Verification & Validation of Power Systems Control Centres KBS

Jorge Santos; Carlos Ramos
Polytechnic Institute of Porto
Institute of Engineering
Department of Computer Engineering
Porto – Portugal
{jsantos|csr}@dei.isep.ipp.pt

Zita A. Vale
Polytechnic Institute of Porto
Institute of Engineering
Department of Electrical Engineering
Porto - Portugal
zav@dee.isep.ipp.pt

Albino Marques
REN – Portuguese Transmission Network (EDP Group)
Maia – Portugal
albmarq@ren.edp.pt

ABSTRACT

In last decades Knowledge Based System (KBS) became a common tool in a large number of Power Systems Control Centres (CC). In fact, the quantity, diversity and complexity of KBS increased significantly leading to important changes in KBS structure, especially in its nuclear components: knowledge representation and reasoning. The design of KBS applied in extremely dynamic environments demands the consideration of incomplete and/or incoherent knowledge as well as temporal aspects of knowledge.

The Verification and Validation (V&V) process allows stating whether the requirements have been correctly and completely fulfilled. In Control Centres domain the V&V process intends to assure the reliability of the installed applications, even under incident conditions.

SPARSE is a KBS used in the Portuguese Transmission Network (REN) for incident analysis and power restoration. VERITAS is a tool initially developed to verify SPARSE Knowledge Base (KB). This tool performs KB structural analysis allowing knowledge anomalies detection. Originally, VERITAS used a non-temporal KB verification approach. Although it proved to be very efficient in other KBS verification in SPARSE case some important limitations were detected.

This paper addresses the SPARSE V&V process, reporting VERITAS last improvements, namely the Temporal Reasoning analysis.

Keywords: KBS, Validation, Verification, Control Centres, Temporal Reasoning

1. INTRODUCTION

The correct and efficient performance of any peace of software must be guaranteed through Verification and Validation (V&V) process; it seems obvious that KBS should suffer the same evaluation process. In the other hand, it is known that knowledge maintenance is an essential issue for the KBS success but it must be guaranteed that the modified KB remains consistent and will not make the KBS incorrect or inefficient, leading again to the problems of V&V.

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, KBS to assist operators of Portuguese Transmission Control Centres in incident analysis and power restoration.

In the following section, some V&V field related definitions are presented. Section 3 describes SPARSE's characteristics, namely, architecture, reasoning model, rule selection mechanism and pointing its implications for Verification and Validation work. Section 4 describes the Validation stage of SPARSE development, especially the field tests and the need of applying formal methods during V&V. Section 5 presents VERITAS, a verification tool based on formal methods, detailing its architecture and main characteristics. Finally, section 6 presents some conclusions and future work.

2. BASIC DEFINITIONS

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 a KBS provides solutions that present a confidence level as high as the ones provided by experts. 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 a 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.
- Expansion – It is an inference chain that could be generated during system execution. For verification purposes, sometimes, an "expansion" could be explicitly computed and stored.
- Anomaly – an anomaly is considered as a symptom of one (or multiple) possible error(s) notice that an

anomaly does not necessarily denotes an error. Rule bases are drawn as a result of a knowledge analysis/elicitation process, including, for example, interviews with experts or the study of documents such as codes of practice and legal texts, or analysis of typical sample cases. The rule base should reflect the nature of this process, meaning that if documentary sources are used the rule base should reflect knowledge sources. Regarding that some anomalies are desirable an intentionally introduced in KB.

3. SPARSE

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.

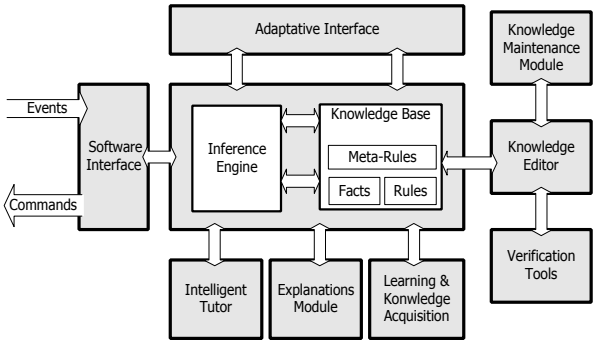

Figure 1 - SPARSE Architecture

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 KBS and are mainly intended to provide operators with assistance, especially in critical situations.

SPARSE [1] started to be an expert system (ES) developed for the Control Centres of the Portuguese Transmission Network, owned and operated by REN. The main goals of this ES were to assist Control Centre operators in incident analysis allowing a faster power restoration. Later the system evolved to a more complex architecture, which is normally qualified as a Knowledge Based System (see Figure 1).

Let us consider the following rule:

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

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 in RHS (Right Hand Side) (A1 to A2 in this example) may be of one of the following types:

- Assertion of facts (conclusions to be inserted in the knowledge base);
- 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(Fact,[(Rule1, Tb1, Te1),…,(Rulen, Tbn, Ten)])`

standing for:

- `Fact` – the arriving fact (external alarm or internal conclusion);
- `Rule1,…,n` – the rule that should be triggered in first place when fact arrives;
- `Tb1,…,n` – the delay time before rule triggering, used to wait for remaining facts needed to define an event;
- `Te1,…,n` – the maximum time for trying to trigger the each rule.

The inference process, roughly speaking, relies on the cycle:

1. SPARSE collects one “message” from SCADA (Supervisory Control And Data Acquisition);
2. The respective “trigger” is selected and some rules are schedule;
3. The “scheduler” selects the next rule to be tested;
4. The inference engines tries to prove its veracity;
5. Any reached conclusions that could be inferred will be asserted and later treated like SCADA messages.

4. VALIDATION

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.

REN's staff developed an application named TTLOGW [2] 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 validation.

In this way, files concerning to 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 when necessary.

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 on the way that 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:

- Simulation of incidents by operators located in chosen substations
- 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 protection equipment behaviour. 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.

In spite of all the difficulties and costs involved, this kind of test 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 a programmable impulse generator (PIG). The alarm messages generated by SCADA are simulated by impulses generated by the PIG.

This technique of test may be used in a chosen substation, being used the most convenient one, namely a substation close to the Control Centre. In this way, the operator who is manipulating the PIG is not far away from the team involved in the analysis of SPARSE results and may participate in co-ordination meetings between two consecutive tests.

This technique has allowed to simulate a wide set of incidents in order to validate SPARSE Knowledge Base as completely as possible.

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 were acceptable, the required time would force the KBS to be either out of service or to be in service without a validated Rule Base longer than it was desirable. It seems obvious that it is impossible to carry out these tests after each KB update, so verification tools offers an easy and inexpensive way to assure the knowledge quality maintenance.

5. VERIFICATION

A specific tool, named VERITAS [3] (see Figure 2) has been developed to be used in the SPARSE verification, performing structural analysis allowing to detect knowledge anomalies.

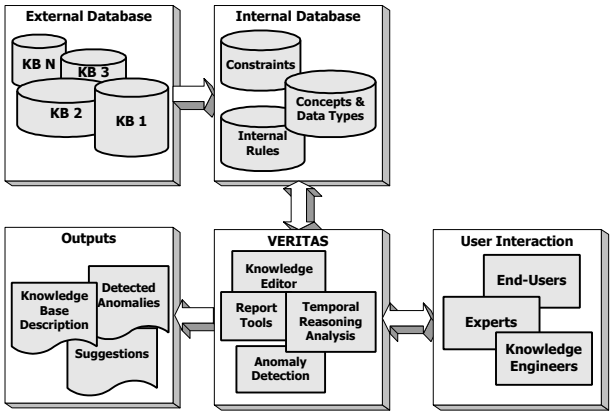

Figure 2 - VERITAS Architecture

VERITAS is knowledge-domain and rule-grammar independent, due to these properties, theoretically any rule-based system can be analysed by VERITAS. It has been developed with an open and modular architecture allowing user interaction throughout all the verification process.

5.1. Verification Constraints

SPARSE presents some features (see Figure 3) that make the verification work more difficult. These features force

the use of more complex techniques during anomaly detection and introduce significant changes in the number and type of anomalies to detect. The most important are:

- Variables evaluation – the variables evaluation is crucial (especially the temporal ones) in order to obtain comprehensive and correct results during verification process. Notice that during anomaly detection (this type of verification is also called static verification) it is not possible to know the values that a variable will have.
- Rule triggering selection mechanism – this mechanism was implemented using both meta-rules and the inference engine. When a "message" arrives, some rules are selected and queued to be later triggered and tested. This mechanism not only avoids some run-time errors (for instance circular chains) but also introduces another complexity axis to the verification, because the inference chains depend directly on the existence of this mechanism. For instance, during system execution, the inference engine should be able to assure that shortcuts (specialists rules) will be triggered before generic rules.

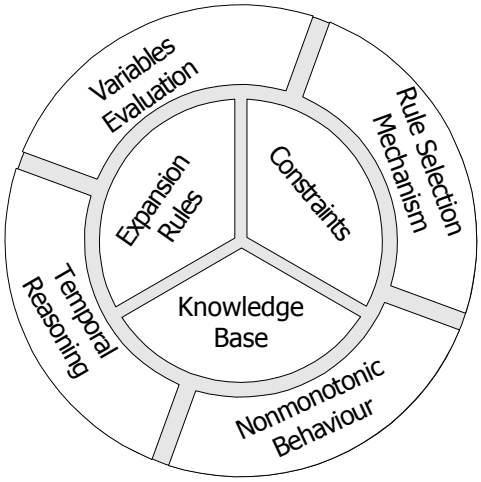

Figure 3 - Verification Constraints

- Temporal reasoning – the most difficult task for alarm processing applications is reasoning about events. Since it is necessary to "think" about time intervals and points, alarms order, duration and information presence or/and absence;
- Non-monotonic behavior – since the system models a dynamic environment the facts database should be non-monotonic by nature. Since the main application was developed using PROLOG, the non-monotonic behavior was design with the database asserting and retracting mechanism (very common in PROLOG).

5.2. Verification Database extraction

In the first step, VERITAS uses one specific filter for each KB to be verified, this filter includes the rules for the conversion between external and internal KB. During this step a database is constructed for verification purposes.

Considerer the following example were the rule `r1` is converted:

```
rule r1 : 'EXAMPLE' :
[
  [
  msg(A) at T1 and
  msg(B) at T2 and
  temporal_relation(T1,T2)
  ]
  or
  [
  msg(A) at T1 and
  msg(C) at T2 and
  ...
  ]
]
⟹
[
  retract(msg(A)),
  retract(msg(B)),
  assert(conclusion(X))
].
```

Some of the tasks executed during this step are:

- Generate a new rule for each conjunction on LHS;
- Replacing variables used in rules and triggers with VERITAS symbols;
- Representation of rules in a relational form in order to create indexes to speed up the expansion rules generation;

```
rule(OldRule,NewRule,Description)
rule(r1,'r1-l1','EXAMPLE').
rule(r1,'r1-l2','EXAMPLE').

literal(Label,Type,Functor/NArgs)
literal(e1,event,msg/1).
literal(t1,temporalOperator,temporal_relation/2).
...

lit_LHS(Rule,Label,TimeTag,LogValue,Args,Preds)
lit_LHS(r1-l1,e1,#T1,true,[#A],PREDS)
lit_LHS(r1-l1,e1,#T2,true,[#B],PREDS)
lit_LHS(r1-l1,t1,none,true,[#T1,#T1],PREDS)
lit_LHS(r1-l2,e1,#T1,true,[#A],PREDS)
...
```

Note: `Preds` is the list of predecessors rules that allows to conclude the respective literal (represented by Label)

Similar information will store for RHS in a structure called `lit_RHS`.

In the second step, some useful information is generated about existing relations between literals (previously obtained). VERITAS considers some type of constraints already described in literature [4]. The considered constraints can be classified according to 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. The following example concerning substations operation mode, where `T,I` means Time and Installation:

  ```
  {remote(T,I), local(T,I)}
  ```

- Single Value Constraints (SVC) – 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 following example shows an automatically detected SVC, where `T,I,P` means Time, Installation and Panel:

```
{
  disconnector(T, I, P, open),
  disconnector(T, I, P, closed),
  disconnector(T, I, P, changing)
}
```

5.3. Anomaly Detection

The anomaly detection module (Verification Tools) works autonomously with no user interaction and runs in batch mode. Presently, this module can interact with Knowledge Editing Tool (see Figure 1) that, among other functions, allows rule edition. This tool 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 directed hypergraph type representation (see Figure 4). This technique allows rule representation in a manner that clearly identifies complex dependencies across compound clauses in the Rule Base and there is a unique directed hypergraph representation for each set of rules [5].

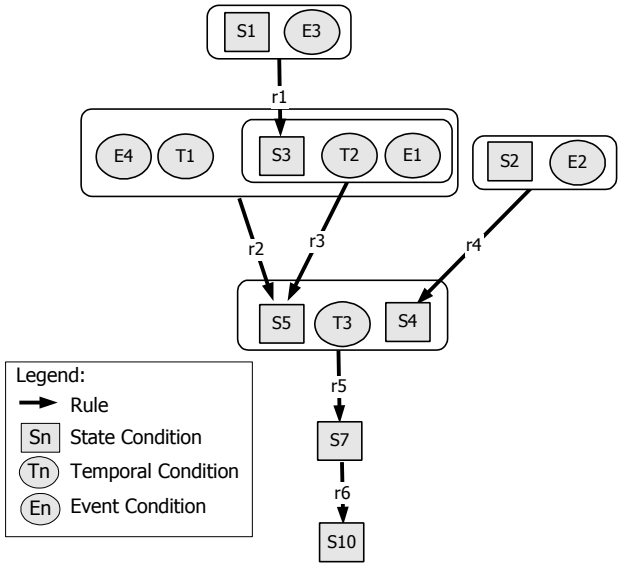

Figure 4 - Hypergraph representation

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 [6] and COVER [7]. As it has been described in section 3, SPARSE has some specific features. According to these features the used technique is a variation of common Assumption-based Truth Maintenance System (ATMS) [8]. Namely, the knowledge represented in the meta-rules had to be considered in rule expansions generation.

Notice that in earlier authors work [3], during rule expansions calculation, one data structure was created for each possible inference chain. In the most recent VERITAS version, each "literal" (see section 5.2) keeps information about the list of it predecessors; it means that it is possible with a graph-visiting algorithm to compute expansions. This approach showed to be more time-efficient (during anomaly detection, since some tests do not need to know all rule expansions) and data flexible (e.g. circular chains became possible to exist in the verification database).

VERITAS verification tool is able to detect KB existing anomalies in a first step using logic tests. Later, the verification system performs temporal reasoning analysis allowing to filter previously detected anomalies, providing really accurate information, namely by preventing false anomaly reporting.

5.4. Variables Evaluation

As it was said before (see section 5.1), the variables evaluation is crucial to obtain acceptable results during anomaly detection.

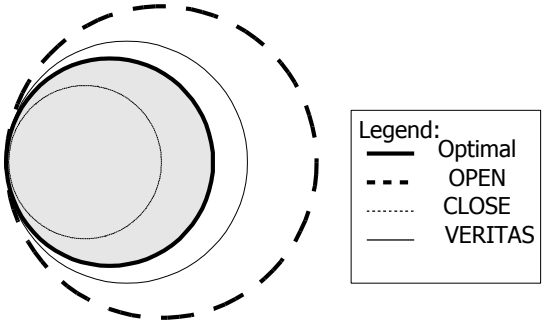

Figure 5 - Relation between match type and expansions number

The used technique relies on variable name replacing in the original rule by a string that uniquely represents the argument (variable), later, during rule expansion generation this information will be used to check possible matches.

Different match types could be used during evaluation (see Figure 5):

- OPEN – in this type of match free variables could instantiate with terminals, consider the following example: `disconnector(X)`, `disconnector(open)`, where `X` could take "`open`" value. In this type of match a large number of expansions have to be considered and consequently a lot of "false" anomalies could be detected;
- CLOSE – if it is not allowed that free variables match with terminals (in opposition to the previous example), the number of rule expansions generated will make the detection process simpler but the

tradeoff is that some anomalies could remain undetected;

- VERITAS – the used approach is an "OPEN" one in order to allow detection of all possible anomalies but at same time generate the minimum number of expansions. How is that achieved? A free variable could be replaced just by valid values in knowledge domain (previously defined in database). The expansion rules are computed as late as possible and kept in a graph structure, during expansion rules computation a similar process to PROLOG instantiation mechanism is used, so, each variable instantiation is reflected in following transitions.

The detected anomalies can be grouped in four major classes: redundancy, circularity, ambivalence and deficiency. Regarding that anomalies definition depends on KBS characteristics, let us consider some definitions:

- Circularity - a knowledge base contains circularity if and only if it contains a set of rules, which allows an infinite loop during rule triggering.
- Ambivalence - a knowledge base is ambivalent if and only if, for a permissible set of conditions, it is possible to infer an impermissible set of hypotheses.
- Redundancy - a knowledge base is redundant if and only if the set of final hypotheses is the same in the rule/literal presence or absence.
- Deficiency - a knowledge base is deficient if and only if there is not a rule that uses a set of conditions representing a possible domain knowledge situation.

This classification is based on Preece classification [9] with some modifications.

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 suitable form for easing its analysis. Special care was putted in this task, in order to reduce the efforts needed in information analysis, so, it is possible to aggregate or/and select information by: type of anomaly, number of rule and literal identification.

6. 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. The use of Verification tools, based on formal methods, increases the confidence of the user and eases the process of changing KB. It also reduces the testing costs and the time needed to implement those tests.

This paper described SPARSE's V&V process, focussing 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.

7. ACKNOWLEDGEMENTS

The authors of this paper would like to acknowledge PRAXIS XXI Research Program, FEDER, the Portuguese Science and Technology Foundation and Innovation Agency for their support of SATOREN Project and Project PRAXIS/3/3.1/CEG/2586/95.

REFERENCES

[1] Z.Vale, A.Moura, M.Fernandes and A.Marques, "SPARSE - An Expert System for Alarm Processing and Operator Assistance in Substations Control Centers", ACM (Association for Computing Machinery) Press, Applied Computing Review, 2(2), December 1994, 18-26

[2] C.Rosado - "Process TTLOGW", EDP Report RESP/SCDS 20/93, July 1993

[3] J.Santos, L.Faria, C.Ramos, Z.Vale, A.Marques - "VERITAS – A Verification Tool for Real-time Applications in Power System Control Centers". *In Proceedings of 12th International Florida AI Research Society (FLAIRS'99).* Eds. AAAI Press, 511-515. Orlando, Florida, 1999

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

[5] M.Ramaswary and S.Sarkar – "Global Verification of Knowledge Based Systems via Local Verification of Partitions", *In Proceeding of the 4th European Symposium on the Validation and Verification of Knowledge Based Systems (Eurovav'97),* 145-154. Leuven, Belgium, 1997

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

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

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

[9] A.Preece and R.Shinghal – "Foundation and Application of Knowledge Base Verification", Intelligence Systems, 1994, 9, 683-701