SURVEILLANCE OF A HYDROELECTRIC POWER PLANT CHAIN USING SOFTWARE AGENTS

Honoriu Valean Liviu Miclea Szilard Enyedi

Dept. of Automation, Technical University of Cluj-Napoca,
15 C.Daicoviciu St., Cluj-Napoca, Romania. e-mail:
{ Honoriu.Valean, LiviuMiclea, SzilardEniedy}@aut.utcluj.ro

Abstract: *The paper presents a software agents society, which implements a more efficient control and monitoring system for a complex heterogeneous system – multiple hydroelectric power plants connected in a chain on Somes River. The agent society can be superimposed over the existing SCADA control and monitoring network and aims to increase performance, efficiency and stability in the existing monitoring system.*

Keywords: *Hydro-power plant, surveillance, software agents*

1. INTRODUCTION

Large systems, such as large manufacturing or power plants, telecommunication or computer networks, require distributed control, monitoring and diagnosis.

Some may even contain so many and various devices, that a centralized control and monitoring management becomes very difficult. This is especially true for heterogeneous systems that contain devices of different types. In this case, multiple control and monitoring management modules can be used, each responsible with a subset of the system's devices.

Another problem in managing distributed control and monitoring management for large systems scattered over large areas is the cost of connections.

Distributed monitoring management usually implies that the management is not done centrally, but locally, in a distributed manner. The system may or may not have a central control and monitoring management module. Most approaches use a central authority to generally organize the control and monitoring process and gather together the results. There are also distributed, decentralized management techniques, some involving agents.

2. SOFTWARE AGENTS

2.1. Introduction

Intelligent agents are software modules able to make decisions on their own, communicate with each other, learn new things and even "travel" from system to system (see also [2]).

This work has been supported by the Romanian National Council of Academic Research under grant INFOSOC no. 143 of 2004.

Extremely dynamic customer requirements and global competition are shifting the production configuration of manufacturing organizations away from the traditional centralized, sequentially flowing planning, control and scheduling mechanisms. This approach is rendered too slow to adapt to evolving production styles and rapid variations in customer requirements, and limits the reconfiguration capability and the flexibility of the manufacturing system. The traditional, centralized organization may also easily lead to a large proportion of the system being shut down due to a single point of failure. The multi-agent society solution, being naturally distributed and decentralized, provides an easy method to overcome these disadvantages.

The agent system can also be considered a sort of an expert system, since it maintains a knowledge base about the subsystems it accesses. However, the components (the agents) have local knowledge. Expert systems are dedicated, whereas the agent system is more flexible and can be developed further. One may even regard the agent society as a collection of expert systems.

2.2. Concepts and terms

It is assumed that the distributed system where the multi-agent society is applied is geographically partitioned – i.e., smaller, spatially separated and network-connected subsystems are working together. Moreover, the objectives of the distributed system can only be reached by the composition of a large number of tasks. Each of these tasks is performed by a particular type of agent, which requires certain skills in order to function correctly. All these skills together form a set. Some of the skills in this set are related, logically leading to a type of agent that exhibits them. We name this subset of related skills a *skill class*. The bottom line is that an agent exhibiting a certain skill class may perform the duties of any other agent in that class. Such an agent is hereafter identified as a *task agent*, as opposed to the *service agents* employed by the solution.

A certain location L, corresponding to a specific geographical partition, requires a number of skill classes. A number of task agents of various types perform the system duties at that location.

Let us consider now a single skill class at this location. This class contains a number of tasks:

$$C = \{t_1, t_2, ..., t_k\} \qquad (1)$$

A numerical score is assigned to each task, s_i, $i = 1..k$, directly proportional to the complexity of the task (which includes, but is not limited to, required resources and task execution time). Obviously, more tasks of a given type can exist at a certain location. We denote the number of tasks of type t_i with n_i, $i = 1..k$. If the number of task agents residing at the considered location and exhibiting skill class C is N, we define the *load factor* of the skill class C at location L as:

$$LF_{C-L} = \frac{\sum_{i=1}^{k} n_i \bullet s_i}{N} \qquad (2)$$

2.3. Agent society structure

The required components of the multi-agent society are presented in Fig. 1.

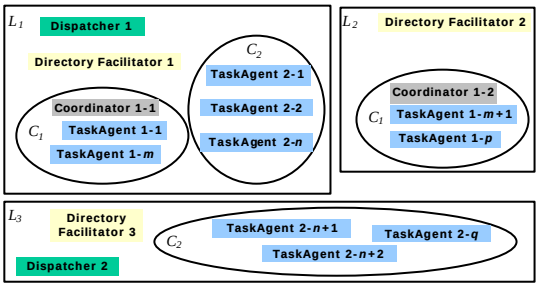

Fig. 1. Agent society structure

Locations are represented as thick-outlined rectangles, skill classes as ellipses and agents as labeled colored rectangles. We therefore have three locations, one of them requiring two skill classes, and the second and third requiring each only one skill class common to the first location. The *Directory Facilitator* (DF) agents implement the directory service. All agents register the services they offer with the local DF, which may also be queried by any agent for addresses of agents advertising a certain service. Therefore, a DF must be present at each network (geographical) location. DFs will federate over the network. An arbitrary number of *task agents* may be present at each location for the skill classes represented there.

The key agent to our approach is the Dispatcher.

There is no need for dispatchers to be present at each location. In fact, a single dispatcher would suffice, but in order to maintain the advantages of distributed processing and minimize the communicational overhead, more dispatchers may work at the same time within the society, assigned to sets of skill classes specific to a group of locations (e.g. parts storage warehouses). The task agents will then maintain contact with the dispatcher closest to their location.

Note also that C1 makes use of a Coordinator agent, while C2 does not. The coordinator is a generic concept representing the agent that performs tasks distribution within a certain skill class at a given location.

All task agents will need to register themselves with the DF as providing the skill class they are part of, so that the Dispatcher may find them when necessary. The coordinators also have to register as suppliers of the coordination service for their skill class.

2.3. Heterogeneous systems and software agents

Most of the large systems we talk about are heterogeneous, comprising a large number of devices of different types. All these devices have different hardware and/or software, tasks, dependability requirements, but all are capable of running software (in order to be able to run the agent code).

The monitoring system of the power plants is based on various data acquisition, surveillance and communication subsystems. The decentralization of this system greatly reduces the communicational overhead and increases flexibility and reliability. The multiagent approach is only natural to such a problem, as multiagent societies are naturally heterogeneous, decentralized and distributed.

An agent is, as implemented here, a piece of software capable of independent existence within an environment provided for it, which is able to communicate with entities similar to it, to unaidedly accomplish the work assigned to it and also to travel between geographically separated locations in its environment (capability named hereafter agent mobility).

The learning capability of the agents is not based on neural networks (although we are considering that, as well); it is a distributed and dynamic database, where each agent keeps the most often applied “knowledge”.

The agents’ communication capabilities and mobility lead to the concept of multiagent society, which is here a distributed collection of interacting, mobile agents, residing in different parts of the multiagent environment (physically realized rather as communicating multiple multiagent environments).

3. SYSTEM STRUCTURE

The purpose of this development is to research the possibilities of such a system and to have an experimental pilot implementation.

The system whose surveillance we want to manage with the agent system is described in table 1 It comprises multiple hydroelectric power plants that are linked together in cascade

At each component of the plant (including water reservoir and flow), there are many sensors and actuators, each of them being monitored. There is a complex monitoring system at each power plant. The system includes embedded controllers, as well as normal desktop PCs.

Monitoring, journaling and report generation for this system is too complex, because of the large amounts of recorded sensory information. Agents can manage that. They might even replace the existing SCADA monitoring and control system, since the agent-based approach is more flexible.

Tasks:

- Plant surveillance: generator, auxiliary services (water, oil, air, pressure); working water management (containers, pressure pipes, dams);
- Building surveillance: dam movement monitoring, concrete monitoring;
- Climate monitoring;
- Management of the hydroenergetic potential: lake filling, water distribution;
- Communication channel monitoring.

The proposed system structure is presented in Fig. 2.

For locations where the agents cannot access directly the sensors or the actuators, a human operator will need to access the latters with a PDA, wirelessly connected to the monitoring system.

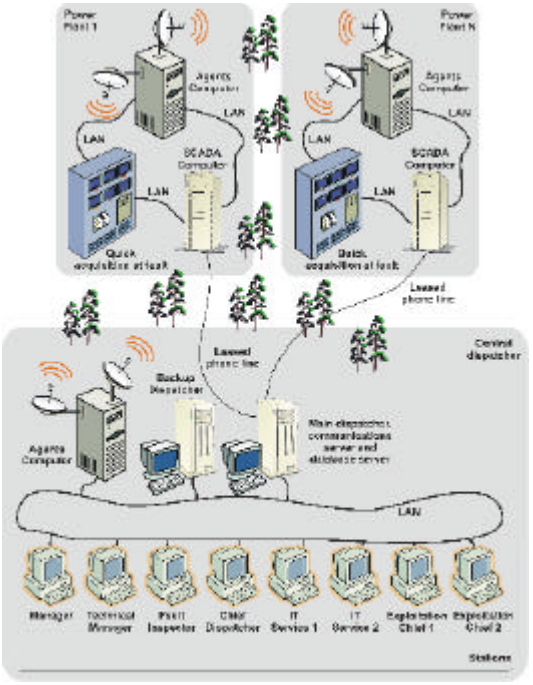

Fig. 2. The proposed structure

4. AGENT PLATFORMS CONSIDERED

With respect to the multiple vendor hardware/software in a heterogeneous system, the agents have the advantage of being based on cross-platform standards. They are different from other existing systems in this domain, because they learn the access methodologies for the previously unknown subsystems.

4.1. An Agent Development Kit for Java

This platform is built upon Java Standard Edition that runs on a usual PC desktop computer and offers a flexible, scalable and consistent task model for the agents' behavior, a natively distributed multiagent environment, strong mobility for agents (i.e. both data and execution state are transferred along with the agent code), and last but not least, powerful and standards-compliant communication facilities. The inter-agent communication in ADK complies with a subset of the FIPA ACL standard (see[8],[10]).

4.2. Our own agent platform for the PC

The aforementioned ADK is a good agent platform for the PC. However, it has two major disadvantages: for commercial use, a license must be paid, and for a fairly high-availability task as the control and monitoring of several power plants, it needs serious auditing.

Therefore, an in-house developed agent platform is a good idea. Our team is currently developing Java code/programs from scratch. When it has the needed functionality, it will be adapted to be FIPA-compliant.

Currently, we are using the ADK for simulations, and building an agent platform of our own.

4.3. Embedded Linux

Linux, the most acclaimed open source operating system, also has many downscaled embedded versions. ? CLinux [11], for example, runs on microcontrollers.

TABLE I.
THE HYDROELECTRIC POWER PLANTS PARAMETERS

Hydro power plant	Type	Basin	Number of power units	Power [MW]	Energy [GWh/Yr]	Flow [m^3/s]	Waterfall height [m]
Marisel	Underground	Fantanele	3	220	390	60	465
Tarnita	Over ground, at the dam base	Tarnita	2	45	80	65,4	81
Somesul Cald	Over ground, at the dam base	Somesul Cald	1	12	19,4	70	20,9
Gilau I	Over ground	Gilau	3	6,63	11,4	59	15,1
Gilau II	Over ground	Gilau	3	6,9	12,2	60	16
Floresti I	Over ground	Gilau	3	6,9	12,2	60	15
Floresti II	Over ground	Floresti	6	1,3	5,2	27	7

Linux, in its embedded versions, is the most powerful and resource efficient platform for embedded computational tasks. The downside is that since the native programs contain native machine instructions, they are not portable to other processors.

For more about devices with embedded Linux, see [11],[12].

4.4. Embedded Java

The Java language was originally intended for embedded application programming. It became only later the language of web applets and PC programs.

Java is currently gaining momentum in embedded programming, although not so much in its original form, but mainly in the reduced version, Micro Edition.

4.4.1. Java 2 Micro Edition

The Java 2 Micro Edition [13] is standardized, portable, has a small footprint (the original reference implementation has about 128 kilobytes), optimized for networking and very flexible.

To ensure portability among different manufacturers' devices, the MIDP 1.0 (Mobile Information Device Profile) and specification establishes some basic functionality for the first generation Java-enabled mobile devices. This guarantees that the programs – "midlets" – will run on any MIDP 1.0 certified hardware.

MIDP 1.0 offers only HTTP type connections by default, but there are a few workarounds to have always-on, flexible, raw socket connections – proprietary network connections – between the server and the mobile device. MIDP 2.0 is more flexible in this respect, but few mobile devices comply with it.

On need, the j2me agents can be easily extended with additional functions, enabling a device's additional testing abilities.

The drawback of the j2me solution is that from its conception, Java (Enterprise, Standard or Micro) has been designed for portability. This means that it does not allow native access to the hardware, only through the functions of the virtual machine. On the other hand, special, device-specific classes can be developed, which bypass the virtual machine and access the hardware directly.

Another drawback is that the "midlets" – j2me programs – can be installed and run only on the user's request. This is a security measure, aiming at protecting the user's handheld – the original target of j2me – from unwanted programs. However, if there is already a midlet running on the device, with an active network connection, it can send and receive data, including agents.

4.4.2. Microcontroller with built-in Internet and Java functions

There are devices [14] on the market which are hardware-software combinations, sort of a microcontroller with Internet and Java functions, its main selling points.

Their main advantage is in providing full network and Internet connectivity for any embedded application.

4.4.3. True Java hardware machines

There are chips available that are true hardware Java machines [15], executing MIDP 2.0 Java programs (called bytecode) directly in silicon. The advantage is that the chip runs Java code ten times faster than software implementations would. The inherent advantage of the Java language is that the programs are two to three time smaller than in other languages, which is very important for embedded systems, where memory is always a concern. The disadvantage of these processors is the price.

4.5. Single Board Computers

An SBC is, in fact, a hardware platform. It is a powerful computer, usually with network access, audio and video capabilities, lots of processing power, but all crammed on one small printed circuit board. There are even 45x45mm SBC boards.

Most of them use x86 compatible processors, thus are able to run MS Windows. Nevertheless, the majority uses Linux, for its flexibility. See [12] on Linux-enabled SBCs.

5. AGENT COMMUNICATION

The IEEE 1232 family of standards describe common exchange formats and software services for reasoning systems used in system test and diagnosis. The goal is to make the data exchange between two different diagnostic reasoners easy. The standard also defines software interfaces, for the use of external tools that can access the diagnostic data in a consistent manner. It allows exchanging diagnostic information and embedding diagnostic reasoners in any test environment.

At software level, the agents communicate with each other through the FIPA (Foundation for Intelligent Physical Agents) ACL (Agent Communication Language) [7]. FIPA ACL specifications describe aspects of the structure of messages and the ontology service.

The FIPA MTP (Agent Message Transport Protocol) specifications [7] present different ways of communication for the agents to exchange data. IIOP (Internet Inter-ORB Protocol), WAP (Wireless Application Protocol) and HTTP (HyperText Transfer Protocol), TCP/IP over wireline are described, as well as generic wireless solutions. They also deal with bit-oriented, string-oriented and XML-oriented message representations. Our agents, in their current development status, use TCP/IP over wireline and wireless connections, with the messages in ASCII string format. They ask information from the central database through HTTP. A newer version, with XML, is being developed, to simplify inter-agent, agent-to-database communication and use of protocols like HTTP and WAP.

At hardware level, the agents use whatever communication layer is available for the device (serial, I2C, dial-up, Ethernet or other). We have also considered embedded TCP/IP solutions.

For a system with mobile subsystems to be tested, short range, standardized radio-based Bluetooth chips can be used. For large scattered systems, radio-based Wi-Fi solutions or GPRS boards are available. GPRS boards are adequate for low-cost, always-on sporadic communication over large distances.

6. AGENT SOCIETY STRUCTURE

The system is decomposed into multiple subsystems, each subsystem being managed by one or more statically or dynamically allocated agents, based on the knowledge level of the agent with respect to that subsystem. The agents evolve through learning – i.e. they store the problems and the solutions that have been found, in order to be able to solve them alone next time.

Each agent of the society has detailed knowledge about the subsystem it manages, as well as some knowledge of the neighboring subsystems.

As shown in Fig. 3, each agent has two main components: **Kernel** and **Interface** (see also [1], [6]).

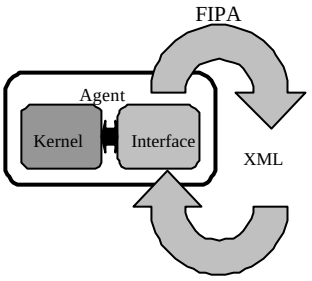

Fig. 3. The agent structure

The Kernel component differs from agent to agent, depending on the specific duty of the agent: collector agent, node agent, connector agent, graphic user interface agent, etc.

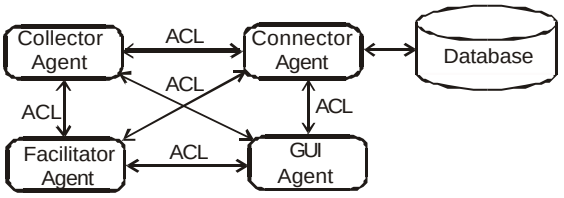

Fig. 4. Data Flow

In order to be able to communicate, each agent has an Interface component, which manages the FIPA-compliant communication [7] with other agents or with a central knowledge base, eventually. This component has the same behavior for every agent, since the agents use the same protocol to communicate.

The Interface has three major responsibilities:

- Generates an XML [9] document that contains data received from the Kernel;
- Sends and receives data towards/from the Interface of other agents or the knowledge base;
- Decodes the received XML documents and sends the data to the Kernel.
- There are two reasons why the agent interfaces communicate via XML:

First, XML is a text document. In this case, the load on the communication lines will decrease and errors will be easier to correct;

Second, different agents can be written in different languages. All of them will be able to "understand" XML and communicate with it.

The following agents compose the society:

- The **Collector Agent** communicates directly with the sensors. It collects data from sensors: voltage, current, active energy, reactive energy, power, etc. and the kernel is designed to support these different types of sensors.
- The **Node Agent** gathers together the data from the Collector Agents or distributes the data to the Executive Agents. It acts at any subsystem level.
- The **Connector Agent** manages the database. It allows other agents to interact with the database, administrates the database and provides some additional actions (eliminate old records for avoiding database to uncontrollable grow).
- The **GUI Agent** (Graphic User Interface Agent) offers a friendly user interface for the human operator, accessible via LAN or via Internet (using a secured access).
- The **Directory Facilitator Agent** facilitates the communication between the agents.
- The **Arbiter Agent** decides in case of possible conflicts that may appear between similar information sent by different agents.

The data flow, inside the agent society, is presented in Fig. 4.

6.1. The Collector agents

The collector agent inputs from the process the measured values. These measurements can be voltage, current, active and reactive power, active and reactive energy, frequency, water level, water flow, etc. The measurement type is specified using an identifier. The agent scans periodically the sensors and sends the measured values to the connector agent for storing the values in the database, and to the GUI agents, for on-line visualisation. The agent executes in parallel several tasks for performing the requested operations.

At the initialisation, the agent generates a map, which stores the input signals for each location. This map can be seen as an indexed table, which contains a non-empty list of objects representing signals. The executed algorithm is the following:

```
obtain the PLC input number
compute the bytes number for multiplexer
for (existing multiplexer cannels ) execute{
  for (bytes number) execute {
    generate the signal group code
     initialize signal information list
    for (signals in group) execute{
      read signal
      add information in list
    }
   add record in map }
  }
}
```

This map is generated because of the following reason: if data is stored in the database when the agent reads the measurements, the connector agent could not be able to deal with the entire data flow. Data will be stored only if the list, with a specified dimension is completed.

6.2 The Connector agent

The Connector agent manages the database. It allows other agents to interact with the database, administrates the database and provides some additional actions (eliminate old records for avoiding database to uncontrollable grow).

6.2.1 Database implementation

Data provided by the collector agents is stored in a database, which contains all the information about the plants parameters. The stored data will be used by the control agents for generating control signals and by the GUI agents for interfacing the human operators. These three types of agents will access the database via the connector agents. Connector agents grant unrestrictioned access for collector agents, but

provide access only via secured lines for GUI agents (based on user and password). Because collected data is stored at relatively short periods of time, another task executed by the database agent is to eliminate old records, for avoiding the uncontrollable growth of the database. The agent will use a threshold with time-variable amplitude. For just recorded data the threshold amplitude is zero and it grows with time. The database agent will periodically interrogate the database and all the adjacent records with values within the threshold will be considered equal and will be deleted.

The following tables compose database (see Fig. 5):

T_SystemLocation – stores information about the plants location. Associates to each location name an identifier;
T_SystemCounters – stores information about the counter's value for each location.
The counters are selected using an identifier and the link with the first table is implemented by the location identifier field;
T_ElementType – stores the types of measurements and associate identifiers to the measurement types;
T_System – stores the types and names of the measurements. The records contains also the location identifier and the measurement identifier;

The records for the following eight tables contain as primary keys the location identifier, the measurement identifier and the measurement data. Also, the records will store the measurement value. Due to this implementation, the database can be interrogated using various filters: location, measurement or time.

T_VoltageValues – stores the measured voltage values;
T_CurrentValues – stores the measured current values;
T_PowerValuesP – stores the values of measured active power.
T_PowerValuesQ – stores the values of measured reactive power;
T_LevelValues – stores the measured water level**s;**
T_FrequencyValues – stores the measured values for frequency;
T_ActiveEnergyCounter – stores the measured values for active energy;
T_ReactiveEnergyCounter – stores the measured values for reactive energy;
T_PLCInputSignalsGroup – stores the digital signal group names. Primary keys used are the location identifier and the group identifier;
T_PLCInputSignals – stores the names associated to the digital signals taking into account the location identifier and the group identifier, as primary keys;
T_ReportValues – stores the reports. The reports will contain the location identifier, the signal identifier, the date of the report and the message;

6.2.2 The connector agent implementation

The connector agent represents the agents society interface with the database. The agent must be designed to efficiently respond to the all

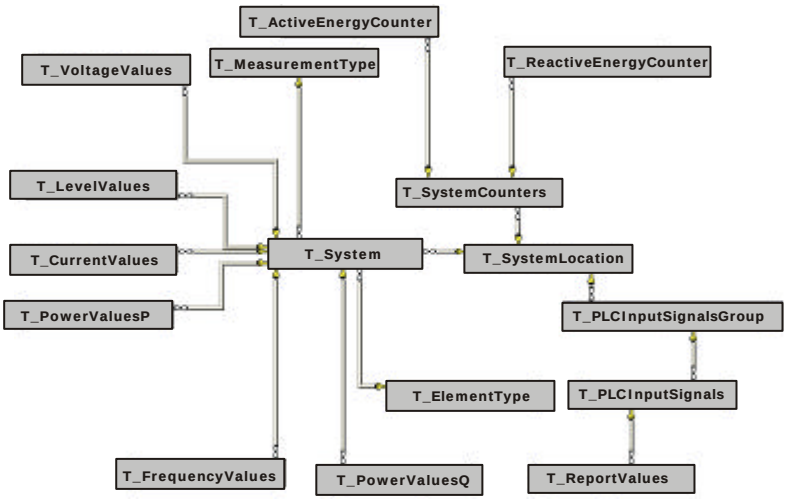

Fig. 5. The database structure

the interrogation requests. For a major number of requests to be made simultaneously in an optimal way, the database connection must be managed in an efficient manner.

An interrogation needs to create a connection to a database. This creation involves usually a high cost, taking into account the hardware and software resources. For a connection, the database server has to allocate a communication channel, an appropriate memory space, but also must execute an autentification operation. These phases are executed for each connection request, and the performance of the application decreases.

The solution implemented for the connector agent creates a database connection pool. The connection pool maintains a set of open connections to the database server and shares these connections among the agents which interrogate the database. Thus, the database agent will eficiently serve all the requests and the overall reliability of the application will increase significantly.

The implementation of the database connection pool is presented in Fig. 6.

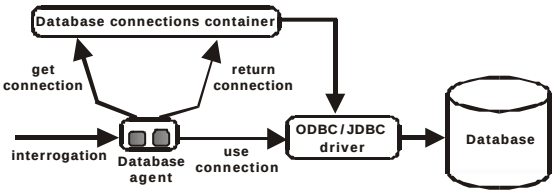

Fig. 6. Solving an interrogation request

6.3 The GUI agent

The GUI agents display data in real time and assist human operator to retrieve past data, stored in database. Data is displayed in a friendly graphical form. The agent society implements a GUI agent at the local dispatcher level and also a GUI agent for each hydroelectric power plant location. The agent implemented at the dispatcher level can display information about all the hydroelectric power plants, but the agents implemented at the hydroelectric power plant level can access only local data. This particular type of behaviour must be implemented transparent for the human operator, thus, the agent must to be able to decide itself the data access level, depending on the habitat in which it was started.

For displaying in real-time data, the connector agents must know the addresses of the GUI agents. These addresses are retrieved during the initialisation phase. If the address retrieve fails, the connector agents will search GUI agents, during the life-time, at well defined time moments.

Data displayed in real time (received from the collector agents) consists in:

- measurements from the turbine-generator ensemble
- measurements from the current transformers
- measurements from the power lines
- water levels

The following information is stored in database and received from the connector agent:

- power (active and reactive) evolution
- voltage evolution
- current evolution
- changes in water level

An example of a GUI agent panel for voltage evolution (for Tarnita hydroelectric power plant) is presented in Fig. 7.

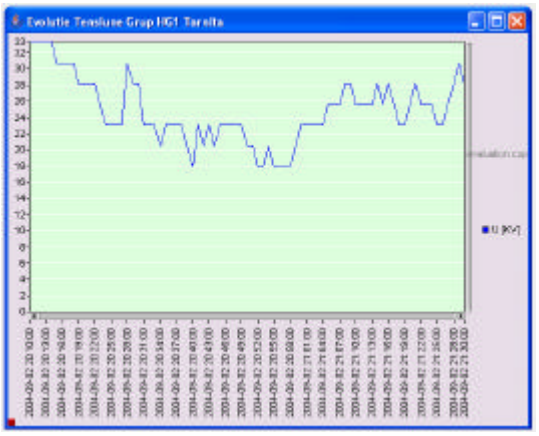

Fig. 7. Voltage evolution panel

6.4 The Directory Facilitator agent

The Directory Facilitator agent provides a yellow pages service for the society. This service involves the maintaining of a list with the skills of the agents. All the agents are able to interrogate the list or modify the own entry. The directory facilitator agents must be connected in the network, for providing global solutions to the requests received from other agents,

solutions that must include references to other remote agents.
The directory facilitator agent execute the following actions:

- registration of the agents. An agent must be registered in the directory facilitator list in order to be visible to other agents and to present its skills
- un-registration of the agents. After un-registration, the agent will not be visible to the rest of the agents and will not be able to communicate with other agents
- interrogation. An agent can interrogate the directory facilitator agent in order to find another agent, but the directory facilitator do not guarantee the validity of the provided information, because there are no restriction in agent registration. The directory facilitator cannot verify the correctness of information provided by another agent at registration.

7. CONCLUSIONS

We have presented the application of software agent mechanisms in improving the control and monitoring management of a hydroelectric power plant chain.

The chain is a complex system, with many sensors and actuators, and all components are interdependent.

The main goal of using agents is that they are able to decide by themselves in their responsibility area, without propagating the problem to upper levels. Moreover, as time goes by, they learn, therefore solving re-appearing problems faster.

There are many possible agent platforms, but, at least for the PC version, the existing platforms are both expensive and have not been tested enough, therefore an in-house-developed agent platform has been started.

For the embedded versions, Linux or embedded Java is the best choice, given the availability of knowledge and development tools, as well as corresponding previous experience of the team.

The agents communicate using the standardized FIPA Agent Communication Language. This ensures that they will be able to communicate even with agents from other platforms.

ACKNOWLEDGMENT

The research has been supported by the Romanian National Council of Academic Research under grant INFOSOC no. 143 of 2004, "Multi-agent and intelligent agent techniques in the telematic control and surveillance system for a chain of hydroelectric plants" – AGENTEL.

The authors wish to thank the students Iulia Pop and Ovidiu Lupas for their interesting ideas and hard work in the implementation of the project. The authors acknowledge the support of The Institute for Automation, Center for Technology Transfer Cluj IPA S.A. and especially thanks to Mr. Ioan Stoian, the manager of IPA S.A.

REFERENCES

[1] M. Albert., T. Längle, W. Worn "Development Tool for Distributed Monitoring and Diagnosis System", DIAMOND project, EUSPRIT EP 28735, 2002.

[2] J. Ferber, *Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence*, Boston, MA: Addison-Wesley, USA, 1999, ISBN 0-201-36048-9.

[3] E. E. Mangina, "Application of Intelligent Agents in Power Industry: Promises and complex issues" Proceedings of the 3rd International/Central and Eastern European Conference on Multi-Agent Systems, June 16-18, 2003, Prague, pp. 564-573.

[4] L. Miclea, Enyedi Sz., H. Valean, I. Stoian, "Software Agents In The Control And Surveillance Of A Hydroelectric Power Plant Chain", *The Mediterranean Journal of Measurement and Control*, Vol I. No. 3, 2005, pp. 129-137, ISSN 1743-9310.

[5] L. M. Tolbert, H. Qi, F. Z. Peng, "Scalable Multi-Agent System for Real-Time Electric Power Management," IEEE *Power Engineering Society Summer Meeting*, July 15-19, 2001, Vancouver, Canada, pp. 1676-1679.

[6] H. Valean, T. Letia, A. Astilean, "Distributed Agent-Based Monitoring And Diagnosis For Large-Scale Systems", *Proceedings of the 2004 IEEE-TTTC International Conference on Automation, Quality and Testing*,

Robotics AQTR 2004 (Theta 14), *Tome I*, pp. 311-316, ISBN 973-713-046-4

[7] ***, "FIPA Specifications", Foundation for Intelligent Physical Agents, 2005, available at [www.fipa.org/repository/index.html]

[8] ***, "Agent Communication Language Specifications", Foundation for Intelligent Physical Agents, 2002, available at [www.fipa.org/repository/aclspecs.html

[9] *** Extensible Markup Language (2001) [www.xml.org]

[10] ***, "ADK Developer's Guide", Tryllian Holding BV, 2002, available at [http://www.tryllian.com], last accessed date: 3/20/2003.

[11] ***, "Embedded Linux/Microcontroller Project", µCLinux, 2005, available at [http://www.uclinux.org], last accessed date: 7/14/2005.

[12] ***, "Linux Devices", Linux Devices site, 2005, available at [www.linuxdevices.com], last accessed date: 7/14/2005.

[13] ***, "Java 2 Platform, Micro Edition (J2ME)", Sun Developer Network, 2005, available at [http://java.sun.com/j2me], last accessed date: 7/14/2005.

[14] ***, "TINI – Tiny InterNet Interfaces", Maxim Integrated Products, 2005, available at [www.maxim-ic.com/TINIplatform.cfm], last accessed date: 7/14/2005.

[15] ***, "Embedded Low Power Java Microprocessors", aJile Systems, 2005, available at [http://www.ajile.com/].