Verification of Knowledge Based-Systems For Power System Control Centres

Jorge Santos[1], Luiz Faria[1], Carlos Ramos[1], Zita A. Vale[2], Albino Marques[3]

[1] Polytechnic Institute of Porto, Institute of Engineering
Department of Computer Engineering
Rua de S. Tomé, 4200 Porto, Portugal
{jsantos | lff | csr}@ dei.isep.ipp.pt

[2] Polytechnic Institute of Porto, Institute of Engineering
Department of Electrical Engineering
Rua de S. Tomé, 4200 Porto, Portugal
zav@dee.isep.ipp.pt

[3]REN – Portuguese Transmission Network, EDP Group
Apartado 3, 4471 Maia Codex, Portugal

Abstract. During the last years, electrical utilities began to install intelligent applications in order to assist Control Centres operators. The Verification and Validation (V&V) process intends to assure the reliability of these applications, even under incident conditions.
This paper addresses the Validation and Verification of Knowledge-Based Systems (KBS) in general, focussing particularly on the V&V of SPARSE, a KBS used in the Portuguese Transmission Network for operator assistance in incident analysis and power restoration.
VERITAS is a verification tool developed to verify SPARSE Knowledge Base. This tool performs knowledge base structural analysis allowing knowledge anomalies detection.

Introduction

Nowadays, Control Centres (CC) are of high importance for the operation of electrical networks. These Centres receive real-time information about the state of the network and Control Centre operators must take decisions according to this information.

Under incident conditions, a huge volume of information may arrive to these Centres, making its correct and efficient interpretation by a human operator almost impossible. In order to solve this problem, some years ago, electrical utilities began to install intelligent applications in their Control Centres. These applications are usually Knowledge-Based Systems (KBS) and are mainly intended to provide operators with assistance, especially in critical situations.

The correct and efficient performance of such applications must be guarantied through Verification and Validation (V&V). V&V of KBS are not so usual as desir-

able but are usually undertaken in a non-systematic way. The systematic use of formal V&V techniques is a key for making end-users more confident about KBS, especially when critical applications are considered.

This paper addresses the Validation and Verification of Knowledge-Based Systems in general, focussing particularly on the V&V of SPARSE, a KBS to assist operators of Portuguese Transmission Control Centres in incident analysis and power restoration.

It is known that knowledge maintenance is an essential issue for the success of a KBS but it must be guaranteed that the modified KB remains consistent and will not make the KBS incorrect or inefficient. There is no general agreement on the meaning of these terms. For the remaining of this paper, the following definitions will be used:

- Validation - Allows to assure that the KBS provides solutions that present a confidence level as high as the ones provided by the expert(s). Validation is then based on tests, desirably in the real environment and under real circumstances. During these tests, the KBS is considered as a "black box" and only the input and the output are really considered important.
- Verification - Allows to assure that the KBS has been correctly conceived and implemented and does not contain technical errors. Verification is intended to examine the interior of the KBS and find any possible errors.

Most KBS are only validated and verified. Although validation process can guarantee that when the system is deployed, its performance is correct, the existing problems may arise when there is a need to change the Rule Base.

Verification should rely on formal methods requiring the development of tools to implement these methods. Although there are already some available verification tools in the market, specific needs of Power System applications usually require the development of specific tools. As formal methods of verification rely on mathematical foundations, they are able to detect a large number of possible problems. In this way, it is possible to guarantee that a KBS that has passed through a verification phase is correct and efficient. Moreover, it is possible to assure that it will provide correct performance with examples that have not been considered in the validation phase.

The present section focus the mains aspects related with KBS knowledge maintenance, stressing its relation with V&V stages.

Section 2 describes the SPARSE's characteristics, namely, architecture, reasoning model, rule selection mechanism and its implications for Verification and Validation work.

Section 3 describes the Validation stage of SPARSE development, especially the field tests and the need of applying formal methods in SPARSE's V&V.

Section 4 presents VERITAS, a verification tool based on formal methods. This tool has been successfully applied to several KBS: SPARSE; ARCA, an expert system applied to Cardiology diseases diagnosis; and another expert system created to assist in Otology diseases diagnosis and therapy. Finally, section 5 presents some conclusions and future work.

SPARSE

SPARSE is a KBS developed for the Control Centres of the Portuguese Transmission network, owned and operated by REN[1]. This KBS assists Control Centres operators in incident analysis and power restoration [7] [8] [9].

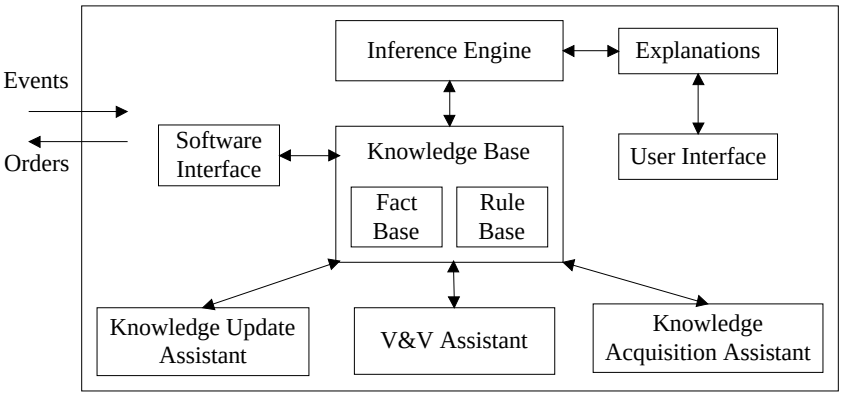

Fig. 1 - SPARSE Architecture

SPARSE (see: Fig. 1) has been developed using PROLOG and C language and runs on-line in a DECstation 5000/240 under ULTRIX operating system. This machine is connected, through a Local Area Network Ethernet of duplicate configuration, with two MicroVAX II machines that support SCADA (Supervisory Control And Data Acquisition) functions in the Control Centre.

SPARSE presents some features that make the verification work more difficult than for most KBS. These features include nonmonotonic behavior, temporal reasoning and the meta-rules used in rule triggering. Considering the following rule:

```
rule xx : 'EXAMPLE' :
[
 [C1 and C2 and C3]
 or
 [C4 and C5]
]
==>
[A1,A2].
```

The conditions considered in the LHS (Left Hand Side) (C1 to C5 in this example) may be of one of the following types:

- A fact which truth must be proved (normally these facts are time-tagged);
- A temporal condition;

The actions/conclusions to be taken (A1 to A2 in this example) may be of one of the following types:

- Assertion of facts (conclusions to be inserted to the knowledge base);

[1] REN is the Portuguese Transmission Network

– Retraction of facts (conclusions to be deleted from the knowledge base);
– Interaction with the user interface.

The rule selection mechanism uses facts with the following structure:

```
trigger(NF,NR,T1,T2)
```

This fact means that rule number NR should be triggered, until it is successful, between instant T1 and T2, because of the arrived fact NF.

SPARSE has passed through a validation phase and is presently installed in one of the two Control Centres of REN - Vermoim Control Centre, providing real-time assistance to operators.

SPARSE's Validation and Verification have been especially important for the success of this project, namely in what concerns knowledge updating.

Validation

The process of Verification and Validation should start as early as possible during the development of the application. The SPARSE V&V have been considered since the very beginning and special arrangements have been made in order to provide conditions for this process performing.

The project team aimed to perform the validation of SPARSE using examples as close as possible to the ones that the application should face in the real environment. According to this, it was considered that validation should be based mainly on real information about the network.

Another important aspect that has been considered since an early stage of development is the software required to interface SPARSE with SCADA applications used in the Control Centre. In fact, it was realised that some limitations imposed by SCADA should be considered since the very beginning in order to allow to take them into account during the development of the prototype, namely during the knowledge acquisition phase.

When integration issues are not addressed in an early phase of the project, the changes that are required when the system is integrated in the real environment may be very significant and impose almost a complete rebuilding of the system. The experts should namely, consider these issues during the knowledge acquisition phase.

REN's staff developed an application named TTLOGW [5] to acquire real-time information from SCADA and to send it to SPARSE. It acquires information related to the state of electrical network equipment, which is used to generate material for SPARSE's validation.

This application acquires the information related to the state of the equipment of the electrical network.

In this way, files concerning real incidents have been obtained and have been used in order to validate SPARSE conclusions. Experts involved in the project commented these conclusions and corrections in the Knowledge Base were made whenever necessary.

New validation techniques need to be applied after SPARSE was first installed in the control centre, since it now received real-time information from TTLOGW. The validation of SPARSE considering real-time information was very important due to several reasons:

- Temporal reasoning should be tested under real situations in order to assure its correction;
- Consideration of multiple faults is an important aspect of SPARSE performance that is very dependent from the way information flows;
- Processing times should be tested in order to guarantee real-time performance, even under incident conditions.

As nowadays electrical networks are very reliable it was not possible to completely validate SPARSE with real incidents. A large number of different types of incidents had to be simulated to allow validation. As this simulation should be as accurate as possible, two different techniques have been used:

(1) Simulation of incidents by operators located in chosen substations

(2) Simulation of incidents using a programmable impulse generator and a Remote Terminal Unit (RTU).

These two techniques complement each other, allowing a complete validation.

The simulation of incidents by operators allowed to obtain real-time information that was forced to be generated but presenting exactly the same characteristics as the information obtained during a real incident. During these tests, operators, making the whole system act as if a real incident was taking place simulated the behaviour of the protection equipment. In this way, the information, used by SPARSE was generated, as it would be under a real incident.

Due to the difficulties of co-ordinating operators in several substations, the simulation is not always correct and the whole process may have to be repeated several times in order to obtain a good test case.

In spite of all the difficulties and costs involved, this kind of tests has been considered absolutely essential for the validation of SPARSE, allowing to increase the confidence in its real-time behaviour.

In order to undertake a complete set of tests without the extremely high costs required by this technique, a different technique of test has also been used. This technique involves the use of a Remote Terminal Unit (RTU) and of a programmable impulse generator (PIG). The PIG generates impulses in order to force the alarm messages creation by the SCADA system. This technique was used to simulate a wide set of incidents allowing a more complete SPARSE Knowledge Base validation with reduced costs.

These methods of validation have been considered sufficient to put SPARSE in service, without the need to undertake formal verification of SPARSE Knowledge Base. However, when a Knowledge-Based System, as SPARSE, is in continuous use, the necessity to make changes in the Rule Base arises sooner or later. In the case of the Portuguese Transmission network, the introduction of new substations, with different types of operation or layout, has already imposed some modifications. Under these circumstances, it is not possible to accept the need to undertake complete validation tests, as the ones described before. Even if the costs are acceptable, the required time

would oblige the Knowledge-Based System to be either out of service or to be in service without a validated Rule Base for longer than desirable. This problem must be addressed with a verification tool using formal methods. The use of this kind of tools to detect possible problems in the modified Rule Base allows to reduce the time required in Verification and validation process.

VERITAS, a verification tool

In what concerns SPARSE, there were two major reasons to start the verification work. First, the SPARSE team carried out a set of tests (see section Validation) in order to assure the quality of the answers of SPARSE to a set of real and simulated cases. Considering the expected high reliability and confidence of the tools to be applied in power systems area, it was decided to develop a verification tool to perform anomaly detection in SPARSE KB, assuring the consistency of the represented knowledge. On the other hand, tests applied in the Validation phase, namely the field tests, are very expensive because during it was necessary to assign a lot of technical personnel and physical resources for their execution (e.g. transmission lines). It seems obvious that it is impossible to carry out those tests after each knowledge updating so the developed verification tool offers an easy and inexpensive way to assure the knowledge quality maintenance.

A specific tool, named VERITAS [6] (see: Fig. 2) has been developed to be used in the verification of the SPARSE, performing structural analysis allowing to detect knowledge anomalies.

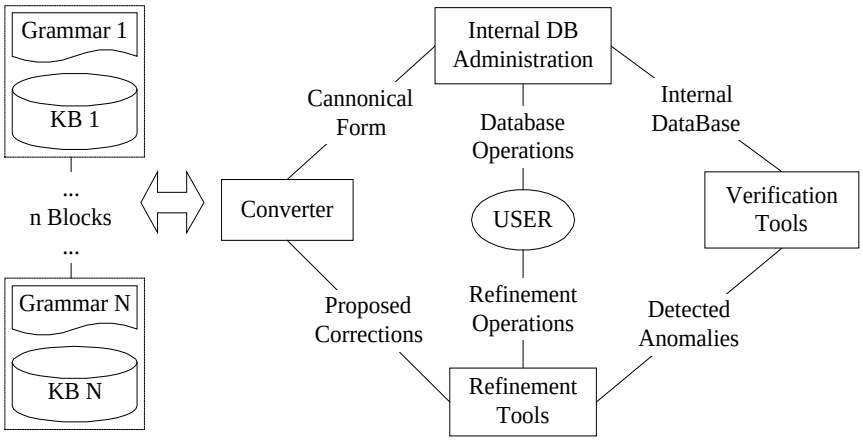

Fig. 2 - VERITAS Architecture

VERITAS is knowledge-domain and rule-grammar independent. It has been developed with an open and modular architecture (Fig. 2) allowing user-interaction along all the verification process. Since the tool is independent of KB grammar, theoretically any rule-based system can be analysed by VERITAS.

The Converter module allows the representation of external rules in an internal canonical form that is recognised by the other modules. Notice that this module works in two directions. It can also convert the canonical form into an external KB, generating new rules during knowledge updating, after anomaly detection, using an external grammar.

The Internal DB Administration module is responsible for the extraction and classification of all the information needed during the anomaly detection phase. In the first step all literals extracted from rules are classified according to the following schema:

- Fact – if it just appears in rule antecedents;
- Conclusion – if it just appears in rule consequents;
- Hypotheses – if it appears in both sides of the rules.

Notice that this classification is domain independent and just makes sense for verification procedures. This classification offers the advantages of a more compact knowledge representation and the reduction of the complexity of the rule expansion generating process. As it will be described later, this process corresponds to the analytical calculation of all possible inference chains.

In the second step, the Internal DB Administration module generates useful information about existing relations between literals (previously obtained). That information will be used not just to make the expansions generation process faster but also in the automatic detection of Single Value Constraints. VERITAS considers some type of constraints already described in literature [10]. Considered constraints can be classified in the following classes:

- Semantic Constraints – this type of impermissible set is formed by literals that cannot be present at the same time in the KB. Semantic constraints have to be introduced by the user.
- Logical Constraints – there are just two types of logical constraints: A and not(A) (where A stands for a literal); A and notPhysical(A); this designation is obtained by analogy with logical negation and allows to represent the constraint defined by a literal and by its retraction from the KB.
- Single Value Constraints – this type of impermissible set is formed by only one literal but considering different values of its parameters. Notice that those potential constraints are automatically detected. After this, the constraint can be either confirmed or changed by users.

The anomaly detection module (included in the Verification Tools) works in an autonomous way with no user interaction (i.e. it can run in batch mode). Presently this module can be used integrated with a developed tool (Knowledge Update Assistant) that, among other functions, allows rule edition. This functionality shows the existing relations between the rules that are to be modified and the remaining existing knowledge in the KB. This information is supplied in a graphical interface using a graph type representation. Moreover, it is possible to verify the rule in question immediately and to assure the KB consistency after the insertion of that rule.

When the verified Knowledge Base has large dimensions according to the number of rules and inference chains, the information generated during anomaly detection can be huge.

The detected anomalies have to be reported using a form suitable for easing its analysis. Special care has been put in this task, in order to reduce the time needed for the information analysis, so, it is possible to aggregate or select information by type of anomaly, number of rule and literal identification.

The anomaly detection relies on the rule expansions and constraint analysis. This method is also used by some well known V&V tools, as KB-REDUCER [1] and COVER [3]. As it has been described before, SPARSE has some specific features, due to these features the used technique is a variation of common ATMS (Assumption-based Truth Maintenance System) [2]. Namely, the knowledge represented in the meta-rules had to be considered in rule expansion generation.

VERITAS allows the rule expansion generation to be done in two different modes (see: Table 1): normal or exhaustive.

As an example, consider the following KB:

```
r1: t(X) and r(a) → s(a)
r2: f(a) → t(a)
r3: f(b) → t(b)
r4: h(a) → r(a)
r5: j(a) → r(a)
```

Table 1 – Rule Expansions Calculation

Normal Mode	Exhaustive Mode
t(X) and h(a)→ s(a)	f(a) and h(a)→ s(a)
t(X) and j(a)→ s(a)	f(a) and j(a)→ s(a)
	f(b) and h(a)→ s(a)
	f(b) and j(a)→ s(a)

It is possible to notice that “normal mode” generates fewer expansions but, on the other hand, the information obtained after anomaly detection is more useful. The “exhaustive mode” wastes a lot of time generating the rule expansions implying also more wasted time to analyse them, but, in principle, it will be possible to detect more potential errors.

The detected anomalies could be grouped in three major classes: redundancy, circularity and inconsistency (see: Fig. 3). There is another type of anomaly that is not, yet, detected by VERITAS, named deficiency. To detect this anomaly it is not enough to know the KB and its syntax, since deficiency detection requires that all inputs and outputs to/from the system are known. For the SPARSE system this work can be done using all types of SCADA messages.

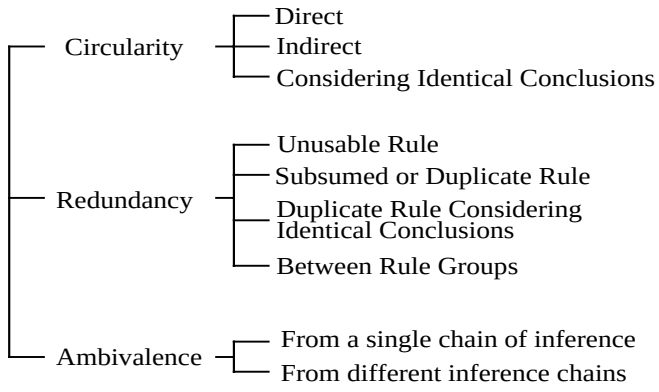

Fig. 3 - Anomaly Classification

This classification is based on Preece classification [4] with some modifications. First, the matching values are considered in rule analysis, meaning that a new set of anomalies will arise. Considering the following circular rules:

```
r1: t(a) and r(X) → s(a)
r2: s(a) → r(a)
```

For `X=a` some inference engines could start an infinite loop.

Another situation concerns to redundancy between groups of rules. In the following example:

```
r1: a and b and c → z
r2: not a and c → z
r3: not b and c → z
```

rules `r1`, `r2` and `r3` could be replaced by `rx` rule:

```
rx: a and b and c or not a and c or not b and c → z
```

Applying logical simplifications to rule `rx`, it is possible to obtain the following rule:

```
rx': c → z
```

Redundancy between groups of rules is a generalisation of the unused literal situation already studied by Alun de Preece [4]. Notice that this type of redundancy could be desirable. VERITAS can detect these situations using an improved Quine-McCluskey method for logical expression simplification.

Conclusions

This paper dealt with some important aspects for the practical use of KBS in Control Centres, namely knowledge maintenance and its relation to the Verification and Validation process.

The systematic use of Verification and Validation methods is very important for the acceptance of Knowledge-Based Systems by their end-users, especially when critical applications are considered. The use of Verification tools, based on formal methods, increases the confidence of the user and eases the process of changing KB, reducing the testing costs and the time needed to implement them.

This paper described SPARSE's V&V process, focusing on field-tests and techniques used during the validation phase. For the verification of SPARSE it was decided to implement a tool using a formal verification method.

VERITAS is a verification tool that performs the structural analysis in order to detect knowledge anomalies. We argue that the usefulness of VERITAS increases proportionally with KB size and the number of knowledge modifications, which must be undertaken.

Presently, VERITAS is being improved in order to allow the detection of anomalies related to temporal and nonmonotonic reasoning. We are also envisaging the use of VERITAS in verification of knowledge generated by Data Mining applications.

References

[1] Ginsberg, A. 1987. A new approach to checking knowledge bases for inconsistency and redundancy. In *Proceedings of the 3rd Annual Expert Systems in Government Conference.* 102-111. Washington, D.C., IEEE Computer Society.

[2] Kleer, J. 1986. An assumption-based TMS. *Artificial Intelligence* (Holland). 28(2):127-162

[3] Preece, A. 1990. Towards a methodology for evaluating expert systems. *Expert Systems* (UK). 7(4):215-223.

[4] Preece, A; and Shinghal, R.1994 Foundation and Application of Knowledge Base Verification. *Intelligence Systems*. 9:683-701.

[5] Rosado, C. 1993. Process TTLOGW. EDP Technical Report, RESP/SCDS 20/93, Electricidade de Portugal

[6] Santos, J. 1997. Verificação e Validação de Sistemas Baseados em Conhecimento – VERITAS, uma Ferramenta de Verificação. MSc Thesis diss., Dept. de Engenharia Electrotecnica e Computadores, Faculdade de Engenharia do Porto.

[7] Vale, Z. and Moura, A. 1993. An Expert System with Temporal Reasoning for Alarm Processing in Power System Control Centres. *IEEE Transactions on Power Systems* 8(3):1307-1314.

[8] Vale, Z.; Faria, L.; Ramos, C.; Fernandes, M.; and Marques, A. 1996. Towards More Intelligent and Adaptive User Interfaces for Control Centre Applications. In *Proceedings of the International Conference on Intelligent Systems Applications to Power Systems (ISAP'96).* 2-6. Orlando, Florida.

[9] Vale, Z.; Moura, A.; Fernandes, M.; and Marques, A. 1994. SPARSE - An Expert System for Alarm Processing and Operator Assistance in Substations Control Centres. *Applied Computing Review*. 2(2):18-26. ACM Press.

[10] Zlatareva, N.; and Preece, A. 1994. An Effective Logical Framework for Knowledge-Based Systems Verification. *International Journal of Expert Systems*. 7(3):239-260.