Monitoring and Diagnosing Railway Signalling with Logic-Based Distributed Agents

Viviana Mascardi [1], Daniela Briola [1], Maurizio Martelli[1],
Riccardo Caccia [2], Carlo Milani [2],

[1] DISI, University of Genova, Italy
{Viviana.Mascardi, Daniela.Briola, Maurizio.Martelli}@disi.unige.it
[2] IAG/FSW, Ansaldo Segnalamento Ferroviario S.p.A., Italy
{Caccia.Riccardo, Milani.Carlo}@asf.ansaldo.it

Abstract. This paper describes an ongoing project that involves DISI, the Computer Science Department of Genova University, and Ansaldo Segnalamento Ferroviario, the Italian leader in design and construction of signalling and automation systems for railway lines. We are implementing a multiagent system that monitors processes running in a railway signalling plant, detects functioning anomalies, provides diagnoses for explaining them, and early notifies problems to the Command and Control System Assistance. Due to the intrinsic rule-based nature of monitoring and diagnostic agents, we have adopted a logic-based language for implementing them.

1 Introduction

According to the well known definition of N. Jennings, K. Sycara and M. Wooldridge an agent is "*a computer system, situated in some environment, that is capable of* flexible autonomous action in order to meet its design objectives" [1].

Distributed diagnosis and monitoring represent one of the oldest application fields of declarative software agents. For example, ARCHON (ARchitecture for Cooperative Heterogeneous ON-line systems [2]) was Europe's largest ever project in the area of Distributed Artificial Intelligence and exploited rule-based agents. In [3], Schroeder et al. describe a diagnostic agent based on extended logic programming. Many other multiagent systems (MASs) for diagnosis and monitoring based on declarative approaches have been developed in the past [4,5,6,7].

One of the most important reasons for exploiting agents in the monitoring and diagnosis application domain is that an agent-based distributed infrastructure can be added to any existing system with *minimal or no impact* over it. Agents look at the processes they must monitor, be they computer processes, business processes, chemical processes, by "looking over their shoulders" without interfering with their activities. The "no-interference" feature has an enormous importance, since changing the processes in order to monitor them would be often unfeasible.

Also the increase of *situational* awareness, essential for coping with the situational complexity of most large real applications, motivates an agent-oriented approach. Situational awareness is a mandatory feature for the successful monitoring and deci-

sion-making in many scenarios. When combined with reactivity, situatedness may lead to the early detection of, and reaction to, anomalies. The simplest and most natural form of reasoning for producing diagnoses starting from observations is based on rules.

This paper describes a joint academy-industry project for monitoring and diagnosing railway signalling with distributed agents implemented in Prolog, a logic programming language suitable for rule-based reasoning. The project involves the Computer Science Department of Genova University, Italy, and Ansaldo Segnalamento Ferroviario, a company of Ansaldo STS group controlled by Finmeccanica, the Italian leader in design and construction of railway signalling and automation systems. The project aims to develop a monitoring and diagnosing multiagent system implemented in JADE [8] extended with the tuProlog implementation of the Prolog language [9] by means of the DCaseLP libraries [10].

The paper is structured in the following way: Section 2 describes the scenario where the MAS will operate, Section 3 describes the architecture of the MAS and the DCaseLP libraries, Section 4 concludes and outlines the future directions of the project.

2 Operating Scenario

The Command and Control System for Railway Circulation ("Sistema di Comando e Controllo della Circolazione Ferroviaria", SCC) is a framework project for the technological development of the Italian Railways ("Ferrovie dello Stato", FS), with the following targets:

- introducing and extending automation to the command and control of railway circulation over the principal lines and nodes of the FS network;
- moving towards innovative approaches for the infrastructure and process management, thanks to advanced monitoring and diagnosis systems;
- improving the quality of the service offered to the FS customers, thanks to a better regularity of the circulation and to the availability of more efficient services, like delivery of information to the customers, remote surveillance, and security.

The SCC project is based on the installation of centralized Traffic Command and Control Systems, able to remotely control the plants located in the railway stations and to manage the movement of trains from the Central Plants (namely, the offices where instances of the SCC system are installed). In this way, Central Plants become command and control centres for railway lines that represent the main axes of circulation, and for railway nodes with high traffic volumes, corresponding to the main metropolitan and intermodal nodes of the FS network.

An element that strongly characterizes the SCC is the strict coupling of functionalities for circulation control and functionalities for diagnosis and support to upkeep activities, with particular regard to predictive diagnostic functionalities aimed to enable on-condition upkeep. The SCC of the node of Genova, which will be employed as a case-study for the implementation of the first MAS prototype, belongs to

the first six plants developed by Ansaldo Segnalamento Ferroviario. They are independent one from the other, but networked by means of a WAN, and cover 3000 Km of the FS network. The area controlled by the SCC of Genova covers 255 km, with 28 fully equipped stations plus 20 stops.

The SCC can be decomposed, both from a functional and from an architectural point of view, into five subsystems. The MAS that we are implementing will monitor and diagnose critical processes belonging to the Circulation subsystem whose aims are to implement remote control of traffic and to make circulation as regular as possible. The two processes we have taken under consideration are Path Selection and Planner.

The *Planner* process is the back-end elaboration process for the activities concerned with Railway Regulation. There is only one instance of the Planner process in the SCC, running on the server. It continuously receives information on the position of trains from sensors located in the stations along the railway lines, checks the timetable, and formulates a plan for ensuring that the train schedule is respected.

A plan might look like the following: “Since the InterCity (IC) train 5678 is late, and IC trains have highest priority, and there is the chance to reduce the delay of 5678 if it overtakes the regional train 1234, then 1234 must stop on track 1 of Arquata station, and wait for 5678 to overtake it”. Plans formulated by the Planner may be either confirmed by the operators working at the workstations or modified by them. The process that allows operators to modify the Planner's decision is Path Selection.

The *Path Selection* process is the front-end user interface for the activities concerned with Railway Regulation. There is one Path Selection process running on each workstation in the SCC. Each operator interacts with one instance of this process.

There are various operators responsible for controlling portions of the railway line, and only one senior operator with a global view of the entire area controlled by the SCC. The senior operator coordinates the activities of the other operators and takes the final decisions. The Path Selection process visualizes decisions made by the Planner process and allows the operator to either confirm or modify them. For example, the Planner might decide that a freight train should stop on track 3 of Ronco Scrivia station in order to allow a regional train to overtake it, but the operator in charge for that railway segment might think that track 4 is a better choice for making the freight train stop. In this case, after receiving an “ok” from the senior operator, the operator may input its choice thanks to the interface offered by the Path Selection process, and this choice overcomes the Planner's decisions. The Planner will re-plan its decisions according to the new input given by the human operator.

The SCC Assistance Centre, that provides assistance to the SCC operators in case of problems, involves a large number of domain experts, and is always contacted after the evidence of a malfunctioning. The cause of the malfunctioning, however, might have generated minutes, and sometimes hours, before that the SCC operator(s) experienced the malfunctioning. By integrating a monitoring and diagnosing MAS to the circulation subsystem, we aim to equip any operator of the Central Plant with the means for early detecting anomalies that, if reported in a short time, and before their effects have propagated to the entire system, may allow the prevention of more serious problems including circulation delays.

When a malfunctioning is detected, the MAS, besides alerting the SCC operator, will always alert the SCC Assistance Centre in an automatic way. This will not only

speed up the implementation of repair actions, but also allow the SCC Assistance Centre with recorded traces of what happened in the Central Plant.

3 Logic-based Monitoring Agents

In order to provide the functionalities required by the operating scenario, we designed a MAS where different agents exist and interact. The MAS implementation is under way but its architecture has been completely defined, as well as the tools and languages that will be exploited for implementing a first prototype. The following sections provide a brief overview of DCaseLP, the multi-language prototyping environment that we are going to use, and of the MAS architecture and implementation.

3.1 DCaseLP: a Multi-Language Prototyping Environment for MASs

DCaseLP [10] stands for Distributed Complex Applications Specification Environment based on *Logic Programming*. Although initially born as a logic-based framework, as the acronym itself suggests, DCaseLP has evolved into a multi-language prototyping environment that integrates both imperative (object-oriented) and declarative (rule-based and logic-based) languages, as well as graphical ones. The languages and tools that DCaseLP integrates are UML and an XML-based language for the analysis and design stages, Java, JESS and tuProlog for the implementation stage, and JADE for the execution stage. Software libraries for translating UML class diagrams into code and for integrating JESS and tuProlog agents into the JADE platform are also provided.

Both the architecture of the MAS, and the agent interactions taking place there, are almost simple. On the contrary, the rules that guide the behaviour of agents are sophisticated, leading to implement agents able to monitor running processes and to quickly diagnose malfunctions.

For these reasons, among the languages offered by DCaseLP, we have chosen to use tuProlog for implementing the monitoring agents, and Java for implementing the agents at the lowest architectural level, namely those that implement the interfaces to the processes. In this project we will take advantage neither of UML nor of XML, which prove useful for specifying complex interaction protocols and complex system architectures.

Due to space constraints, we cannot provide more details on DCaseLP. Papers describing its usage, as well as manuals, tutorials, and the source code of the libraries it provides for integrating JESS and tuProlog into JADE can be downloaded from `http://www.disi.unige.it/person/MascardiV/Software/DCaseLP.html`.

3.2 MAS Architecture

The architecture of the MAS is depicted in Figure 1. There are four kinds of agent, organized in a hierarchy: Log Reader Agents, Process Monitoring Agents, Computer Monitoring Agents, and Plant Monitoring Agents.

Agents running on remote computers are connected via a reliable network whose failures are quickly detected and solved by an ad hoc process (already existing, out of the MAS). If the network becomes unavailable for a short time, the groups of agents running on the same computer can go on with their local work. Messages directed to remote agents are saved in a local buffer, and are sent as soon as the network comes up again. The human operator is never alerted by the MAS about a network failure, and local diagnoses continue to be produced.

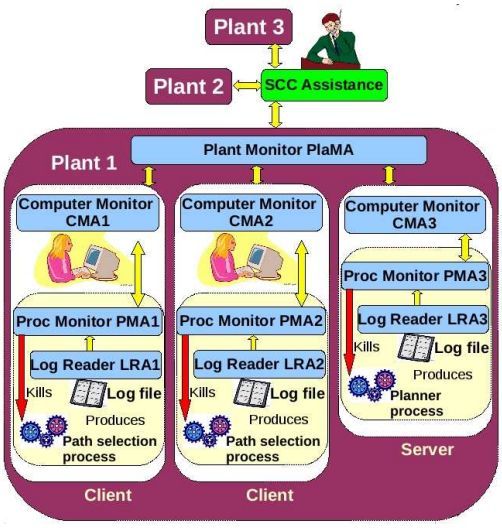

Fig. 1. The system architecture

Log Reader Agent. In our MAS, there is one Log Reader Agent (LRA) for each process that needs to be monitored. Thus, there may be many LRAs running on the same computer. Once every *m* minutes, where *m* can be set by the MAS configurator, the LRA reads the log file produced by the process *P* it monitors, extracts information from it, produces a symbolic representation of the extracted information in a format amenable of logic-based reasoning, and sends the symbolic representation to the Process Monitoring Agent in charge of monitoring *P*. Relevant information to be sent to the Process Monitoring Agent include loss of connection to the net and life of the process. The parameters, and corresponding admissible values, that an LRA associated with the "Path Selection" process extracts from its log file, include `connection_to_server (active, lost); answer_to_life (ready, slow, absent); cpu_usage (normal, high); memory_usage (normal, high); disk_usage (normal, high); errors (absent, present).`

The `answer_to_life` parameter corresponds to the time required by a process to answer a message. A couple of configurable thresholds determines the value of this paramenter: *time* < *threshold1* implies `answer_to_life ready`; *threshold1* <= *time* < *threshold2* implies `answer_to_life slow`; *time* >= *threshold2* implies `answer_to_life absent.`

In a similar way, the parameters and corresponding admissible values that an LRA associated with the "Planner" process extracts from its log file include `connection_to_client (active, lost); computing_time (normal, high); managed_conflicts (normal, high); managed_trains (normal, high); answer_to_life, cpu_usage, memory_usage, disk_usage, errors`, in the same way as the "Path Selection" process.

LRAs have a time-driven behaviour, do not perform any reasoning over the information extracted from the log file, and are neither proactive, nor autonomous. They may be considered very simple agents, mainly characterized by their social ability, employed to decouple the syntactic processing of the log files from their semantic processing, entirely demanded to the Process Monitoring Agents. In this way, if the format of the log produced by process *P* changes, only the LRA that reads that log needs to be modified, with no impact on the other MAS components.

Process Monitoring Agent. Process Monitoring Agents (PMAs) are in a one-to-one correspondence with LRAs: the PMA associated with process *P* receives the information sent by the LRA associated with *P*, looks for anomalies in the functioning of *P*, provides diagnoses for explaining their cause, reports them to the Computer Monitoring Agent (CMA), and in case kills and restarts *P* if necessary. It implements a sort of social, context-aware, reactive and proactive expert system characterized by rules like:

- **if** the answer to life of process *P* is slow, **and** it was slow also in the previous check, **then** there might be a problem either with the network, or with *P*. The PMA has to inform the CMA, and to wait for a diagnosis from the CMA. If the CMA answers that there is no problem with the network, then the problem concerns *P*. The action to take is to kill and restart *P*.
- **if** the answer to life of process *P* is absent, **then** the PMA has to inform the CMA, and to kill and restart *P*.
- **if** the life of process *P* is right, **then** the PMA has to do nothing.

Computer Monitoring Agent. The CMA receives all the messages arriving from the PMAs that run on that computer, and is able to monitor parameters like network availability, CPU usage, memory usage, hard disk usage.

The messages received from PMAs together with the values of the monitored parameters allow the CMA to make hypotheses on the functioning of the computer where it is running, and of the entire plant. For achieving its goal, the CMA includes rules like:

- **if** only one PMA reported problems local to the process it monitors, then there might be temporary problems local to the process. No action needs to be taken.
- **if** more than one PMA reported problems local to the process it monitors, **and** the CPU usage is high, **then** there might be problems local to this computer. The action to take is to send a message to the Plant Monitoring Agent **and** to make a pop-up window appear on the computer monitor, in order to alert the operator working with this computer.

- **if** more than one PMA reported problems due either to the network, or to the server accessed by the process, **and** the network is up, **then** there might be problems to the server accessed by the process. The action to take is to send a message to the Plant Monitoring Agent to alert it **and** to make a pop-up window appear on the computer monitor.

If necessary, the CMA may ask for more information to the PMA that reported the anomaly. For example, it may ask to send a detailed description of which hypotheses led to notify the anomaly, and which rules were used.

Plant Monitoring Agent. There is one Plant Monitoring Agent (PlaMA) for each plant. The PlaMA receives messages from all the CMAs in the plant and makes diagnoses and decisions according to the information it gets from them. It implements rules like:

- **if** more than one CMA reported a problem related to the same server *S*, **then** the server *S* might have a problem. The action to take is to notify the SCC Assistance Centre in an automatic way.
- **if** more than one CMA reported a problem related to a server, **and** the servers referred to by the CMAs are the different, **then** there might be a problem of network, but more information is needed. The action to take is to ask to the CMAs that reported the anomalies more information about them.
- **if** more than one CMA reported a problem of network **then** there might be a problem of network. The action to take is to notify the SCC Assistance Centre in an automatic way.

The SCC Assistance Centre receives notifications from all the plants spread around Italy. It may cross-relate information about anomalies and occurred failures, and take repair actions in a centralized, efficient way.

As far as the MAS implementation is concerned, LRAs are being implemented as "pure" JADE agents: they just parse the log file and translate it into another format, thus there is no need to exploit Prolog for their implementation. PMAs, CMAs, and PlaMA are being implemented in tuProlog and integrated into JADE by means of the libraries offered by DCaseLP. Prolog is very suitable for implementing the agent rules that we gave in natural language before.

4 Conclusions

The joint DISI-Ansaldo project confirms the applicability of MAS technologies for concrete industrial problems. Although the adoption of agents for process control is not a novelty, the exploitation of declarative approaches outside the boundaries of academia is not widespread. Instead, the Ansaldo partners consider Prolog a suitable language for rapid prototyping of complex systems: this is a relevant result.

The collaboration will lead to the implementation of a first MAS prototype by the end of June 2008, to its experimentation and, in the long term perspective, to the im-

plementation of a real MAS and its installation in both central plants and stations along the railway lines. The possibility to integrate agents that exploit statistical learning methods to classify malfunctions and agents that mine functioning reports in order to identify patterns that led to problems, will be considered. More sophisticated rules will be added in the agents in order to capture the situation where a process is killed and restarted a suspicious number of times in a time unit. In that case, the PlaMa will need to urgently inform the SCC Assistance.

The transfer of knowledge that DISI is currently performing will allow Ansaldo to pursue its goals by exploiting internal competencies in a not-so-distant future.

Acknowledgments
The authors acknowledge Gabriele Arecco who is implementing the MAS described in this paper, and the anonymous reviewers for their constructive comments.

References

1. Jennings, N. R. , Sycara, K. P., Wooldridge, M.: A roadmap of agent research and development. In: Autonomous Agents and Multi-Agent Systems, 1(1):7–38 (1998).
2. Jennings, N. R., Mamdani, E. H., Corera, J. M., Laresgoiti, I., Perriollat, F., Skarek, P., Zsolt Varga, L.: Using Archon to develop real-world DAI applications, part 1. IEEE Expert, 11(6):64–7 (1996).
3. Schroeder, M., De Almeida Mòra, I., Moniz Pereira, L.: A deliberative and reactive diagnosis agent based on logic programming. In Singh, M. P., Rao, A. S., Wooldridge, M. editors, 4th International Workshop on Agent Theories, Architectures, and Languages, ATAL'97, Proc., LNCS, volume 1365, pp 293–307. Springer (1998).
4. Leckie, C., Senjen, R., Ward, B., Zhao, M.: Communication and coordination for intelligent fault diagnosis agents. In 8th IFIP/IEEE International Workshop for Distributed Systems Operations and Management, DSOM'97, Proc., pp 280–291 (1997).
5. Semmel, G. S., Davis, S. R., Leucht, K. W., Rowe, D. A., Smith, K. E., Boloni, L.: Space shuttle ground processing with monitoring agents. IEEE Intelligent Systems, 21(1):68–73 (2006).
6. Weihmayer, T., Tan, M.: Modeling cooperative agents for customer network control using planning and agent-oriented programming. In IEEE Global Telecommunications Conference, Globecom'92, Proc., pp 537– 543. IEEE (1992).
7. Balduccini, M., Gelfond, M.: Diagnostic reasoning with a-prolog. Theory Pract. Log. Program., 3(4):425–461 (2003).
8. Bellifemine, F. L., Caire, G., Greenwood, D.: Developing Multi-Agent Systems with JADE. Wiley (2007).
9. Denti, E., Omicini, A., Ricci, A.: tuProlog: A lightweight prolog for internet applications and infrastructures. In I. V. Ramakrishnan, editor, 3rd International Symposium on Practical Aspects of Declarative Languages, PADL'01, Proc., pp 184–198. Springer (2001).
10. Mascardi, V., Martelli, M., Gungui, I.: DCaseLP: a prototyping environment for multi-language agent systems. In Dastani, M., El-Fallah Seghrouchni, A., Leite, J., Torroni, P. editors, 1st Int. Workshop on Languages, Methodologies and Development Tools for Multi-Agent Systems, LADS'007, Proc., LNCS. Springer (2008). To appear.