A Prolog-Based MAS for Railway Signalling Monitoring: Implementation and Experiments

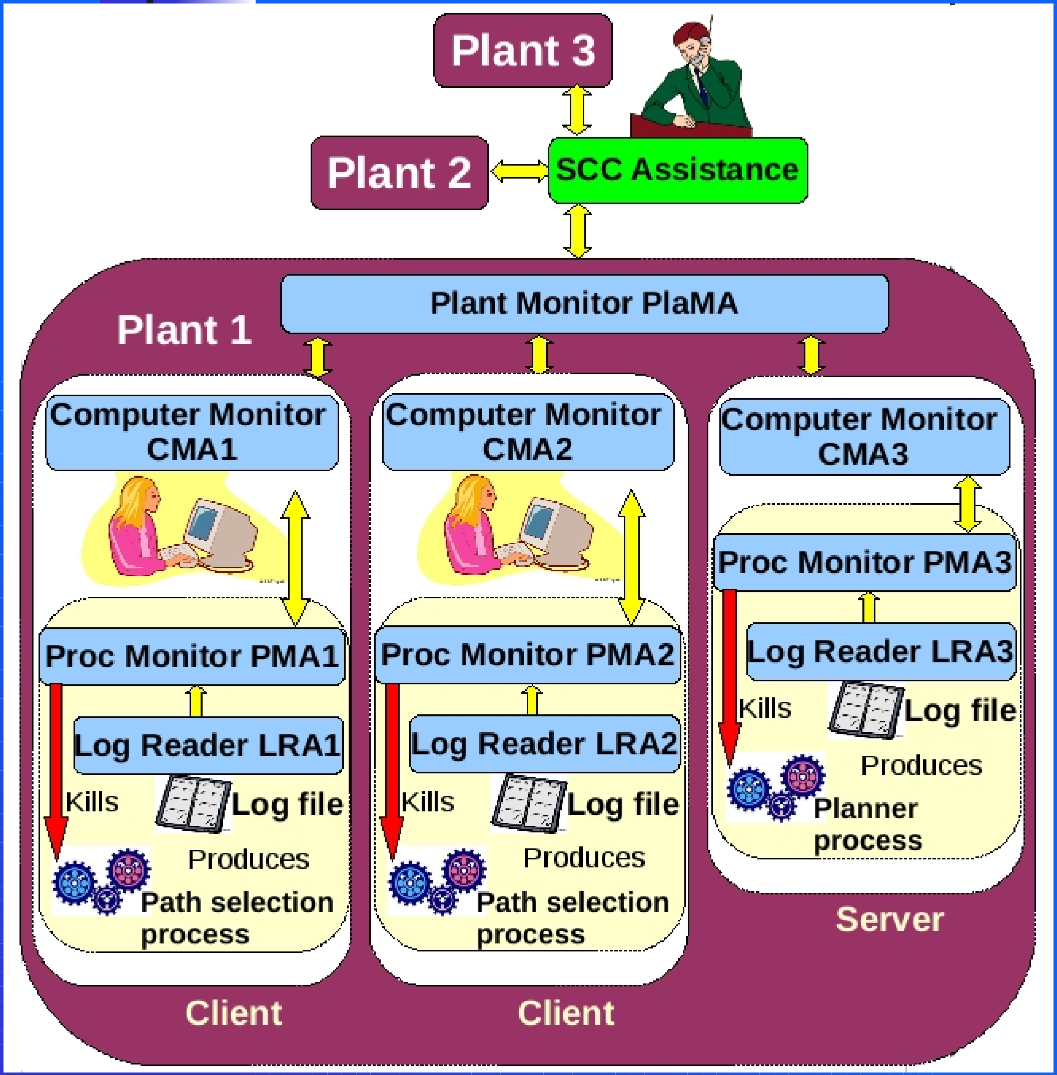
<u>Viviana Mascardi</u>

Daniela Briola
Gabriele Arecco
Maurizio Martelli
Riccardo Caccia
Carlo Milani

The project

- Involved DISI (CS Department of Genova University) and Ansaldo Segnalamento Ferroviario
- Design and implementation of a MAS that:
    - Monitors the Path Selection process running on different machines of the SCC (Command and Control System for Railway Circulation)
    - Uses rules to discover if the process or the network has problems
    - Discovers the problem before it manifests to the user or it worsens, and reports it to the SCC remote assistance
- MAS developed on JADE using DCaseLP libraries, with Prolog based agent to monitor process

System architecture



On every client there is

- A LogReaderAgent associated with each process (LRA)
- A Process MonitoringAgent (PMA) for each process
- One Computer MonitoringAgent (CMA)

On the server there is

- One (the) Plant MonitoringAgent (PlamA)

System architecture

- Every process produces a LogFile
- The LRA can read this logfile
- The LRA sends a message to the PMA for each event in the logfile
- The PMA stores the information and looks for anomalies
  - Can ask the CMA
- The CMA
  - Monitors different processes on a machine
  - Can ask the PlamA

PMA (Process Monitoring Agent)

- Parameters are of two types:
  - Private to the process: not influenced by the state of the network or by other processes
  - Common: influenced by the network or by other processes
- PMA will manage parameters
  - Of type 1) locally
  - Of type 2) asking further information to the CMA
- It is the only agent with the authority to kill a process
- Can ask for more information, not for what to do

CMA (Computer Monitoring Agent)

- CMA will manage different processes and the same parameters of PMAs
- It looks for the same problem from other PMAs
    - If true, depending on the parameter, it can:
        - Answer directly to the PMA
        - ask PLAMA for further information
    - If false: answers the PMA to manage it locally
- CMA will manage the problem depending if it is common to more CMAs or if it has been reported only locally

PLAMA (Plant Monitoring Agent)

- Takes track of all the requests from the CMAs
- Is the point of reference for all the CMAs, the only which knows about the network
- Reports a shared problem to the SCC remote assistance
- Does not decide about how to manage a problem, only reports on its presence

Implementation

- Based on JADE
- LRA: pure JADE agent (JAVA)
- PMA, CMA, PLAMA: Prolog agents integrated into a JADE agent by means of the DCaseLP libraries

Environment model

- Agents live and act in the software Environment consisting of the already existing processes developed by Ansaldo plus the SCC Assistance Centre, and interact with it in a limited way:
    - LRA is the only agent able to get information from the Environment where the MAS is situated.
    - PMA can interact withthe Environment by killing and restarting the process it monitors.
    - PlaMA alerts the SCC Assistance Centre. It interacts with the Environment by alerting the remote assistance centre

Knowledge model

- The parameters managed are:
  - Connection_to_server
  - View, Errors
  - Answer_to_life
  - Cpu_usage, Disk_usage, Memory_usage
- The information about problems is stored as Prolog facts of this form
- “log(time(“Mon Feb 11 21:30:43 CET 2008”), [view(normal), cpu usage(normal), connection to server(active), disk usage(normal), answer to life(slow), errors(absent), memory usage(normal)])”

Agent Architecture

- The architecture of each agent, apart LRA ones, is a declarative architecture where the knowledge base is modeled as a set of Prolog facts, the behavior is determined by Prolog rules, reactivity is implemented by allowing agents to look at their message box and to react to incoming messages.

Behaviour

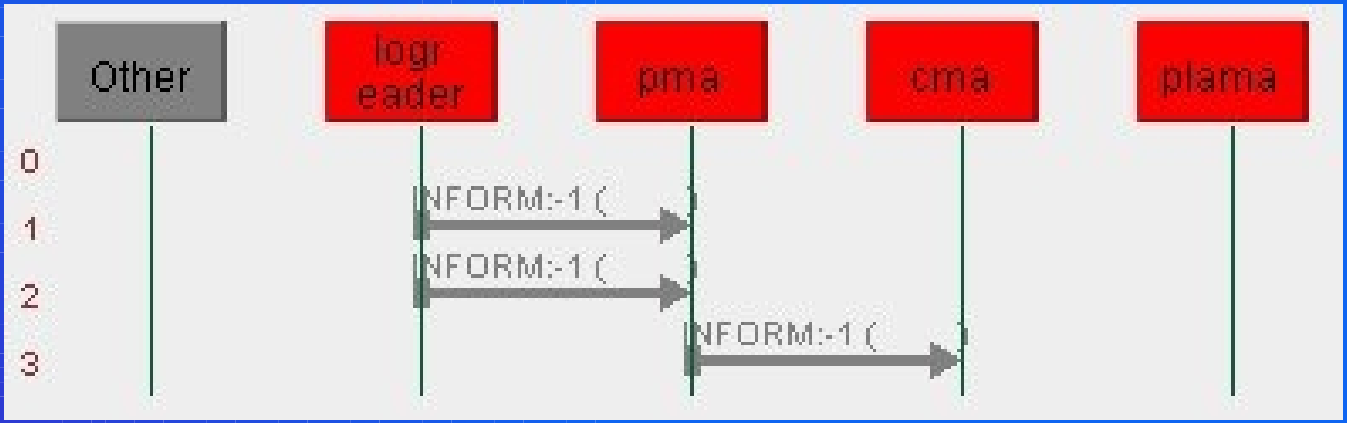
CMA, PMA and PlaMA: cyclic “observe-think-act” behaviour (a “cyclic behaviour” in Jade) where they

- look if a new message matching a given template has been received
- retrieve the message from their message queue and store it in their history
- manage the message according to the rules in their program, and to their knowledge base (that includes all the messages received in the past)
- answer to the agent that has sent the message, and, in case, send messages to other agents in the MAS

Rules Example 1: “cpu usage”

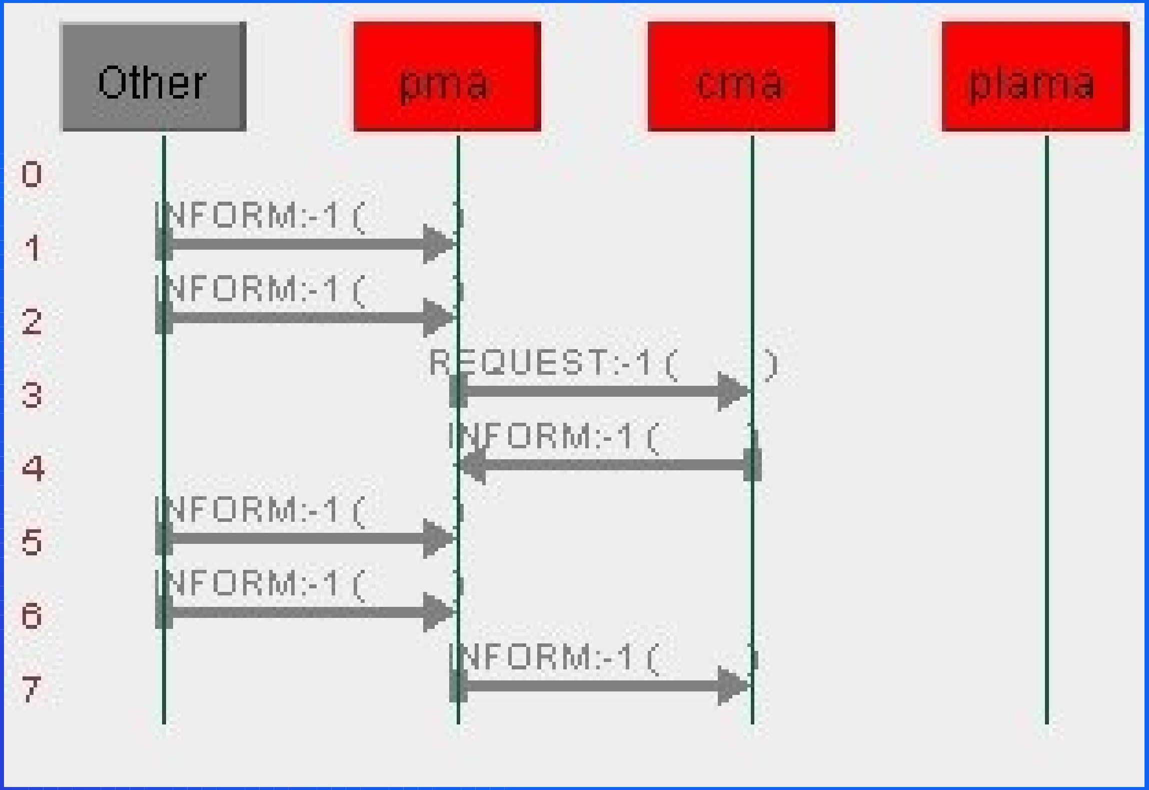
When the PMA receives a message from the LRA:

- 1) If the value is “normal”, no action needs to be taken.
- 2) If the value is “high”, and it remains high in the successive message sent by the LRA, the PMA kills and restarts the process, and informs the CMA.

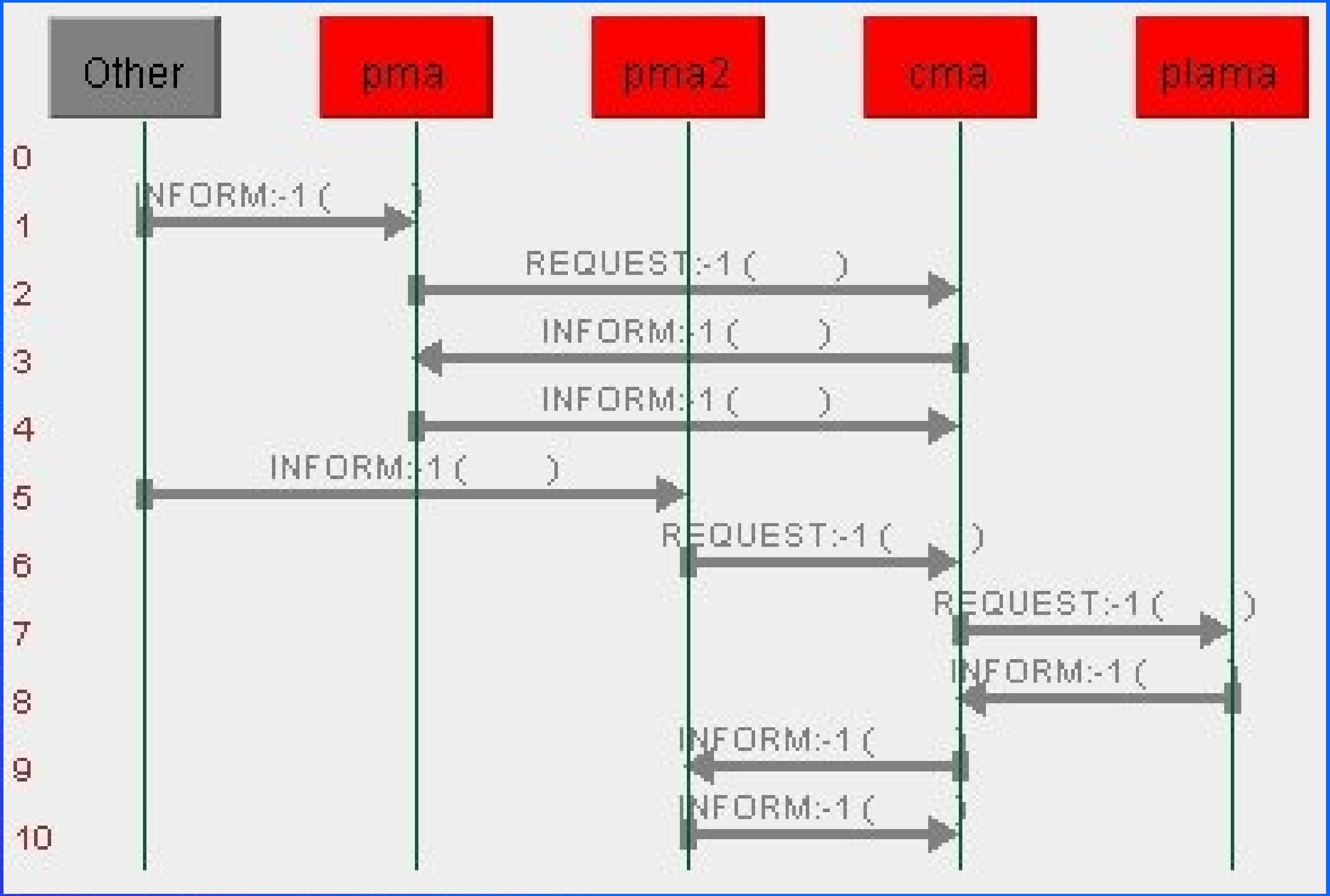


Rules Example 2: “answer to life”

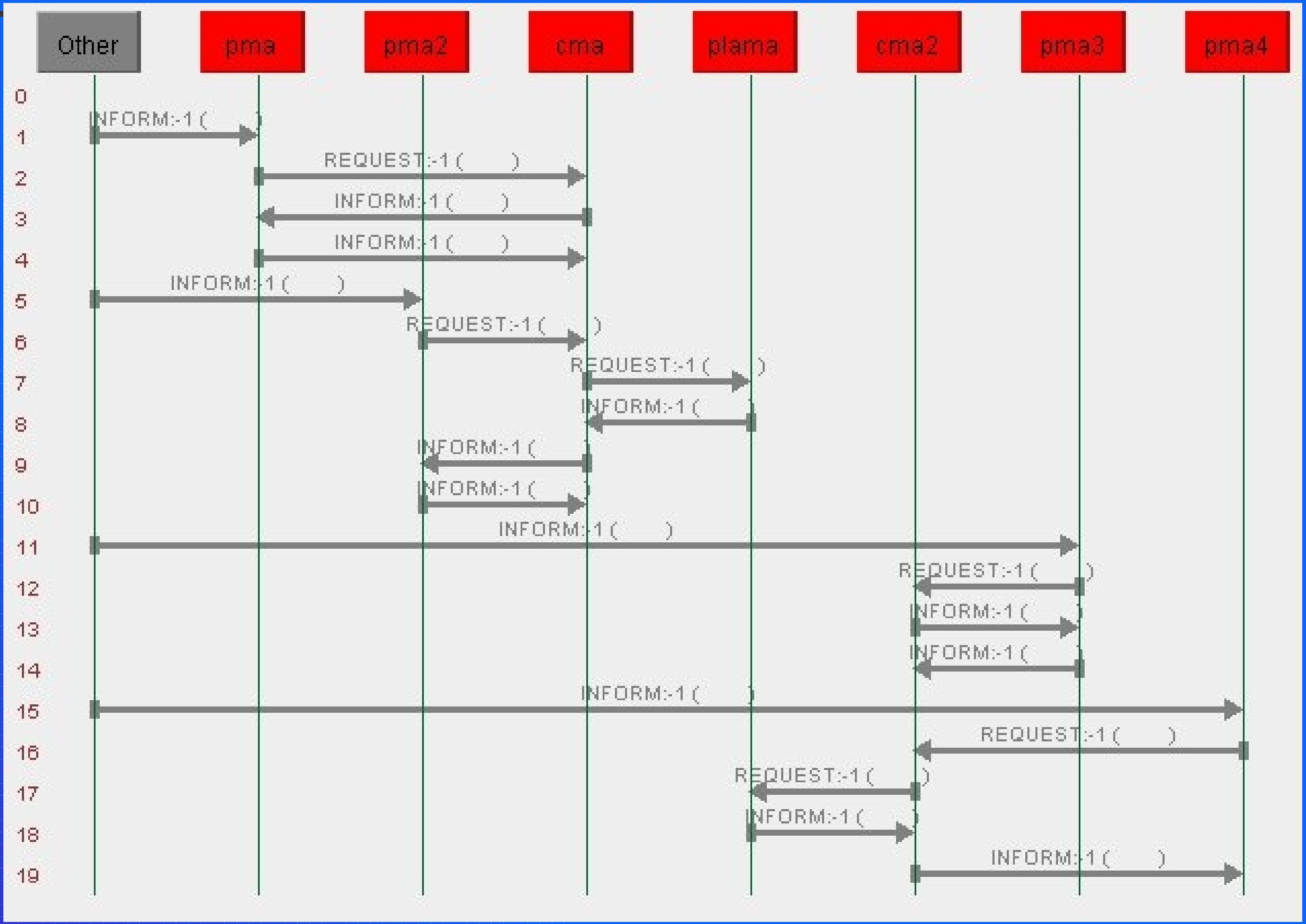
Simple execution, with only one PMA reporting the problem, and only one CMA



Rules Example 3: “connection to server” with a simple configuration



Rules Example 4: “connection to server” with 4 PMA, 2 CMA and the PLAMA



Conclusions

- MAS extensively tested on real log files provided by Ansaldo STS, but off-line.
- Full integration of the MAS into the SCC system still to come; it will require no changes to the existing SCC system.
- The role of academia in providing a good support during the design and implementation of MASs is a key factor in the take-off of the agent technology
- The joint DISI-Ansaldo project represents a success story in making agent technology trusted and accepted by industry.