BULETINUL Universităţii Petrol – Gaze din Ploieşti	Vol. LX No. 1/2008	63 - 72	Seria Matematică - Informatică - Fizică

AgentAirPol System, an Agent-based System for Air Pollution Analysis

Elia Petre

Universitatea Petrol-Gaze din Ploieşti, Bd. Bucureşti 39, Ploieşti, Catedra de Informatică
e-mail: elia_petre@yahoo.com

Abstract

This article describes the basic steps which may be followed in building an intelligent agents-based system, from its analysis and design to the implementation and running system. These steps are followed out to create such a system using Zeus, a specific software. The system, called AgentAirPol, is targeted at the environment protection domain, especially to the industrial activity and its effects on the athmosphere. It has been designed for a careful tracing of the air pollution rate produced by an industrial plant.

Key words: *intelligent agent, air pollution, task agent, utility agent, ontology*

Introduction

One of the most important and interesting domains developed during the last years is *Computer Technology and Artificial Intelligence*. In the last decade, a certain area of this domain has known an explosive evolution - Intelligent Agents. Many scientists agree that: "Agent-based systems are one of the most vibrant and important areas of research and development to have emerged in information technology in the 1990s" [6].

The basic idea of an agent-based system is the following: problems that are difficult to be solved with centralized, traditonal methods, can be simply solved by some software entities in a helpful environment, based on cooperation and communication [8]. Therefore, the characteristics of an intelligent agent are:

- autonomy - the agent has full control of his internal actions without an external input;
- reactivity - the agent reacts as a result of the environment input;
- proactivity - the agent can initiate an action when it considers it is needed to achive a goal;
- social ability - it can communicate with the other agents from the system, in order to finish his task or to help the others.

In this paper, we systematically create an intelligent agent-based system, using Zeus, one of the specialized software. AgentAirPol is our system's name and it tries to ascertain if the air releases from an industrial plant are environment pollutant or not.

The AgentAirPol System Description

Before deciding which software is the most appropriate for our system, we have to establish the domain where we will use it, the needed agents and their tasks. Because nowadays, the environment pollution is one of the most important problems of the humankind, this system has been designed for a careful tracing of the air pollution rate produced by an industrial plant (Fig.1). It contains four agents, each one with a specific task:

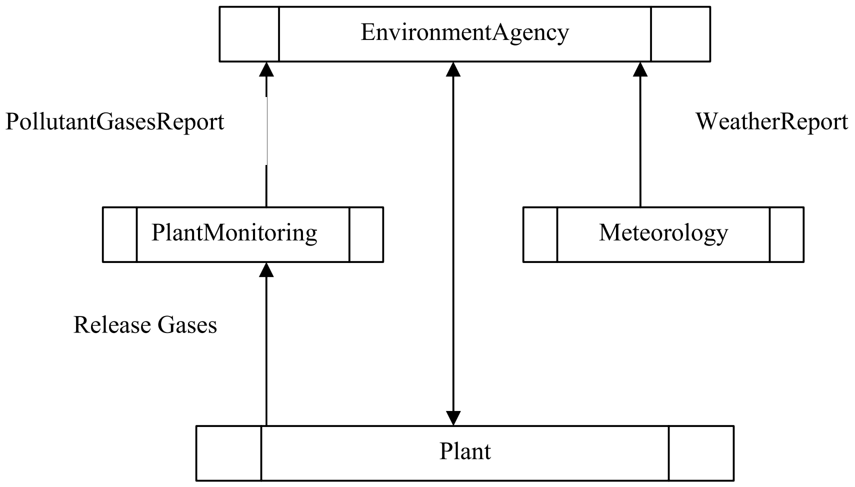


Fig 1. The information flow diagram in the AgentAirPol System

The agents included in our system are the following:

- First is *Fabrica* (*Plant*), an entity which has a production line. We are not interested in the products of this plant, but in the polutant gases released in the atmosphere, as a result of the industrial process. So the only task of this agent is *EliminareNoxe* (Release Gases);
- The second agent in our system is *Monitorizare* (*PlantMonitoring*), which has to measure the amount of gases released by the plant and to report if it is over the legal limits. We named the task *ÎntocmireRaportDateÎnregistrate* (*PollutantGasesReport*);
- To have a complete report regarding the plant pollution, we must take into consideration the wheather conditions. Why? For exemple, if the wind direction is from another pollution source, the collected data may not refer only to the plant emissions [7]. For this task, *ÎntocmireRaportStareaVremii* (*WeatherReport*), we have shaped the agent *Meteorologie* (*Meteorology*);
- The fourth agent is *Garda_Mediu* (*EnvironmentAgency*), the agent who brings together the two reports and finds if in that area is some pollution activity. His task is *AnalizaSituatie* (*PollutionReport*).

We have to mention that the measurements and their interpretations are accomplished by request from the agent *Garda_Mediu*, and all the used tools are omologated. The gases closely traced are SO_2 (sulfur dioxine), SO_3 (sulfur trioxine) and CO_2 (carbon dioxine).

Agent Interaction

The manner in which our agents cooperate is something more than a simple message exchange; this process must follow some schemes and rules, some methods and techiques gathered under the name *Interaction Protocol*. This allows defining a sequence of messages between the agents as well as a methodology for the agents to react to those messages.

For the system built for air pollution monitoring it is not necessary to create some negotiation protocols, like we have in the electronic commerce, because our agent's interaction is based on cooperation rather than competing [1].

Knowledge Modelling

The next step in our project is to define the system ontology - meaning to set up the words, the rules, the restrictions, the concepts inherent to the application, as well as their attributes and possible values. These are on one hand related to the gas emissions, their concentrations and analysis and on the other hand referring to the weather conditions.

Creating the Application

Until now, all we have designed, related to different shapes, responsabilities, roles of the agents, to their interactions and the concepts used, was just theoretical.

From practical perspective, we will build this system in one of the most popular software targeted at intelligent agents, Zeus. The Intelligent Systems Reserch Group from British Telecomunication created it in 1999 and since then, many important agent-based systems have been built using Zeus.

The creation process combines the steps necessary to create a generic ZEUS agent with the steps necessary to implement the role-specific solutions identified during the previous phase [3].

As described in the ZEUS Application Realisation Guide, "the purpose of these stages is to translate the design we have derived from the role models into agent descriptions that can be automatically created by the ZEUS Agent Generator tool" [3].

Ontology Creation

This stage involves the use of the Ontology Editor tool. Once opened, we will start defining the concepts necessary for our application: click on ZeusFact- Entity- the button "Add New Child Fact" and create *SO2*. In the same way, we form the other concepts: *SO3, CO2, limitaSO2* (*SO2limit*), *limitaSO3* (*SO3limit*), *limitaCO2* (*CO2limit*), *Vant* (*Wind*), *Precipitatii* (*Precipitation*), *TemperaturaAerului* (*AirTemperature*), *RaportFinal* (*FinalReport*), *RaportAnalizaDate* (*DataAnalisysReport*) and *RaportStareaVremii* (*WeatherConditionReport*) (Fig. 2).

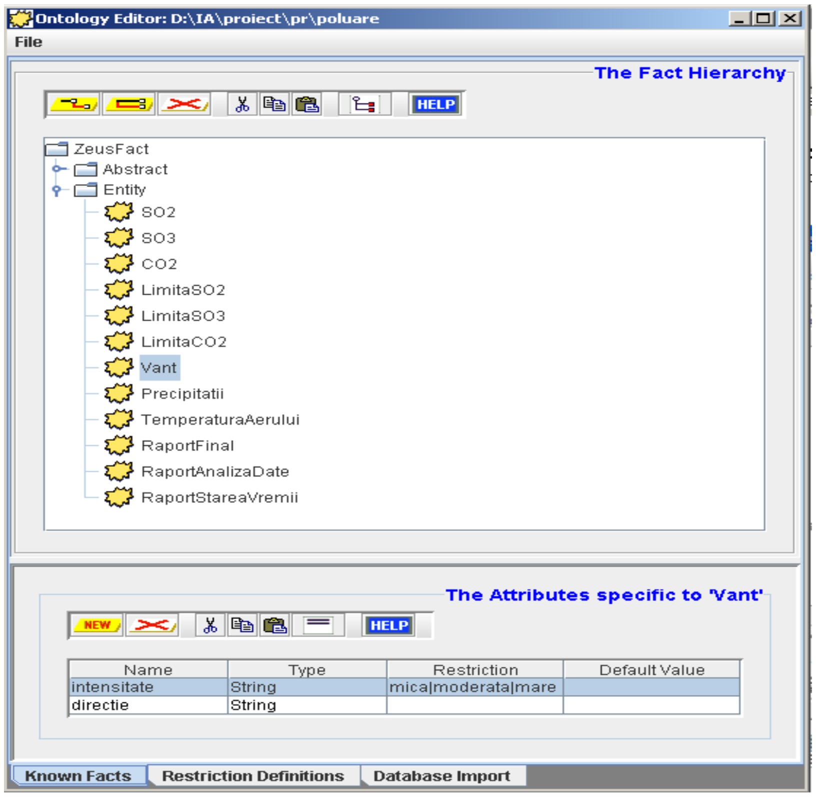
Their attributes and types, as well as their restrictions, are generated after clicking on the "Add New Attribute" button on the lower panel [5].

With the ontology created, we can leave the Ontology Editor and begin the agent creation process.

Agent Creation

The agent creation stage consists of several sub-processes, iterated for each different task agent in the application. To start, then we have to create an agent: click on the "New Agent" button from the Zeus Generator Panel and rename it *Garda_Mediu*.

Next, we have to define it: which are its task identification, its initial resources, its acquaintances, its communication protocols.



ZeusFact
- Abstract
- Entity
  - SO2
  - SO3
  - CO2
  - LimitaSO2
  - LimitaSO3
  - LimitaCO2
  - Vant
  - Precipitatii
  - TemperaturaAerului
  - RaportFinal
  - RaportAnalizaDate
  - RaportStareaVremii

Name	Type	Restriction	Default Value
intensitate	String	mica|moderata|mare	
directie	String		

Fig 2. The Ontology Editor for AgentAirPol System

In the Agent Editor Window, we have to set:

- Planning Parameters;
- Task Identification;
- Initial Agent Resources.

In the Planning Parameters fields, we have to decide the maximum number of simultaneous tasks, as well as the planner length. Here it is important to mention that in Zeus the duration of all agent activities is evaluated in time-grains rather than seconds or minutes. “The granularity refers to the smallest possible indivisible period of time for the application: a period known the time-grain. The time-grain determines the period between ‘ticks’ of the application-wide clock that is maintained by the Agent Name Servers” [3]. For the *Garda_Mediu* agent we set the first textfield to 2 and the second to 20.

Having in mind that the task of this agent is *AnalizaSituatie*, meaning to collect the two reports from other agents and decide if the plant is a pollutant factor for the environement or not, it has not initial resources (Fig. 3).

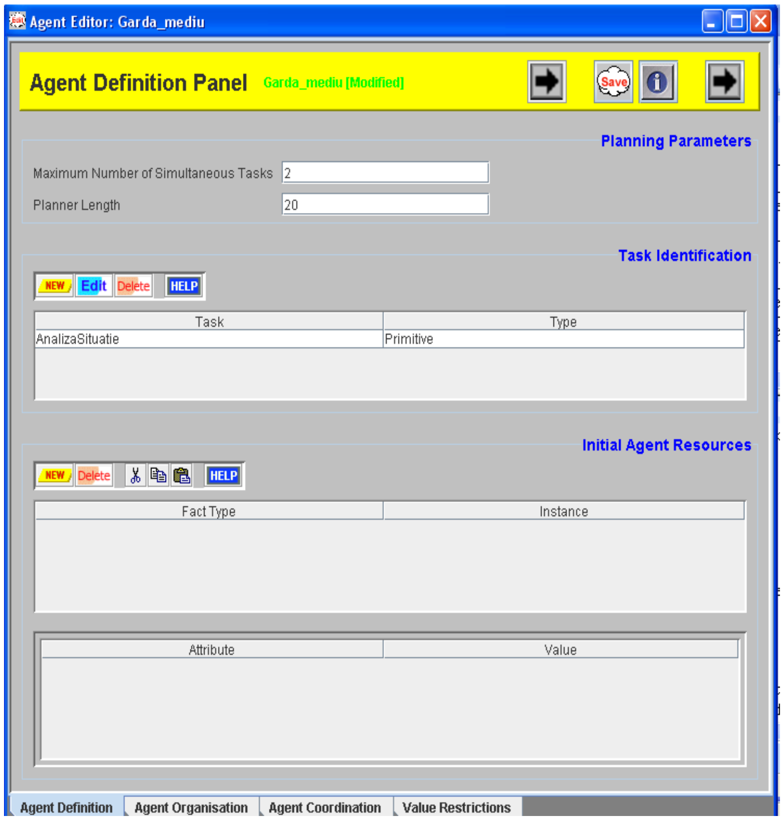
Agent Organization

This stage has to settle on if this agent has acquaintances, if agent has prior knowledge of other agents, especially if it interacts with them on a regular basis. For *Garda_Mediu* agent we complete this panel with *Meteorologie*, as a subordinate acquaintance by clicking “Create new Acquaintance” button from Agent Organization Panel.

Agent Coordonation

Here the focus is on the coordination protocols - whether the agent is an Initiator, Fipa-Contract-Net-Manager Protocol, whether the agent is Respondent, Fipa-Contract-Net-Contractor

Protocol. Because there is no negotiation between agents, there is no need to implement negotiation strategies [2].



Planning Parameters

Maximum Number of Simultaneous Tasks 2

Planner Length 20

Task Identification

Task	Type
AnalizaSituatie	Primitive

Initial Agent Resources

Fact Type	Instance

Attribute	Value

Fig 3. The Agent Definition Panel for *Garda_Mediu* agent from AgentAirPol System

Task Definition

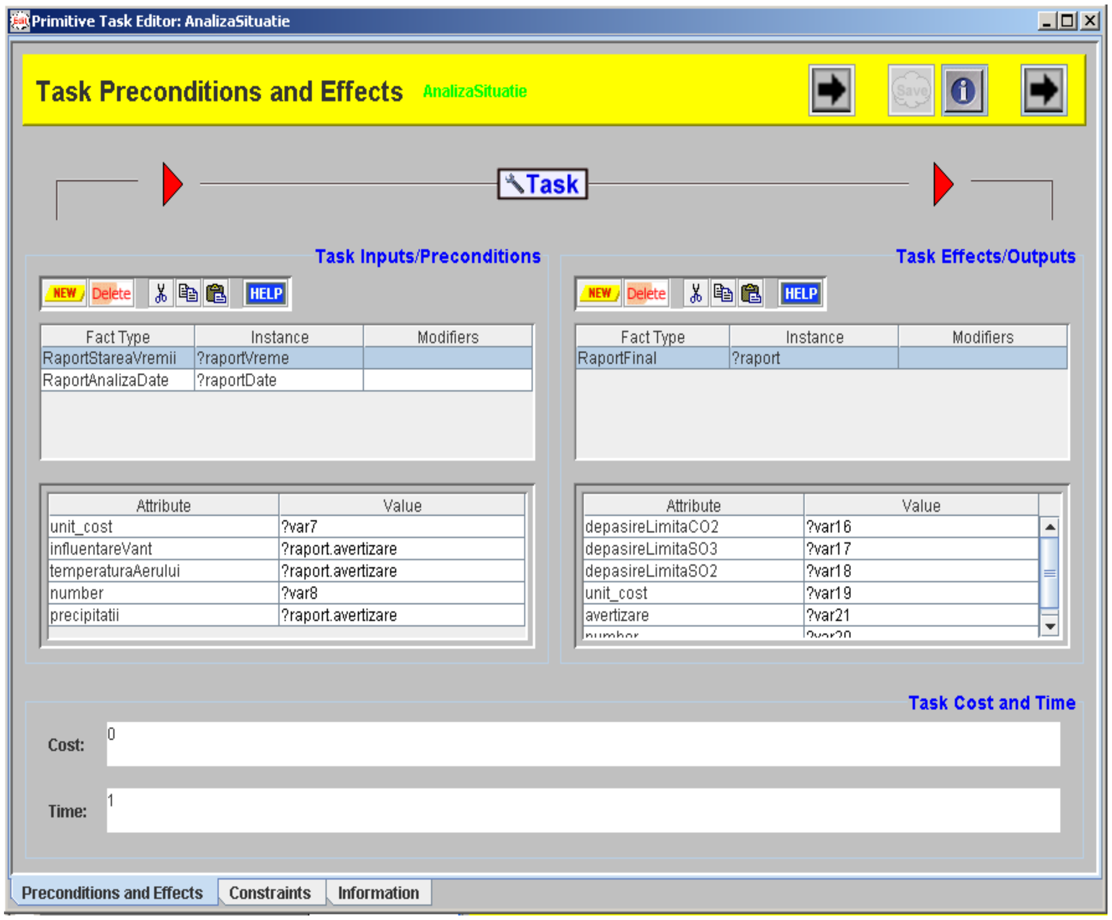
So far, the largest part of the *Garda_Mediu* agent has been set. The only thing necessary for the agent to be complete is its task definition. It has been already established in the application design stage that *Garda_Mediu* has to have as input two reports - from *Meteorologie* and *Monitorizare* - and to merge them into a final report about the plant pollutant activity. In Zeus, the inputs are called preconditions and the task's output effect. The task of the agent *AnalizaSituatie* can also have restrictions.

To define this task we need to open the Primitive Task Editor Window. To set up the task's preconditions, we push the "New" button from the Task Inputs/Preconditions panel and select from the ontology concepts the two ones suitable: *raportDate*, a *RaportAnalizaDate* variable and *raportVreme*, a *RaportStareaVremii* variable.

Next, we enter the constraints relevant to *raportDate* consumption, whose values *depasireLimitaSO2*, *depasireLimitaSO3* and *depasireLimitaCO2*, specified for the final report, *raportFinal*, are influenced by the values of the attributes with the same names from the *raportDate*.

After that, the constraints have to be entered; they are relevant to report *Vreme* consumption, namely that the *avertizare* attribute from the task effect depends on the fields *influentareVânt*, *temperaturaAerului* and *precipitatii*.

Now we have to set up that the effect of this task is raport, a *RaportFinal* variable, by pressing the “New” button from the Tasks Effects/Outputs Panel and we have finished this task’s definition (Fig. 4).


Primitive Task Editor: AnalizaSituatie
Task Preconditions and Effects
AnalizaSituatie
Save
Task
NEW
Delete
HELP
Preconditions and Effects
Constraints
Information

Task Inputs/Preconditions

Fact Type	Instance	Modifiers
RaportStareaVremii	?raportVreme	
RaportAnalizaDate	?raportDate	

Attribute	Value
unit_cost	?var7
influentareVant	?raport.avertizare
temperaturaAerului	?raport.avertizare
number	?var8
precipitatii	?raport.avertizare

Task Effects/Outputs

Fact Type	Instance	Modifiers
RaportFinal	?raport	

Attribute	Value
depasireLimitaCO2	?var16
depasireLimitaSO3	?var17
depasireLimitaSO2	?var18
unit_cost	?var19
avertizare	?var21
number	?var20

Task Cost and Time

Cost: 0

Time: 1

Fig 4. The Task Definition Panel for *AnalizaSituatie* task from AgentAirPol System

The same process that built the *Garda_Mediu* agent can be used to create the *Fabrica*, *Monitorizare* and *Meteorologie* agents. Looking back at the design derived from the agent roles, we can only see small differences between the agents’ definition as well as their tasks.

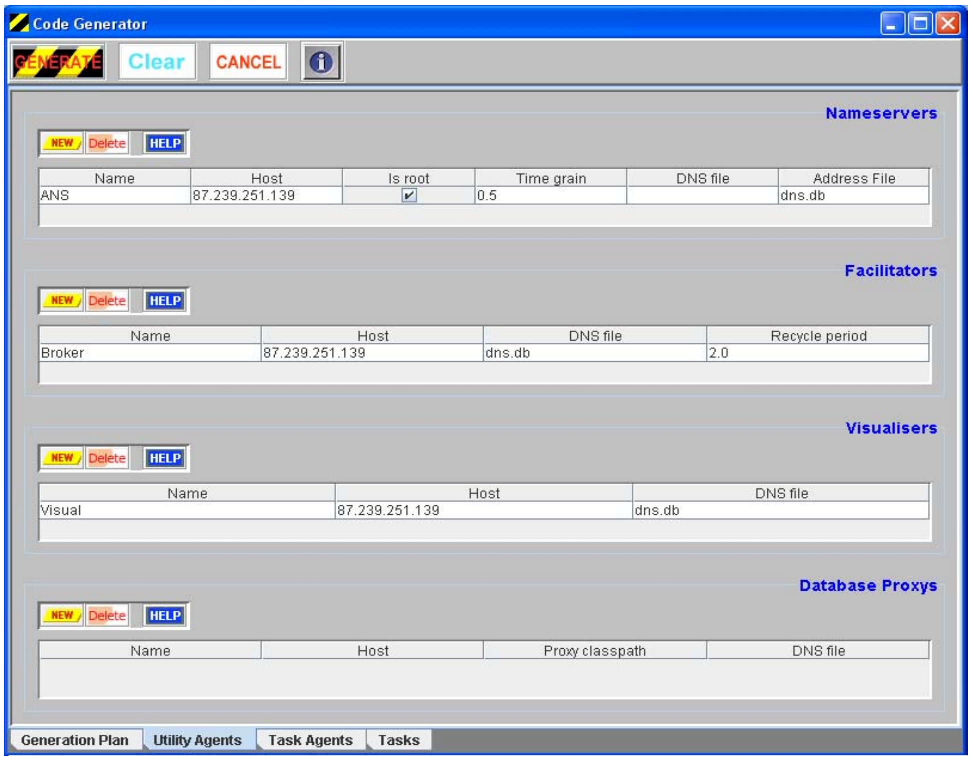
Utility Agent Configuration

During this phase, we can decide the utility agents to create and configure them accordingly. The Address Name Server (ANS) and Facilitator are essential to the application because they know the addresses and the caracteristics of each agent in the system, the Visualiser is optional but always useful, offering a graphical interface for the user [4].

Their configuration can be done from the Utility Agents tab pane of the Code Generator tool (Fig. 5).

Task Agent Configuration

In the Task Agents tab panel, we mention that *Garda_Mediu* agent has attached an external program, *Analiza,* the application that creates the AgentAirPol user’s interface. More precisely, it generates the interface’s elements that are dedicated to the user input, the database information display, used by *Meteorologie* agent to build the weather conditions report, and of course for the analysis final report.



Nameservers

Name	Host	Is root	Time grain	DNS file	Address File
ANS	87.239.251.139	✓	0.5		dns.db

Facilitators

Name	Host	DNS file	Recycle period
Broker	87.239.251.139	dns.db	2.0

Visualisers

Name	Host	DNS file
Visual	87.239.251.139	dns.db

Database Proxys

Name	Host	Proxy classpath	DNS file

Fig 5. The Utility Agents tab Panel for AgentAirPol System

Code Generation

In this stage, we will create in a user specified directory the java files of our application, by pressing the "Generate" button. This way, every agent and task will have a java file generated with his name. Beside this, there are three special files: *run1.bat*, *run2.bat* and *run3.bat* used in the running application stage. The programmers that are acquainted with Java platform will know that it is required to be generated the class files, too. We can do that by using the command line and the *cd* command to locate our application directory and the *javac *.java* command to create the class files.

Having all these files, we are now ready to run the application.

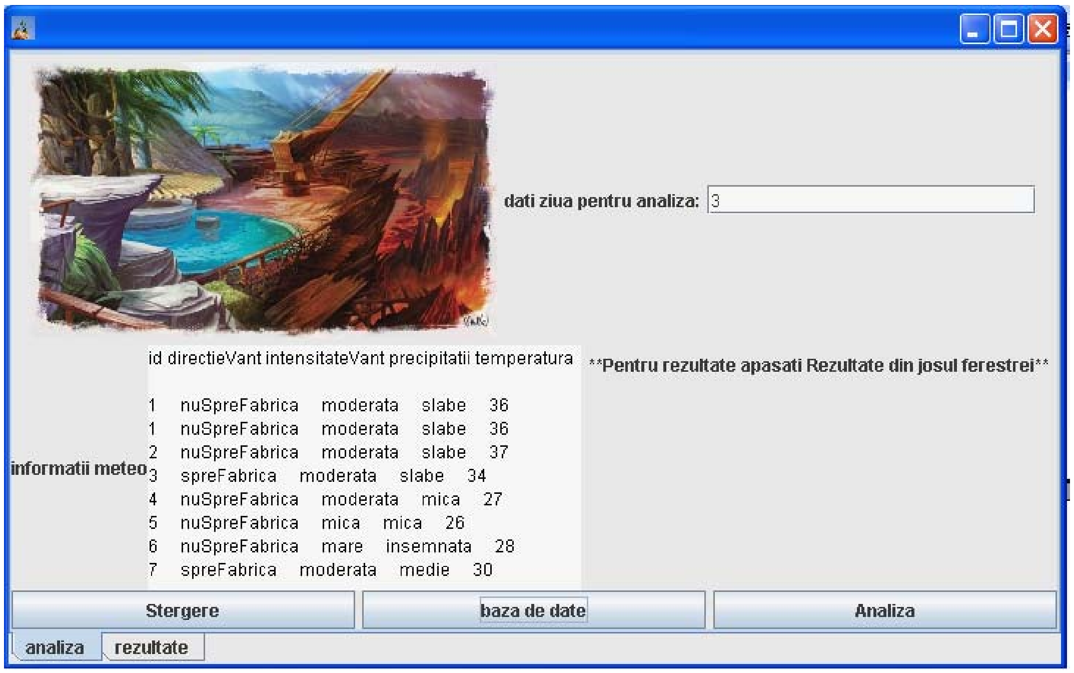
Running the application

Launching the application will involve the following process:

- execute *run1.bat* script and start the Agent Name Server;
- execute *run2.bat* script to start the 4 agents and the external program, *Analiza*;
- execute *run3.bat* to start the Visualiser and Facilitator agents.

Once launched, the agents will register themselves with the Name Server and reply to Facilitator and Visualiser.

The user interface has two parts: on the first tab panel, the user can decide on the day to analize and, after select the *Baza_de_Date* button, on the second tab panel the user can verify the information used by the *Meteorologie* agent in his report (Fig. 6).



dati ziua pentru analiza:
3
Pentru rezultate apasati Rezultate din josul ferestrei
informatii meteo
Stergere
baza de date
Analiza
analiza
rezultate

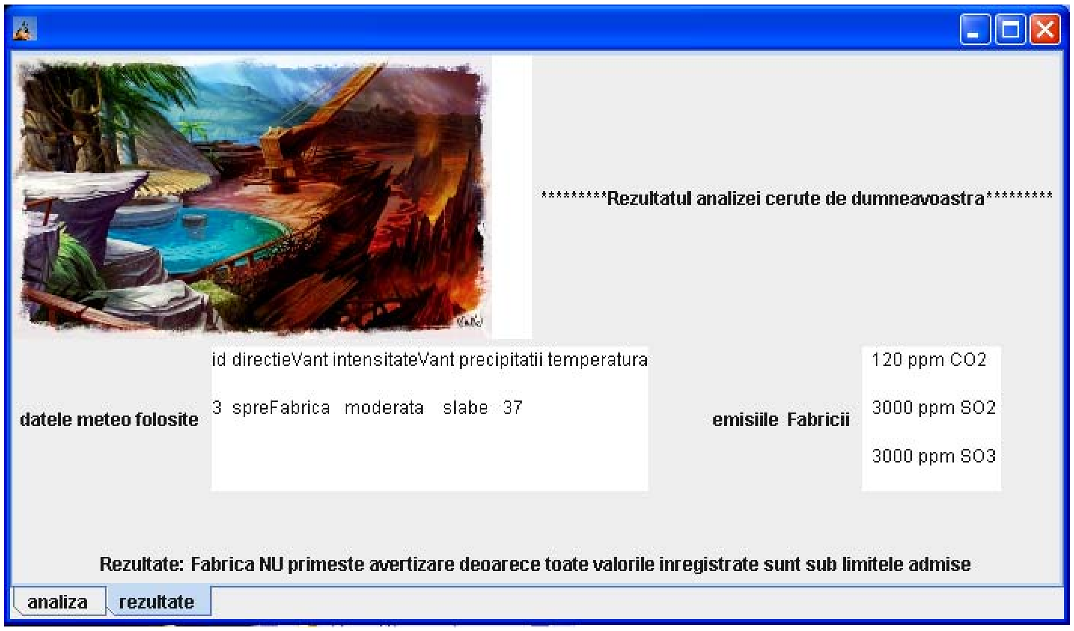
```
id directieVant intensitateVant precipitatii temperatura

1   nuSpreFabrica   moderata   slabe   36
1   nuSpreFabrica   moderata   slabe   36
2   nuSpreFabrica   moderata   slabe   37
3   spreFabrica   moderata   slabe   34
4   nuSpreFabrica   moderata   mica   27
5   nuSpreFabrica   mica   mica   26
6   nuSpreFabrica   mare   insemnata   28
7   spreFabrica   moderata   medie   30
```

Fig 6. The AgentAirPol System's User Interface

The button *Stergere* is designed to clear the components on the first tab panel.

The *Analiza* button issues a goal to the *Garda_Mediu* agent, through the external program, *Analiza*. The results can be seen in the second tab panel: the concise information for the day picked by the user, but also the input values for the *Monitorizare* report: the registred data from the plant, as well as the final report from the *Garda_Mediu* (Fig. 7).



*********Rezultatul analizei cerute de dumneavoastra*********
datele meteo folosite
emisiile Fabricii
analiza
rezultate

```
id directieVant intensitateVant precipitatii temperatura

3  spreFabrica   moderata   slabe   37
```

```
120 ppm CO2

3000 ppm SO2

3000 ppm SO3
```

Rezultate: Fabrica NU primeste avertizare deoarece toate valorile inregistrate sunt sub limitele admise

Fig 7. The AgentAirPol System's User Interface

Conclusion

AgentAirPol is an efficient and useful solution for the air pollution domain, because it can ascertain, by some independent measurements, for the local authorities, if there is an unusual pollution activity with direct results on the environment.

We gradually described all the steps needed for put into practice this application: its analysis and design, by establishing the agents and their tasks, their ontology and their coodonation protocols. As well, we put together the system by creating the user-friendly interface. This offers the possibility to the user to select a precise day for analizing (knowing) the pollution rate. Additionally, for a better understanding of the process, there are listed the meteorological database information, as well as the pollutant gases values.

We consider that for our first agents-based application implemented in Zeus, all we have achieved until now is a great start in our research related to intelligent agents.

Of course, this application is not covering all the aspects that can be implemented in an intelligent agent-based system. A lot more can be done, like other communication stategies implementation, aside those distributed with the Zeus package, particular to certain application.

Another aspect looked-for to be improved is the database connection. We may also want to try to populate a table from a database with facts from the agents, not only to acquire information from a relation.

Furthermore, we can expand the system by increasing the number of the participant agents and using a distributed agent-based system. That means, not only a computer can host our agents, but they can be dispersed in a computer network.

References

1. Barbuceanu, M. - Fox M.S. COOL: a language for describing coordination in multi-agent systems, *Proceedings of the International Conference on Multiagent Systems* (*ICMAS*) 1995, pp. 17-24, 1995
2. Collins, J., Ndumu, D. - *Zeus Technical Manual*, The Zeus Agent Building Toolkit, Intelligent Systems Research Group, BT Labs, September 1999
3. Collins, J., Ndumu, D. - *The Application Realisation Guide*, The Zeus Agent Building Toolkit, Intelligent Systems Research Group, BT Labs, May 1999
4. Collins, J., Ndumu, D. - *The Runtime Guide*, The Zeus Agent Building Toolkit, Intelligent Systems Research Group, BT Labs, November 1999
5. Collins, J. - *PC Manufacture – A supply chain simulation*, The Zeus Agent Building Toolkit, Intelligent Systems Research Group, BT Labs, November 1999
6. Luck, M., McBurney, P., Shehory, O., Willmott, S. and the AgentLink Community - *Agent Technology: Computing as Interaction, A Roadmap for Agent Based Computing*, University of Southampton, 2005
7. Matei, V. - *Interacția substanțelor chimice cu agenți de mediu*, Editura Universității din Ploieşti, 2004
8. Oprea, M. - *Curs Inteligență Artificială*, Universitatea Petrol Gaze din Ploieşti, specializarea Master Informatica, semestrul I, 2007-2008
9. Petre, E. - *Sistem multi-agent pentru monitorizarea poluării aerului*, Dissertation Paper, Universitatea Petrol – Gaze din Ploieşti, 2008
10. Russel, P., Norvig, T. - *Artificial Intelligence – A modern approach*, Prentice Hall, 1995

Sistemul AgentAirPol, un sistem bazat pe agenți pentru analiza poluării aerului

Rezumat

Acest articol descrie paşii de bază care ar trebui urmați în construirea unui sistem bazat pe agenți inteligenți, de la analiza şi proiectarea aplicației până la implementarea şi rularea sistemului. Aceşti paşi sunt urmați pentru a crea un astfel de sistem, folosind un software specializat - Zeus. Sistemul, numit AgentAirPol, este dedicat protecției mediului, mai ales activității industriale şi efectelor sale asupra atmosferei. Acesta a fost proiectat pentru monitorizarea ratei de poluare a aerului de către o fabrică.