Course on Artificial Intelligence (ACAI 2001), Prague

[13] Roy, P.V.; Haridi, S.; Brand, P.; Smolka, G.; Mehl, M.; Scheidhauer, R. (1997): *Mobile Objects in Distributed Oz*, ACM Transactions on Programming Languages and Systems, Vol. 19(5)

[14] Tomášek, M. (2002): *Model of Communication and Mobility in Distributed Systems*, Proceedings of the 5th International Scientific Conference Electronic Computers and Informatics 2002 (ECI 2002), Košice - Herľany

[15] *The Mozart Programming System*, Deutsches Forschungszentrum für Künstliche Intelligenz GmbH, Universität des Saarlandes. Swedish Institute of Computer Science, Université catholique de Louvain, http://www.mozart-oz.org (2003)

[16] Wooldridge, M.; Jennings, N.R.; Kinny, D. (2000): *The Gaia Methodology for Agent-Oriented Analysis and Design*, Journal of Autonomous Agents and Multi-Agent Systems, Vol. 3(3)

Received: 17 December 2003
Accepted: 3 July 2004