VALENS: A Knowledge Based Tool to Validate and Verify an Aion Knowledge Base

Silvie Spreeuwenberg, Rik Gerrits, Margherita Boekenoogen*

Abstract . We present VALENS (VALid Engineering Support) for the validation and verification (V&V) of a knowledge base (KB). Validation techniques become more and more important when knowledge based systems (KBS) are widely used to automate business critical processes. The tool we present may be used during and after the development of a KBS. It focuses on a logical verification of the KBS. The techniques used to verify a KB are: meta-rules, an inference engine to verify hypotheses posed by meta-rules (proof-by-processing) and meta information (provided by the user). VALENS is written in the same language as the KBS it verifies: Aion.

1. PROBLEM DESCRIPTION

In our everyday practice we encounter the following situation: we have a KBS to determine the type of a concept. The KBS was tested and correct until a new rule is added to determine a new type. The results are completely different than expected and it took some time to find out that a form of circularity in the rules caused the problem. This is what could be called the butterfly theory in a KBS. Solutions to avoid this problem are V&V techniques. The ideas are incorporated into VALENS, a tool that applies validation and verification techniques to Aion KBS.

1.1 What's the need?

An iterative development cycle is preferred for developing KBS [1]. Decreasing the number of cycles between analysis, implementation and test decreases development time. Early detection of faults using V&V by detection of inconsistencies and incompleteness can contribute to this reduction. Furthermore the issue of assuring the quality of a KBS is becoming an increasingly important challenge as KB components are more and more often embedded within safety critical or business critical applications [2].

During maintenance a change in just one knowledge rule may introduce contradictions, redundancy or incompleteness in a complex chain of knowledge rules. Especially when people without a background in system programming or system analysis define and maintain the knowledge in a KBS, the support of a V&V tool helps them to cope with the complexity.

In all the main phases of the knowledge engineering life cycle, V&V is an important aspect when it comes to delivering a high quality KBS. In this article we will focus on the automatic support of V&V in the implementation phase of the knowledge engineering process.

1.2 Definitions of checks

When verifying the logical correctness of a KB, it is checked whether the rules in a KB are logical consistent, non-circular, complete, not redundant and not obsolete[1]. These checks may cover all rules in the KB or a subset of the rules in a KB. The control structure present in most knowledge-based systems facilitates the dynamical posting of rules or rule sets. If this is the case, these subsets have to be verified in isolation.

2. APPLICATION DESCRIPTION

VALENS can be used by a developer after or during construction of a KB or can be integrated in a tool that allows users to write their own business rules. The output of the tool is a document in which all invalid rules (combinations) detected are reported. Each fault is classified and explained.

In the next paragraphs, the main algorithm and benefits in our approach to V&V is briefly discussed. Next, the development environment and the tool itself are discussed.

2.1 Verification algorithm

The verification algorithm that VALENS uses performs three main steps:

a) Construction of meta model

In this step all rule constructs, necessary to reason about the rules in the KB are instantiated. This step is performed on a "when needed" basis to reduce performance overhead.

b) Select potential anomalies

Potential anomalies are selected with the use of heuristics. These heuristics where designed as meta rules but are implemented as procedures due to performance considerations.

c) Proof anomalies

The theses (potentially invalid rules) are proved by running the rules to be tested in a forward chaining mode, while providing

* The authors are employed by LibRT B.V., PO Box 90359, 1006 BJ Amsterdam, Netherlands, Email: info@LibRT.com

[1] The taxonomy of anomalies from A. Preece [3].is followed except that the term contradiction is used in stead of ambivalance.

them with the right truth-values (input). We call this process proof-by-processing.

This process resembles the "saturation" process used by Nouira and Fouet [4]. Nouira and Fouet generate the largest possible number of properties for an object that would certainly appear during a real execution and then start forward chaining during which it tries to fire constraints. In VALENS the heuristics defined in the meta rules search for the smallest number of properties for an object wherefore the potential anomaly can be proved. VALENS does not use explicit constraints. The knowledge contained in the constraints of Nouira and Fouet are integrated in the meta rules of VALENS. In contrast to the meta rules of VALENS, the constraints of Nouira and Fouet contain semantic knowledge. Semantic meta rules are foreseen but not yet implemented in VALENS (see 3.3 Future Enhancements)

For a more detailed description of the proof-by-processing algorithm used in VALENS to detect anomalies the user is referred to Gerrits and Spreeuwenberg [8].

2.1.1 Benefits of Proof-by-processing

The proof-by-processing algorithm has some benefits in comparison to the algorithms based on formal logic. The most important benefit is that proof-by-processing allows us to deal with predicate logic (i.e. functions are allowed to be used in rules). The functions that are used in the rules may contain procedural algorithms. An example of the use of functions in rules is given in the result window in figure 3. The rule "clients must be older than 15" uses the function or method (as it is called in an OO environment) "Age". This method returns the number of years between the current date and the attribute "Birthdate" of the class "Relation". This is not a predefined method, the user may design any method and use it in rules. VALENS will find which attributes are used in the method (by recursively assessing the method and the methods that are being called by the method) and execute the method during the proof-by-processing algorithm. The consequence of executing the method on the execution of the rules will therefore be taken into account.

A second important benefit of the proof-by-processing algorithm is that we can work in an object-oriented (OO) environment. As seen in figure 3 the rules work on attributes of the current instance. These attributes can be inherited.

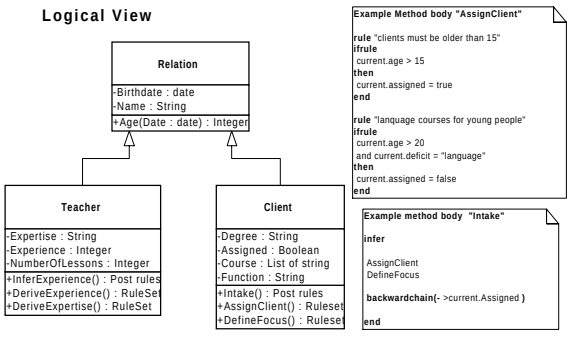

Figure 1. UML Class diagram

In figure 1 the UML class diagram of the knowledge and domain concepts seen in figure 3 is shown. The rules are modelled in methods and can contain a very rich OO language shown in the notation elements in the right hand side of figure 1.

A third and very important benefit of proof-by-processing is that there can be no discrepancy between the run time logic and the logic used in the validation process because the inference engine used in the verification process is the same as the inference engine used to evaluate and fire the knowledge rules in the application.

2.2 V&V in AION

The V&V tool VALENS is built in, and made for Aion8[2] (short: Aion) applications. Aion is a widely used commercial development environment for KBS and intelligent components. Some characteristics are:

- The inference engine supports rule and decision table processing in a backward, forward chaining or recursive forward chaining mode.
- The programming language is object-oriented.
- Meta-programming features enable a programmer to obtain information about the state of the inference engine.
- The Callable Object Building System (COBS) feature allows one to automate all the functions a developer can use in Aion.

At the moment Aion does not include any rule V&V strategies but customers who maintain Aion rule bases with more than 100 rules have asked for these facilities. Integration in the Aion development environment is one of the future directions of VALENS.

2.3 The tool VALENS

The V&V application consists of three components: a user interface, the verification engine and a reporting component. The relationship between these components is given in the context diagram of figure 2.

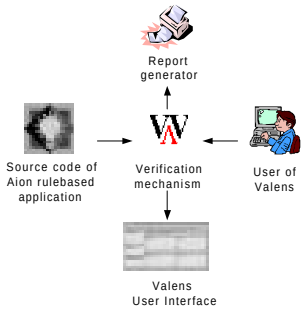

Figure 2. Context diagram of V&V tool

The user selects the KB and rule sets within that KB that need to be verified. When there are potential 'invalid rules' detected during the verification process, the KB is started in a forward chaining mode to test the thesis. We than capture the results of the inference engine for analysing whether a thesis is satisfied, and to catch the chain of logic that has caused a thesis to be satisfied.

[2] Aion is a product and trademark from Computer Associates.

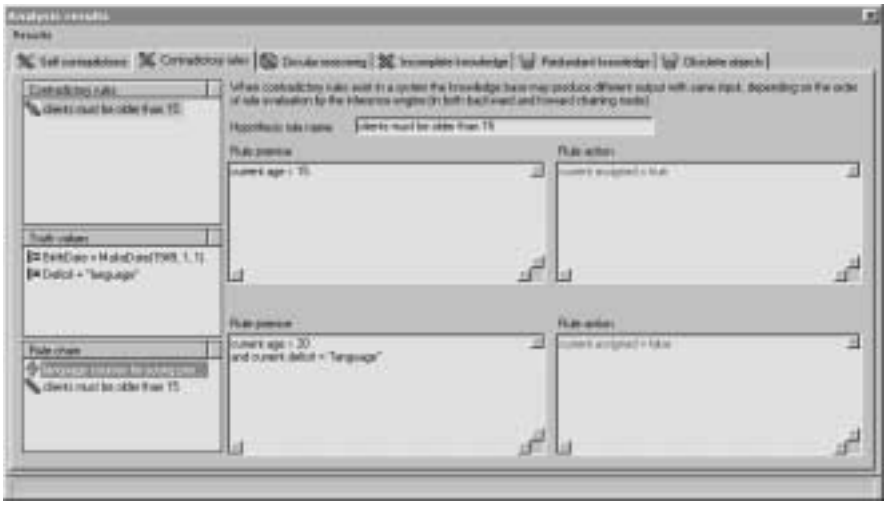

Figure 3. Result window of VALENS

Invalid rules are reported in a HTML document and on a result window (containing the same information). Each fault is classified and explained as shown in figure 3, which shows the result-tab for contradictory rules.

The results window shows a general explanation of the anomaly and the conflict that is detected. A conflict is defined in [9] as a minimal set of rules, eventually associated to an input fact set, that is a sufficient condition to prove an anomaly. The rules, which have been detected as "contradictory", are shown in the list on the upper left-hand side of the window. When a rule is selected, the rule chain (the set of rules that caused the rule, which is contradictory to the selected rule, to fire) is shown in the list at the lower left-hand side of the window. The rule that is contradictory to the selected rule has the fault-icon (thunderbolt-sign) in this list. When a rule in the rule chain is selected, the rule text, rule premise and rule action are shown in the text boxes. The list in the middle shows the truth-values (input fact set) under which the particular contradiction is proved.

Groups of knowledge rules and the status of faults found during V&V can be stored in a database to allow regression testing.

3. APPLICATION BUILDING

VALENS is developed by LibRT BV. LibRT is a young, Dutch company which aims at helping customers integrate knowledge-based solutions into their own information-based solutions. We strongly feel that the art of knowledge engineering should be added to the default skills of any business analyst, application consultant or software engineer. To achieve this goal, LibRT develops products containing the knowledge of knowledge engineers, of which VALENS is an example.

3.1 Development and project team

It required about one man-year to develop VALENS. The project team consisted of the three authors of this article. They complement each other's skills and had their own focus in the project.

All three project members have a background in KBd system development and their respective focus in the project was:

- Functional requirements and the main concepts involved in our approach to V&V.
- Development and coding in Aion
- Testing and exploitation of VALENS

3.2 Exploitation phase

The exploitation phase of VALENS is divided in three stages. In the first stage we have searched for collaboration with a commercial party that had already developed knowledge-based systems in Aion. We investigated the quality of their applications by using VALENS and delivered an evaluation report. This stage gave us the opportunity to create references, test VALENS on real world applications and investigate the market for V&V tools

In the second stage we will collaborate with a commercial party where we can integrate VALENS in the knowledge engineering life cycle. We expect that in this phase, the costumer will have specific wishes regarding the user interface and validation capabilities of VALENS. This second stage gives us the opportunity to communicate with the users of VALENS about the requirements for a user interface, system integration and domain specific validation.

In the third stage, we will have developed a generic user interface for VALENS and VALENS will be sold under a license agreement[3]. At this moment, we are in the second stage of the exploitation phase of VALENS.

3.3 Future enhancements

At present, LibRT is working at a further refinement of the user interface component and the reporting component. Further development of VALENS will involve the active support of users in solving faults detected and in suggesting solutions to the problem at hand. Automatic repair of faults will be supported but only by the explicit agreement of the user. We emphasize that the end-user has to be in charge in this problem solving process.

Also, we would like to extend VALENS with a form of semantic validation by using domain specific meta-rules. These domain specific meta-rules are constraints or business rules. These business rules will be specified and defined by a business analyst or domain expert. The defined business rules can be stored and edited from within the V&V tool. These business rules will be handled in the same way as the meta rules described in section 2.

4. APPLICATION BENEFITS

The quality and maintainability of software is becoming a more and more important issue for managers of large scale IT projects. When rule-based techniques are used it is possible to detect anomalies automatically during the development process which can reduce development and test time and, even more important,

[3] Presumably as an add-on in the Aion development environment.

increase the quality of the rule base. The risks of maintenance are reduced when the application is of high quality. Furthermore, the use of VALENS will speed up the learning curve of novice programmers with respect to the basics of good rule-base programming.

4.1 Experience with real life applications[4]

VALENS is tested using a KB that contains a large veriety of anomalies based on the sample rule bases of A. Preece and extended with specific tests for the rich Aion rule language constructs and object orientation concepts.

Postbank Nederland BV became interested in the promise of a V&V tool for their AionDS[5] assessment KB. In a two months pilot project VALENS was evaluated in a real business situation. The target KB had been written in AionDS 7 and contained approximately 250 rules. Because the current version of VALENS is developed for Aion8 KBS, the target KB had to be converted to an Aion8 KB. The conversion process was accomplished in one day. Important in this respect is that VALENS only needs a 'valid rule base' to verify the rules: no GUI or other interfaces need to be converted.

LibRT got the first version of the customer's KB to verify when the developing team of the Postbank had finished the rule base and the testing phase was at hand. Though VALENS can be applied earlier in the application development lifecycle, it was perfect timing: there would be a parallel V&V and testing phase so the results of both processes could be compared.

VALENS did not detect any real errors in the KB. Though this might look disappointing, the testing phase neither did reveal any error that could have been detected by verification. VALENS did find many redundant and obsolete constructs in the KB. Some of these constructs were intentional, others were not, but everyone was impressed with the fact that VALENS was able to highlight these 'points of interest'.

VALENS proved to be of good use in maintaining the integrity of the functional specifications of the KB and the realized (and revised!) KB. Of course, VALENS detected problems that were not problems at all but thanks to this pilot project, Valens has become more mature and robust. After two months, the pilot project was not continued because it became clear that the target KB will probably never be converted to an Aion8 KB.

At present, we have started a collaboration with a party that uses VALENS for the V&V of legal-models. VALENS is used, in the first place, by the analysis and development team (approximately 5 people). The results of VALENS are discussed by domain experts, which have a background in legislation. In the first results VALENS detected incomplete knowledge. The expert knowledge that was added to resolve the incompleteness resulted in the detection of a circular reasoning by VALENS, which was than again corrected by the domain experts.

4.2 Usability

Performance issues are important during the development of VALENS. During the use of VALENS performance is less important because users need not be present while the system runs. We advise the user to start VALENS after work or during lunch. Our test application contains 30 rules and does the search for redundancy, the most time consuming check, in less than two minutes on a Pentium II. However, this time indication doesn't say anything about another rule base containing 30 rules. It is not the number of rules that determine verification time but the structure of the knowledge that is contained in the rule base. We are not aware of a measure indicating complexity of a rule base. Such a measure is needed when a prediction of the validation time for a given rule base has to be made.

The result window shown in figure 3 is a sufficient explanation to be used by a knowledge engineer who has an understanding about KBS and an inference engine. However, the result window is not sufficient for explanation of the results to experts or inexperienced programmers. Therefore LibRT would like to improve the presentation of results with graphics, for example a visual dependency graph of the logical chain, and more explanatory text.

At this moment VALENS is a stand alone application installed on a local PC which has a valid license of Aion. The user selects the Aion source code that he wants to verify. VALENS will show the user a view on the selected rule base in which the user can select rules to be verified and the different analyses to be performed on these rules. LibRT is intending to integrate VALENS in the Aion development environment for use during the development and maintenance of an application. This will relieve the programmer from opening Aion source code in the VALENS application. VALENS can also be integrated in a domain specific knowledge editor tool and used early in the analysis stage. One can also think of integrating VALENS in a test-tool, however, we recommend to start with V&V techniques early in the development life cycle.

4.3 Comparison to similar systems

In the beginning of the '90s, the universities devoted much attention to V&V of KBS. There were some tools developed to verify rule bases of which Preece [10] has given an overview and comparison. An even more extensive overview comes from Plant [11] who lists 35 V&V tools build in the period 1985–1995. Most of the systems where developed at a university and it is hard to find out what the current status of those systems is. What happened with those systems? Some of them will still have a research status and are used to explore new research domains. For example, the COVER tool of Preece is evolved in the COVERAGE tool for verifying rule bases in a multi agent architecture. [12]. And the PROLOGA tool [13] is extended with intertabular verification [14]. But perhaps the 'boost' for V&V tools failed to occur because the promise of KBS failed in commercial environments. Another factor might be that not only business environments but also university research is driven by 'hypes' like 'knowledge mining', 'knowledge management' and 'intelligent agents' which follow each other in such tempo that there is no time to pick the fruit of planted trees. However, the current need for more quality, less unpredictable software development projects and better maintainable systems might change the prospects of V&V tools.

The verification tools can be compared on a number of criteria. One criterion is formed by the anomalies that are detected by the tool. Some tools do not detect anomalies in a chain of logic, for example the Rule Checker Program (RCP) [15] and CHECK [16].

[4] Due to confidentiality contracts with our customers we are not able to show the real results obtained on the real knowledge bases

[5] Aion is a new version of AionDS.

Others like RCP, CHECK and EVA [17] do not detect missing rules and unused literals. VALENS is complete with respect to the anomalies defined by Preece [10].

Another criterion is the language that is supported by the tool. Most V&V systems, which verify a KB, cope with a restricted language, for example first order predicate logic [6][7] or formal specification language [5] as opposed to the rich language of a programming environment like Aion. There are also tools which have their own internal language defined and which, manually or automatically, translate diverse languages to the internal language. EVA is an example of a system with its own internal language and provides a set of translation programs that translate the rule languages of some expert system tools (for example, ART, OPS5 and LES) to an internal canonical form, based on predicate calculus. PROLOGA works the other way around, it allows a user to create and verify decision trees and then generate code in diverse programming languages (for example, Aion, Delphi and C++). COVER and VALENS work in the programming language they where developed with, which is respectively Prolog and Aion. It will be straightforward to automatically convert KBS written using other rule-based expert system shells into the rule language used by the V&V tool. However, if one wants to propose revisions of the rule base based on the verification results, the translation process will pursue extra complexity to this process.

If tools are bounded to a (programming) language, they are likely to be integrated in an expert system shell or, as we call it today, development environment. VALENS will be integrated in Aion in the future. This makes the tool less generic; however, the core algorithms of VALENS are independent of the programming language[6]. The algorithms do pose assumptions on the 'trace-ability' of the inference engine. For the algorithms of VALENS to work correctly the inference engine needs to give information about the rule-firing network after a chaining process is finished. LibRT has plans to verify Java code, in combination with a Java rule engine, in the future.

The last criterion for comparison of V&V tools is their respective behavior in the analysis and development phase of a system. The work of Nouira and Fouet [7] concentrates on the analysis phase of a system but results in a valid and executable KB. The work of van Harmelen [5] also concentrates on the analysis phase and verifies formal specification language. The idea is that the formal specification has to be translated to a programming language to get an executable program. VALENS is a V&V component. With the integration of the tool in Aion, we focus on the development and maintenance stage of an application but its use is not restricted to this stage because the tool is developed as a component and can be integrated in an analysis support environment.

5. DISCUSSION AND CONCLUSION

We have discussed a tool developed in and for Aion applications that supports the verification and validation of rules. The tool can be used during development and maintenance of the KBS. Another application is the integration of our technique with a domain specific editor to support the definition of knowledge rules by a domain expert or business analyst.

[6] A Component Based Development (CBD) approach has been used during the project.

The tool uses meta-rules, meta information and the inference engine of Aion to accomplish this task. By using the same inference engine in the verification process as in the execution process of the rules, there can be no discrepancy between the run time logic and the logic used in the validation process.

We think this tool will reduce the 'time to market' for knowledge-based systems, improve their quality and has added value during maintenance of applications by developers, especially for those who are not familiar with the functionality of the application. A domain expert can not be expected to have an extensive background in logic, as a developer should have. Therefore, the tool is a "must have" when domain experts define their own knowledge rules.

REFERENCES

[1] Boehm, B.W., 1986, A Spiral Model of Software Development and Enhancement. Computer, 21, 61-72

[2] Ed P. Andert Jr., 1992, Automated Knowledge Base Validation, AAAI Workshop on Verification and Validation of Expert Systems (July 1992)

[3] Preece, Shingal, 1994, Foundation and Application of Knowledge Base Verification, International Journal of Intelligent Systems, 9, 683 – 701

[4] Rym Nouira, Jean-Marc Fouet, 1996, A Knowledge Based Tool for the Incremental Construction, Validation and Refinement of Large Knowledge Bases, Workshop Proceedings ECAI96

[5] F.V.Harmelen, 1995, Structure Preserving Specification Languages for Knowledge Based Systems, International Journal of Human Computer Studies, Vol. 44, 187-212

[6] Alon Y. Levy And Marie-Christine Rousset, 1996, Verification of Knowledge Bases on Containment Checking, Workshop Proceedings ECAI96

[7] Z. Bendou And M. Ayel, 1996, Validation of Rule Bases Containing Constraints, Workshop Proceedings ECAI96

[8] R. Gerrits, S. Spreeuwenberg 1999, A Knowledge Based Tool to Validate and Verify an Aion Knowledge Base, Validation and Verification of Knowledge Based Systems, Theory, Tools and Practice, 67 – 78

[9] F. Bouali, S. Loiseau, M.C. Rousset, 1997, Revision of Rule Bases, Proceedings Eurovav 97, 193 – 203

[10] A. Preece, 1991, Methods for Verifying Expert System Knowledge Bases.

[11] Robert T. Plant, 1995, Tools for Validation & Verification of Knowledge-Based Systems 1985 – 1995 References, *Internet Source*

[12] N. Lamb And A. Preece, Downloaded: 01-05-2000, Verification of Multi-Agent Knowledge-Based Systems, *Internet Source*

[13] J. Vanthienen, 1991, Knowledge Acquisition and Validation Using a Decision Table Engineering Workbench", World Congress of Expert Systems, 1861 – 1868

[14] J. Vanthienen, C. Mues, G. Wets, 1997, Inter-Tabular Verification in an Interactive Environment, Proceedings Eurovav 97, 155 – 165

[15] M. Suwa, A.C. Scott, E.H. Shortliffe, 1982, An Approach to Verifying Completeness and Consistency in a Rule-Based Expert System, AI Magazine, Vol. 3, Nr. 4

[16] W.A. Perkins, T.J. Laffey, D. Pecora, T.Nguyen, 1989, Knowledge Base Verification, Topics in Expert System Design, 353 – 376

[17] C.L. Chang, J.B. Combs, R.A. Stacowits, 1990, A Report on the Expert Systems Validation Associate (EVA), Expert Systems with Applications, Vol. 1, Nr. 3, 217 – 230