SYSTEM for MOBILE AGENTS

Authors:
Alessandro Genco - Professor of Operating Systems in University of Palermo Italy - Corresponding author of the paper
Phone: 0329 6147704
Salvatore Pennacchio - Computer Engineer and Assistant Professor in University of Palermo Italy
Phone: 091 6825702
Rosario Sorbello - Computer Engineer and Assistant Professor in University of Palermo Italy
Phone: 333 3525677

Address:
Università di Palermo
Viale delle Scienze
90129 Palermo
ITALY

Abstract

Mobile agents are in this period at the centre of numerous plans of scientific research in the field of Computer science Engineering. IBM research centre of Tokyo, for example, has dedicated many years to the plan of a framework for mobile agents called IBM Aglets. The present project has an objective: to contribute to the affirmation and to the development of mobile agents in a generalized manner and to concur to the visualization and to the control of the agents in the World Wide Web to make them usable for every user. Our system, that we have called MAWI (Mobile Agents with Web Interface), is constituted of:

1) a Java package for the achievement of a client-server system that emulates the strong mobility of the code;
2) a language (compiler) for the achievement of a Web interface for the monitoring and the control of every mobile agent (this language replaces the Tahiti software of IBM Aglets, with the difference that it is possible to create one different interface for every agent);
3) a database for mobile agents for the management of the information (for customer and agents);
4) a system of asynchronous and synchronous communication between agents. (Communication is very important for e-commerce with mobile agents).

When the customer connects itself with the Web page (created using our language for agents) of the server Apache, can insert data inputs and dispatch the agent. The agent is in the server of mobile agents. The data of input will be sent to the file PHP in the server Web. This program in PHP will create a file text with the data inserted from the customer. This file text will go to the server of mobile agents. The agent now can move itself with the instructions of the users. When the agent is at the end of its job it will send the results to the database. When the agent move itself, it sends changed data (IP address, name of computer…) to the database.

The communication between agents can be synchronous or asynchronous. I have taken the cue from the best programs for chat, that are ICQ and IRC. The concept is the same but the difference is that communication is not between persons but between software agents. When an agent wants to communicate with another, it asks database IP address of the wished agent and it will be banally connected with it. ICQ works in this way (synchronous) when the customer is online, it works in an asynchronous way when the customer is offline. If an agent wants to establish an asynchronous communication it sends a message to the database.
This message will be put in the table of database and the agent will read it when it will want it.

The management of strong mobility, as we have already said, is linked to the same concept of mobile agent. We have created a Java package to manage in an efficient way a connection of the client-server that emulates complete mobility of the code with the state. To make this, we need to serialize the relative data of the agent and the relative class (in this case it represents the same agent). Serializing means: making a packet of all the data in sequence. At the end the data will be de-serialized and the agent will be back in life but in a new computer. The Java language help us to make these operations in an elegant way taking advantage of the heredity of some classes. The class loader, for example, is very important.

Interface customer system MAWI will be create using a compiler (we have called it Interscape) to make Web interfaces for mobile agents. The programmer will have to create a web page (using this language) for the users to make mobile agents very easy to use for everyone. The language is composed from a compiler created with the Lex, the Yacc and the language C++.

The syntax analysis is carried out through the LEX and the generator of parser YACC. The program source, written in the Interscape language, will be translated in the HTML language. The phase of translation is activated with the execution of the program. The program has been written in language C. This program is connected to the LEX and the YACC through the files lexyy.c (given back from the LEX) and sas14 ta.c (given back from the YACC) and moreover it recalls the functions yyparse() and yyerror().

The database has the scope to manage the information of the agents, to save the results and to give all information for communication between agents. The database is of much profit for the communication (synchronous and asynchronous) between agents.

Interscape was born as an alternative to HTML language to create Web pages, but it is very easy. In this version of the language we have changed something to make it better for mobile agents. We have particularly taken care of connection and the data input from the customer to the database mySql. The programmer of mobile agents have to create a Web interface (for the customer) with the following functions:
1) monitoring of mobile agent
2) control of mobile agent
3) a web page to connect to the database mySql with possibility to change data
4) recording new agents in the circuit

Every framework for mobile agents has a software for the monitoring and the control of the agents, for example: system IBM Aglets has created a software called Tahiti that carries out this task. However no system offers the possibility to create a program of control for every mobile agent with different degrees of freedom in function of the type of agent.

Only our system MAWI can do it! If an agent must carry out a complex search, the Web interface must contain varies fields input because the customer can insert the data to search. If there are no data of the customer, the Web interface will be different and not complex.
Example

Our job consists of a simulation of a control system in Java language for the famous robot Nomad 150. We have used this control system for the Teambots software for mobile robots created by Georgia Tech College of Computing.

The Teambots software consists of two parts: one is called TBSIM simulation application and it has to make a simulation to try the control system for the mobile robots; and the other one is called TBHARD robot execution environment and it has to make the real software to run on the real robot Nomad 150.

The good idea is, of course, to make a simulation first and to control its results, and if everything is fine it is possible to use the same software in the real robot Nomad 150. Our software is a new instrument, for robots to find an object in a room. We just use a simulation first: then we design a room, put the linear attractions and obstacles and make a control system program in Java language.

To create our control system in Java we extend a class of the original Teambots package.

First of all, to design the room, we create a file .dsc. This file has got the information about robot position, obstacles and attractions. The room consists of some different areas. When the robot goes from one area to another one the problem goes from state to another one. When the robot sees the object that it is looking for, the problem is solved and that area is the final area. If one room has been visited one time the robot doesn't go there any other time.

The motor schemas are only four: north south east west. We also use a noise vector to make a simulation near the reality. The motor schemas are just vectors and what the program does is just a sum of these vectors. We can regulate the noise vector as we want and as we need. Of course this noise vector is a particular instrument of the simulation.

During the simulation is possible to see: the instant time and the current position of the robot.

It is interesting to consider that it is possible to put more than one robot in the room and the robots can communicate between them.

By the way our problem is this:

States: every area of the room;
Operators: the robot can go to the north south east west;
Test: the program verifies if the robot finds
the right object;
Costs: the distance that the robot has done.

The most important thing of the simulation is that it is possible to consider different research strategy and later to use the right one in the real robot.
In this example we just consider one strategy research.

Another one new is that we use a mobile agent system for a co-operative robot system. When a robot finds the object it can move one research program with its history recorded and with its strategy to another robot. Our mobile agent system is a framework in Java.

Bibliografia

VSB, University of Frankfurt, Germany.
Asynchronous Message Transfer Agent System (Ametas).
http://www.vsb.cs.uni-frankfurt.de/ametas.
F.M. de Assis Silva and R. Popescu-Zeletin.


