DECISION SUPPORT FOR POWER SYSTEM CONTROL CENTERS – A MODEL BASED REASONING COMPONENT

Nuno Malheiro[1], Zita Vale[1], Carlos Ramos[1], Manuel Cordeiro[2], Albino Marques[3], Vieira Couto[3]

[1] GECAD – Knowledge Engineering and Decision Support Group
Instituto Superior de Engenharia
Instituto Politécnico do Porto
Porto, Portugal
ntm@dei.isep.ipp.pt

[2] Engineerings Section
University of Trás-os-Montes e Alto Douro
Vila Real ; Portugal
cordeiro@utad.pt

[3] REN – National Electrical Network, S.A.
(EDP Group)
Apartado 3 - 4471 Maia Codex

Abstract – The increasing demand on Power System Control Centers (PSCC) for optimization of the use of the Electrical Network implies, as one of its key focuses, the fastest possible restoration of service. Human PSCC operators (PSCCO) control the Electrical Network aided by a distributed system of sensors and actuators, which produce temporal information about the networks' state. These distributed systems are known as Supervisory Control and Data acquisition or SCADA systems.

In incident situations, in which SCADA systems can produce huge amounts of information, PSCCOs can become very stressed and loose the necessary focus to perform an accurate diagnosis and hence proper power restoration.

Several approaches have been made to develop systems which can cope with the particularly hard features of this problem in real-time. This paper presents a model based reasoning component which is integrated in a multi-component system with the task of Electrical Network diagnosis and restoration. This particular component adds adaptability to the system ensuring correct behaviour on any topological variation of the part of the electrical network in service or its protection schemes. The whole system is fully functional and operating in the Portuguese Electrical Network Dispatch Center.

Keywords: Model Based Reasoning, Temporal Reasoning, Non-Monotonic Reasoning, Incomplete Information, Power Systems

I. INTRODUCTION

Control Operation of Electrical Power Networks is usually performed in Power System Control Centers (PSCC) by very qualified professionals. These professionals, or Power System Control Centre Operators (PSCCO), normally use a SCADA (Supervisory Control and Data Acquisition) system to acquire information about the Electrical Network's state. Control Centre operators possess knowledge about the network's dynamic behaviour, namely its physical laws, its protection devices and their operation and how to combine their knowledge with incoming SCADA information (alarms) to assess the Electrical Network's state, in order to perform fault diagnosis and to plan and execute power restoration.

Many computer applications have tried to solve some of the problems of fault diagnosis [1]. In [2] several systems are presented, with various characteristics, but only the AUSTRAL system, presented in detail in [3] tries to solve a problem similar to the one presented on this paper. Also in [4], another system is presented, developed by the authors of this paper: the SPARSE system, which provides some solutions to the fault diagnosis problem. Reference [5] also presents a possible solution to the fault diagnosis problem. In this case, causal graphs transformed in an operational event calculus form are used to diagnose nuclear power plant problems. Still another approach is presented in [6], which presents a logic-based system for fault diagnosis (including missing information). The referenced systems try to perform fault diagnosis, based on information acquired by sensors in the electrical network (or other physical object of application), they work in soft real-time and, even though there are differences regarding their implementation, every one which works in real-time, presents the same problems in the presence of unexpected situations. A dynamic change in network topology such as the change of a protection signal from one device to another is enough to create additional complexity and increase the possible protection topologies.

The DRUM-II work [7] presents a model based reasoning engine and has been applied to alarm correlation. Work has also been done in Extended Logic Programming to achieve the same goals [8]. Unfortunately, none dealt with temporal real-time issues.

Some of the referenced systems are knowledge based systems and others are model based systems. The knowledge based systems are more accurate and reach conclusions using heuristic search which considerably shortens the possible search tree. The model based approaches search a greater space which can slow them down but increases their robustness.

To design a faster and still robust system, a knowledge-based approach, combined with a model-based

approach has already been proposed and is presented in [9]. This paper focuses on the model based component, which is of vital importance for the ability of the system to perform in sometimes unexpected situations.

II. PARADIGM & ALTERATIONS

In order to comply with the domain's properties, some formalism had to be chosen, which coped with several restrictions, namely:

1. Deal with a metric time representation;
2. Possess temporal and non-monotonic characteristics;
3. Be able to represent both models and knowledge;
4. Work in soft real-time in the desired application.

Several paradigms deal with temporal information, such as the Situation Calculus, Event Calculus, Allen's Interval Calculus, etc. Several other paradigms deal with temporal issues in less flexible form, namely temporal constraints in production rules, recognition of chronicles or temporal causal graphs. Between all these paradigms and focusing on the restrictions, the Event Calculus [10][11] was chosen for development, due to its metric time representation, ability to model knowledge with some extension and the ability to work in soft real-time with an implementation from [12].

A. *Simple Event Calculus with Timeouts*

To perform temporal event correlation, we have developed an extension of the Event Calculus [9] that will be briefly presented here.

The SEC is extended, incorporating special events (the timeout events), becoming simple event calculus with timeouts (SECT). These timeout events are added to the fact base whenever a new event arrives. The pair (E,timeout(E)), in which E is one event and timeout(E) its corresponding timeout, captures the time window in which E is relevant for reasoning. Associated with this time window, a fluent dependent on its event and respective timeout can represent this time window. The reasoning, on its lowest level, will be based on the intersection of these fluents, thus increasing robustness to lack of chronology.

The timeouts added to the Simple Event Calculus allow an easier way to explicitly represent the knowledge relevance of an information particle. The flow of time is hence indirectly represented.

SECT's axioms are the following:

$$\text{holdsAt}(F,T) \leftarrow \text{initiallyP}(F) \wedge \neg\text{clipped}(0,F,T) \quad \textbf{(SECT1)}$$

$$\text{holdsAt}(F,T2) \leftarrow \text{happens}(A,T1) \wedge \text{initiates}(A,F,T1) \wedge T1 < T2 \wedge \neg\text{clipped}(T1,F,T2) \wedge \neg\text{timedout}(T1,A,F,T2) \quad \textbf{(SECT2)}$$

$$\text{clipped}(T1,F,T2) \leftrightarrow \exists A, T\ [\text{happens}(A,T) \wedge T1 < T < T2 \wedge \text{terminates}(A,F,T)] \quad \textbf{(SECT3)}$$

$$\text{timedout}(T1,A,F,T2) \leftrightarrow \text{isTimedout}(A,F) \wedge \exists T\ [\text{happens}(\text{timeout}(A),T) \wedge T1 < T < T2] \quad \textbf{(SECT4)}$$

The ontology of the items undefined by the Event Calculus are: *isTimedout* definition is: *isTimedout*(**A**, **F**) means that action A is able to time out fluent F; *timedout* definition is: *timedout*(**T1**, **A**, **F**, **T2**) tries to prove that action **A** is timed out between time points **T1** and **T2**, hence timing out fluent **F**.

B. *Definition of Event Calculus Schemas*

To limit the number of explicit fluent initiation and termination formulae, the concept of initiation and termination formula schema will be introduced.

In our case, two parameters are added to both the *initiates* and *terminates* formulae: the dependency parameter **D** and the grounder parameter **G**. The complete initiation and termination formulae become (Sc1) and (Sc2).

$$\text{initiates}(A, D, G, F, T) \quad \textbf{(Sc1)}$$

$$\text{terminates}(A, D, G, F, T) \quad \textbf{(Sc2)}$$

The dependency parameter **D** is required by the implementation described in [10], in order to propagate any change a Maximal Validity Interval (MVI) of a fluent to any other MVI, in its dependence. An MVI is defined as the longest temporal interval possible in which a fluent is true and is not terminated.

The grounder parameter **G** is an arbitrary predicate logic formula (may contain free variables), which produces different formulae for all its ground substitutions. An initiation (termination) formula schema looks like a simple formula, the only difference being that its components, namely **F**, **D** and **G** are arbitrary predicate logic formulae and will be different for each ground substitution of the grounder argument. No variables other than the ones which appear in the grounder argument are allowed in the **F** and **D** arguments.

Notice that being (Sc1) and (Sc2) the heads of a predicate logic formula, their body can exist and has to be proven before the grounded formula is considered.

III. A MODEL OF THE ELECTRICAL POWER NETWORK

Based on the presented paradigm, more specifically on event calculus schemas, as well as topological knowledge representation, a model of the Electrical Power Network for fault detection will be presented. This model will be able to identify fault areas (where lines are disconnected) and remaining interconnectivity (identifying separated islands).

A. Topological Representation

Since our primary concern is fault detection, we shall focus on the presence (or absence) of voltage in each of the network's components. Voltage will be hence forth also be referred to as effort. Several simplifications can be taken into account such as the view of transformers as isolators, generators as ideal sources and absence of resistance in every component which is not a load to the network (client). There is no error introduction through these simplifications since only the effort propagation is of concern in this topological analysis.

Notice that our model of the network will be a form to incorporate the network's topology in the observed findings (SCADA alarms). Since SCADA alarm information is limited to its producing component, global view of the effects of each alarm on the network can be performed with a model which represents the network's topology and is able to propagate the effects of an alarm in one component, to others which are connected to it. This kind of model increases robustness to information incompleteness.

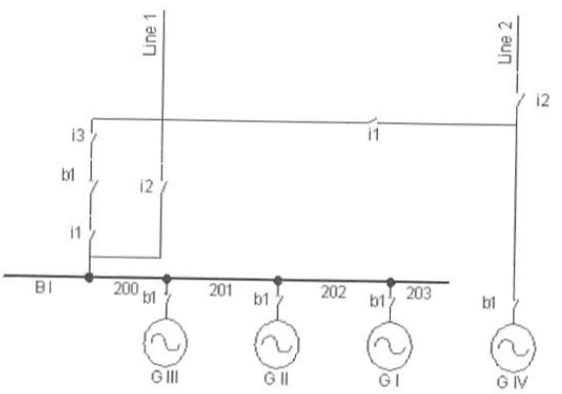


Figure 1 - Schematics of a Simplified Power Plant

Each model component has the following structure: *component(* **Type**, **Identification**, **Connections**), in which the component types are: generators, loads, buses, lines, isolators and circuit breakers. We categorize components in active (producing SCADA messages): generators, breakers and isolators; and passive: loads, lines and buses.

The component's identification is the one assumed in the SCADA alarms and the connections indicate the list of other components to which each component is attached to. This mimics the networks topology and can represent, for instance, a power plant as it is depicted in Figure 1.

The option for this type of paradigms for electrical network representation, disregarding their numerical simulation counterparts was due to the need of assuring the model's non-monotonic behaviour, for instance in the presence of alteration of alarm chronology.

Notice that the task of the model based reasoning component is to map the alarms of the network and use the model's knowledge to verify unexpected behaviour. Unexpected behaviour is usually associated with a malfunction in some device in the power network or in some device in the SCADA network.

The whole network is represented in the form of component; components are associated to create larger order components like panels, protected lines, installations or islands.

B. Alarm Based Effort Propagation

The propagation of effort (voltage, in our case) is viewed as occurring instantaneously. A generator is seen as an ideal source, which imposes effort on each and every component to which it has a connection path. All active components allow effort to propagate if they are "in service" and all passive components always allow effort propagation.

The model created to propagate the effort in the network is presented by (TP1) through (TP4), in an operational PROLOG form. It is modeled using the Event Calculus and no timeout information.

```
initiates( scada(generator,ID,on), [],                    (TP1)
 component(generator, ID, _),
 effort(generator, ID), T ).

terminates( scada(generator,ID,off), [],                  (TP2)
 component(generator, ID, _),
 effort(generator, ID), T ).

initiates( scada(Type,ID,on),                             (TP3)
 [effort(_,_), inService(_,_)],
 (connectedWithoutEffort( Type,ID,OType,OID,T),
 OType \== generator ), effort(OType, OID), T ):-
       effortLink( OType, OID, T ).

terminates( scada(Type,ID,off),                           (TP4)
 [effort(_,_), inService(_,_)],
 (connectedWithEffort(Type,ID,OType,OID,T),
 OType \== generator ),
 effort(OType, OID), T ):-
       not( effortLink( OType, OID, T ) ).
```

The explanation of the model is as follows. (TP1) imposes effort on a generator if the SCADA system states that this generator is active. (TP2) terminates effort on the generator if it is inactive. (TP3) propagates the effort of any component which just got under effort. It states that any component which has just entered in service can propagate effort to any other component, which is connected to it, which is not in effort and which became connected to an effort source, through the entry in service of the component which produced the alarm. (TP4) represents the possible effort termination in a component connected to one which is disconnected. It states that effort is terminated in a component which was in effort and which was connected to one which left service and for which an effort link form any source cannot be established.

Notice that propagation is limited ensuring a faster model. Propagation stops whenever change in the network ceases to occur. Nevertheless, this approach is

computationally expensive and can provide no answer in limited time. In [12] a complexity analysis is also performed and, even though the particular features of this application domain could improve the worst case complexity, the absence of an answer in a time window useful to the PSCCO might be possible in an incident situation.

C. *Higher Order Components*

In order to provide meaningful decision support, the system must be made to show features similar to those included in the mental processes of a PSCCO. It is also easier to define the behaviour of components in a top-down approach, initially defining the behaviour of larger components and afterwards map this behaviour in the component's internal subcomponents. On the other hand, subcomponent behaviour can also alter higher order component behaviour.

The complex component definition is such as: *complexComponent(* **Type**, **Identification**, **ComponentSet**). These complex components are of a certain type, which can be a panel, installation, etc. Each component has a unique identification and these complex components are defined by their component set. The component set is the set of simpler components which constitute the more elaborate component. Any connections to the outside are defined by the lower order components. This introduces hierarchy in the system, enabling the definition of status and behaviours in simple and complex components thus simplifying knowledge acquisition and maintenance.

D. *Component Behaviour*

The component behaviours are defined for each component as: *componentBehaviour(* **Type**, **Identification**, **State**, **Behaviour**). The component behaviour can be assigned to a type of components or a particular component. In either case, a state will be defined by a particular set of fluent values. If the state matches, the behaviour is triggered. This mechanism will be used to mimic the network, thus ensuring a predictive model of the system. Behaviours can be as simple as "inform the user" and as complex as any automatic device in the electrical network. This will also allow some proactiveness in making inferred conclusions or contradictions available to other modules of the system.

Since the model based reasoning is temporal, some special behaviour must be defined for future situations. When some component performs some behaviour, a sub product of this can be an expected behaviour from a fellow component in response. This expected behaviour is placed in standby and will try to be fulfilled before a certain amount of time. This is defined as *expectedBehaviour(* **Identification**, **State**, **Behaviour**, **Timeout**, **TriggerTrue**, **TriggerFalse**) being specified for a component with a particular identification, the state which triggers the expectation, the expected behaviour, a timeout before which it will active and two triggers: one if the expected behaviour is detected and another if it is not.

Any expected behaviour which is not fulfilled is considered a contradiction by the system. This can be solved by the user

IV. Model Based Reasoning

The complete system works according to Algorithm 1. It is data driven and the reception of new information does not block so that temporally dependant behaviours can exist such as the timeout of expected behaviours.

```
ModelBasedReasoning()
        A:=getNewAlarm();
        integrateNewAlarm( A, AF);
        WHILE AF≠ϕ
                C:=Select( AF );
                AF:=AF\{C};
                handleBehaviours( C );
        END WHILE
        verifyExpectedBehaviours();
```

Algorithm 1 – Model Based Reasoning

```
handleBehaviours( C )
        BL:=getBehaviours(C);
        WHLE BL≠ϕ
                B:=getNewBehaviour(BL);
                BL:=BL\{B};
                IF checkState(B) THEN
                        TriggerBehaviour( B )
                END IF;
        END WHILE;
```

Algorithm 2– Behaviour Handling for component C

```
verifyExpectedBehaviours();
        getExpectedBehaviours( EBL );
        WHLE EBL≠ϕ
                EB:=getNewExpectedBeha(EBL);
                EBL:=EBL\{B};
                IF timeout(EB) THEN
                        TriggerFalse(EB);
                        ELSE TriggerTrue(EB);
                END IF;
        END WHILE;
```

Algorithm 3– Verification of Expected Behaviours

In Algorithm 1 the integration of a new alarm will produce a list of components which fluents were affected in the process. The procedure *integrateNewAlarm(*A,AF) is the implementation of the SECT. New alarms are integrated in the knowledge base and all its consequences are derived, generating **AF** or the affected fluents list. For these components, behaviour must be checked. In Algorithm 2 the handling of component behaviours is performed. In Algorithm 3 the verification of the expected behaviours is performed.

V. Incomplete Information Example

To better understand the use of this system in an incomplete information environment, an example will be presented. Initially, a complete information example will be provided, followed by some considerations on missing information.

Figure 2 shows the initial state of a very simplified power line L1, fed by generator G1 and feeding load Load1. There are two protective devices: circuit breakers B1 and B2. In the initial state, all components are under voltage.

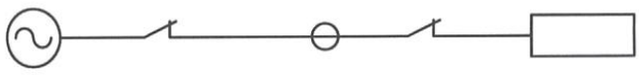


Figure 2 – Initial State

This scheme will be added to the model in the way Program 1 depicts.

```
component(generator, 'G1',[(breaker, 'B1')]).
component(breaker,'B1',[(generator,'G1'),(line,'L1')]).
component(line,'L1',[(breaker, 'B1'), (breaker,'B2') ]).
component(breaker,'B2',[(load, 'Load1'), (line,'L1') ]).
```

Program 1– Electrical Network Definition

```
componentBehaviour( line, L, not(effort(line,L)),
 informUser(lineDown(L)) ).

componentExpectedBehaviour( (breaker,'B1'), 1000,
 deviceTripping( (breaker, 'B1') ),
 (  deviceTripping( (breaker, 'B2') );
    deviceOpen( (breaker, 'B2') );
    deviceClosed( (breaker, 'B1') )      ),
 true,
 informUser( expectedTrip( (breaker, 'B2') ) ) ).

componentExpectedBehaviour( (breaker, 'B2'), 1000,
 deviceTripping( (breaker, 'B2') ),
 (  deviceTripping( (breaker, 'B1') );
    deviceOpen( (breaker, 'B1') );
    deviceClosed( (breaker, 'B2') )      ),
 true,
 informUser( expectedTrip( (breaker, 'B1') ) ) ).
```

Program 2– Component Behaviour Definition

In Program 3, the SCADA alarms are transformed into fluents within the system. A tripping alarm will start a tripping fluent, while a closure alarm will terminate it. Nevertheless, there is a timeout to the tripping fluent, stating that its persistence is temporary.

Program 2 shows the definition of component behaviour. The first is the behaviour of any line without effort, which should proactively inform someone. The second and third clauses represent a synchronization device, which forces a breaker on one extreme of a line to trip or open when the breaker on the other extreme has already tripped. This is a line protection mechanism. This mechanism will not be triggered for fugitive faults, hence if the circuit breaker which tripped closes due to a fast restoration mechanism, the whole process is discarded.

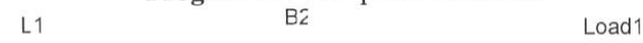
```
initiates( scada(Type,ID,trip), [], true,
 deviceTripping(Type, ID), T ).

terminates( scada(Type,ID,off), [], true,
 deviceTripping(Type, ID), T ).

isTimedOut(scada(Type,ID,trip),
 deviceTripping(Type, ID) ).
```

Program 3– SECT Model Definition

The simplified alarm list for a simple trip is shown in Program 4. The timeouts are added automatically to the list by the system. To further simplify consider rimes in milliseconds and the first alarm in instant 0.

```
happens( scada( breaker, 'B2', trip ), 0 ).
happens( timeout(scada( breaker, 'B2', trip )), 300 ).
happens( scada( breaker, 'B2', off ), 200 ).
happens( scada( breaker, 'B1', trip ), 900 ).
happens( timeout(scada( breaker, 'B1', trip )),1200).
happens( scada( breaker, 'B1', off ), 1000 ).
```

Program 4 – Complete Alarm Set



To graphically show the occurrence of the alarms, Figure 3 shown the line at instant 200 when the line protection synchronism occurs and Figure 4 shown the system after its occurrence at instant 1000.

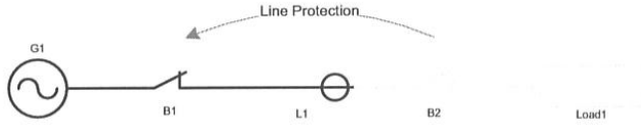


Figure 3 – State at instant 200

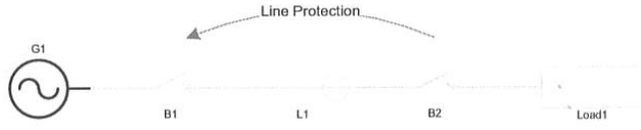


Figure 4 – State at instant 1000

Notice that at instant 0, the behaviour of breaker B1 triggers the expected behaviour of breaker B2. In this case, the expected behaviour will be satisfied before the timeout occurs. In the absence of this signal, the user would be informed that an expectation has failed.

In both cases, the presence or absence of information, the PSCCO would be informed of an altered network state.

Notice that either signal (trip or opening) will fulfil the expected behaviour. Only in the absence or both of them will the missing expected behaviour be reported. This and all other modelling issues were gathered from the PSCCOs and coded in the system.

Notice the behaviour to the incomplete alarm set presented in Program 5. In this case, circuit breaker **B2** trips and for some reason no response is obtained from the other extreme of line **L1**. There are several possible causes for this occurrence:

1. the protection mechanism at B2 did not emit the synchronization signal to B1 or
2. the protection mechanism at B1 did not receive the synchronization signal or
3. there is a malfunction in device B1 which does not allow it to comply or
4. there is a malfunction in information acquisition/transmission from device B1.

In each of these cases, information must be gathered from the environment, sending a team to the local to verify the failure. Nevertheless, the operator must be warned that a problem has occurred. It can additionally be shown expected procedure in these situations.

```
happens( scada( breaker, 'B2', trip ), 0 ).

happens( timeout(scada( breaker, 'B2', trip )), 300 ).

happens( scada( breaker, 'B2', off ), 200 ).
```

Program 5 – Incomplete Alarm Set

This is a small example of the system's potential to both model the domain and additionally reason within that domain. Information incompleteness is dealt with and can originate validation procedures to cope with the inherent uncertainty.

VI. Conclusions

The architecture of a model based reasoning component for fault diagnosis with incomplete or incoherent temporal information, applied to incident analysis in Electrical Power Networks has been presented. It is based on an altered for of the Event Calculus, thus ensuring temporal reasoning.

The model-based approach produces, in fact, an alternative to a knowledge-based system in altered conditions. When a protection scheme is dynamically altered and an incident occurs, a knowledge-based application could not adapt to new circumstances in real-time but the model-based approach is more robust, being able to handle these cases. The model based approach on its own is not able to provide higher level information about the system without increasing its computational time to a point when decision support may not be useful, so the set of both approaches is more effective than each of the separate approaches. False alarms or alarms which were produced by protective device malfunction can also be detected with this approach.

A prototype implementation is already at work in the Control Centre of the Portuguese Transmission Network. The system was tested and validated. It is important to refer that its results are more reliable than those of SPARSE, its predecessor, in cases where events have chronological problems, or some incompleteness like one missing event, being equally good in cases without these disturbances.

Due to its solid theoretical background, the verification of such a system can further benefit of advances made by the scientific community in the verification of systems using the Event Calculus.

Acknowledgment

The authors gratefully acknowledge the contributions of Eng. Amarante dos Santos, Eng. Rui Pestana and Eng. Nuno Salema from REN (EDP) for their inestimable commitment to this project.

References

[1]Z. Vale, "Knowledge Based Techniques and Applications in Power Systems Control Centers" in *Intelligent Systems – Technology and Applications*, vol. VI, Cornelius T. Leondes (Editor), CRC Press, pp. VI-63 to VI-112, 2003

[2]ALARM research group, "Monitoring and Alarm Interpretation in Industrial Environments", AI Communications vol.11(3,4), pp. 139-173, IOS Press, 1998

[3]J.-P. Krivine and O. Jehl, "The AUSTRAL system for diagnosis and power restoration: an overview", in *Proc. International Conference on Intelligent Systems Application to Power Systems (ISAP'96)*, Orlando (USA), August 1996

[4]Z.A. Vale, A. Machado e Moura, M.F. Fernandes, A. Marques, C. Rosado, C. Ramos "SPARSE: An Intelligent Alarm Processor and Operator Assistant", IEEE Expert, 12(3), Special Track on AI Applications in the Electric Power Industry, 1997, pp. 86-93

[5]I. Grosclaude, M-O. Cordier and R. Quiniou, "Causal interaction: from a high-level representation to an operational event-based representation", in *Proc. International Joint Conf. on Artificial Intelligence (IJCAI 2001)*, Washington (USA), August 2001, pp. 565-572

[6]J. Jung, M. Hong, C-C Liu, G. Tornielli, A. Di Stefano, M. Gallanti and D. Maratukulam, "Logic and validation techniques for handling of missing information in fault diagnosis", Int. J. Engineering Intelligent Systems, Dec. 2001, pp. 213-217

[7]P. Froelich and W. Nejdl, "A static model-based engine for model-based reasoning", in *Proceedings of the Fifteenth International Joint Conference on Artifical Intelligence (IJCAI-97)*, pp. 446--471, Nagoya, Japan, (1997)

[8]P. Froehlich, W. Nejdl, M. Schroeder, C. V. Damásio and L. M. Pereira, "Using Extended Logic Programming for Alarm-Correlation in Cellular Phone Networks", Int. Journal of Applied Intelligence 17(2), pp. 187-202, Kluwer Academic Press, 2002.

[9]N. Malheiro, Z. Vale, C. Ramos, M. Cordeiro, A. Gomes, A. Marques and V. Couto "Fault diagnosis with incomplete and temporal information – an application using a knowledge and model-based architecture", *Int. Conf. on Intelligent Systems Application to Power Systems (ISAP 2003)*, Lemnos, Grécia, 31 de Agosto a 3 de Setembro, 2003

[10]R.A.Kowalski and M.J.Sergot, "A Logic-Based Calculus of Events", New Generation Computing, n. 4, 1986, pp. 67–95.

[11]M. Shanahan, "The event calculus explained", in M.Wooldridge and M. Veloso (eds), Artificial Intelligence Today, vol. 1600 of LNCS, Springer-Verlag, 1999, pp. 409-430

[12]L.Chittaro and A.Montanari, "Efficient Temporal Reasoning in the Cached Event Calculus", Computational Intelligence, vol.12(3), 1996, pp. 359-382

[13]A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R.t Manchek and V. Sunderam, "PVM: Parallel Virtual Machine - A Users' Guide and Tutorial for Networked Parallel Computing", Scientific and Engineering Computation, MIT Press 1994