Verification and Validation of Knowledge Bases Using Test Cases Generated by Restriction Rules

André de Andrade Bindilatti andre_a_a@yahoo.com.*br*
Federal University of São Carlos
Computing Department
São Carlos, Brazil

Ana Estela Antunes da Silva *aeasilva@ft.unicamp.br*
State University of Campinas
Faculty of Technology
Limeira, Brazil

Abstract

Knowledge based systems have been developed to solve many problems. Their main characteristic consists on the use of a knowledge representation of a specific domain to solve problems in such a way that it emulates the reasoning of a human specialist. As conventional systems, knowledge based systems are not free of failures. This justifies the need for validation and verification for this class of systems. Due to the lack of techniques which can guarantee their quality and reliability, this paper proposes a process to support validation of specific knowledge bases. In order to validate the knowledge base, restriction rules are used. These rules are elicit and represented as *If Then Not* rules and executed using a backward chaining reasoning process. As the result of this process test cases are created and submitted to the knowledge base in order to prove whether there are inconsistencies in the domain representation. Two main advantages can be highlighted here: the use of restriction rules which are considered as meta-knowledge (these rules improve the knowledge representation power of the system) and a process that can generate useful test cases (test cases are usually difficult and expensive to be created).

Key-words: Knowledge Based Systems, knowledge Base Inference, Restriction Rules, Validation, Verification.

1. INTRODUCTION

Initial research efforts in Artificial Intelligence (AI) were done towards general models for problem solving. One example of this was the General Problem Solver (GPS) [10].The search for such general models was a great challenge. In the evolution line, a new approach was a kind of system that could deal with specialist knowledge. Those systems are knowledge based systems (KBS).Knowledge presented in a KBS is restricted to a unique domain or competence. The architecture of a KBS is composed by three fundamental components: (1) A user interface that allows the interaction of the user to the system; (2) an inference engine which performs the search in the knowledge base to solve the problem; (3) a knowledge base in which knowledge is stored in some kind of formal representation, usually production rules.

The process of collecting, organizing and representing knowledge is called knowledge acquisition and it is one of the most difficult phases in the construction of a KBS[1]. The professional responsible for this phase is the knowledge engineer [5]. The knowledge acquired is stored in the knowledge base. It is possible to observe the presence of several factors that can compromise the consistency of a knowledge base such as: knowledge inconsistency resulting from specialists interviewed or sources as books; misinterpretation of the domain by the knowledge engineer; errors in knowledge representation; insufficient knowledge to solve problems among others.

Considering those factors the processes of Validation and Verification (V&V) become a fundamental phase for the development of KBS. The processes of V&V for the interface and the inference engine can be done by traditional techniques of software engineering, however V&V for the knowledge base need specific techniques. This is due to the declarative nature of the knowledge base representation, instead of procedural algorithms as in traditional systems.

This paper proposes a process for V&V and tests of knowledge bases in KBS based on restriction rules and generation of test cases. Section 2 describes concepts of V&V; section 3 mentions related works; in section 4 a description of restriction rules is done; section 5mentions works of restriction rules for V&V process of KBS; section 6 presents the process proposed in this article; section 7shows the trace of the process using a practical example; section 8 presents a comparison of our work to other approaches and section 9 concludes the paper.

2. VALIDATION AND VERIFICATION

Validation and verification are part of a series of techniques and methodologies applied to the evaluation of a system with the purpose of defining whether its construction is correct (verification); and for defining whether the system corresponds to its initial specifications (validation) [3].

In [8], validation process is defined as the process of assuring that a system satisfies user requirements; and verification as an activity in the validation process which aims to verify whether the system attends formal requirements. In other terms, validation is the process that evaluates the system capacities to solve problems. On the other hand, verification process evaluates whether the system is appropriately developed.

Verification is widely associated to the question: "are we constructing the system correctly?". The V&V process focus on assuring that a system can present characteristics such as quality and reliability in an acceptable level in order to be considered as a real system in a situation in which it is common the existence of risks and losses in consequence of system failures.V&V techniques applied to software development can be classified, in general, among the following types [3]:

- **Informal**: are based in the human interpretation and experience, and take as principles good sense practices, but, sometimes, become inefficient;
- **Static**: are based on techniques applied to the representation model verifying its data flows, control flows and syntax;
- **Dynamic**: are based on techniques applied to the functions or modules of the software. They are usually applied to complement the use of static techniques;
- **Formal**: consist in the mathematical proof of the software.

Most V&V techniques in KBS are dedicated to evaluate inference rules in knowledge bases representation [16]. Some of the problems which can be presented in the knowledge base are [17]:

- Subsumption Rules: occur when the antecedent part of a rule is a subset of the antecedent of another rule and both rules present the same consequent. The rule with fewer predicates in the antecedent (more general) is considered and the others (more specific) left aside. In the example:R_1: $p \wedge k \rightarrow q$ and R_2: $p \rightarrow q$, R2 would be maintained in the knowledge base.
- Duplicated rules: occur when two or more rules in the knowledge base are identical.
- Inconsistency: occur when two rules have the same antecedents, but different consequents. In the following example: R_1: $p \rightarrow q$ and R_2: $p \rightarrow \neg q$ contains a paradox.
- Rules in Circles: occur when the execution of a chain of rules leads to the first hypothesis creating some kind of looping. This can be noticed in the example: R_1:$p \rightarrow q$; R_2: $q \rightarrow k$ and R_3: $k \rightarrow p$.
- Unreferenced inputs: occur when there are not values for all predicates present in the rule base.
- Impossible rules: when the antecedent of a rule cannot be satisfied. Example: R_1: $p \wedge \neg p \rightarrow q$.

According to [17], such anomalies are source of problems and let the knowledge base susceptible to possible inefficiencies. However, by treating the problems mentioned there is no guarantee of the knowledge validation. When those anomalies are corrected is possible to say that we are treating those anomalies in terms of verification (are we constructing the knowledge base correctly?) but we are not answering the question: are we constructing the right system?, i. .e, we are not validating knowledge in the knowledge base. This work addresses the validation problem by considering domain restriction rules as a way to validate a specific knowledge base.

3. VERIFICATION AND VALIDATION IN KBS

Because of this lack of techniques for V&V for KBS several works have been developed in this area. Some of them are briefly mentioned as follows. In [16], it is possible to find points related to KBS development. They point some common problems related to KBS development such as: (1) absence of requirements documents; (2) difficulty to generate test cases to validate the system; (3) use of prototypes as a way to elicit requirements; (4) absence of a life cycle; (5) difficulty to maintain the system as new rules are included in the knowledge base.

A study about an overlapping area between KBS and Databases technologies is presented in [2]. This work presents the possibility of using V&V techniques of KBS in Databases. On the other hand, [4] presents a study of the use of V&V and integrity techniques of databases to KBS.

In [14], authors propose the use of cover analysis for V&V of KBS. Cover analysis is one technique applied to V&V in software engineering. It can be classified as a dynamic approach once it implies the execution of the software product. Using cover analysis is possible to know the extension of the software that was tested. Besides, it can demonstrate how efficient a set of test cases is when applied to the software being tested. An ideal set of test cases is the one that, with a minimal amount of test cases, finds the biggest amount of defects. In the approach used by [14], the concept of cover analysis is adapted to KBS using the concept of cover as being the number of rules executed by each test case and for the total number of rules executed during the test activity.

In [11], test cases are created and selected as a way to refine knowledge. If a test case is used and it is not able to detect a violation then the method proposes that a new test case can be generated. They affirm that since test cases are difficult or hard to obtain it would be necessary a strategy to choose them in order to accept or refuse a system refinement

The work in [7] proposes the use of data flow as a graphical representation of a rule base. This graphical form, called a logical path graph, captures logical paths through a rule base. These logical paths create the abstractions needed in the testing process.

4. RESTRICTION RULES FOR V&V

Several works have proposed the use of restriction rules to validate and to maintain the integrity of data. In [16], authors use restriction rules to validate the content of data in XML messages in applications based on the service oriented architecture (SOA). In SOA, where traffic of complex data is common, the consistency of private data is done in the application level. Here, data and content consistencies, which are susceptible to privacy, are performed in the application level. In this approach, restriction rules are defined in a specific XML file and define restriction conditions and actions to be performed when a restriction is detected.

In system development, inconsistencies between the final system and its initial specification of architecture can be a problem. The authors in [9] propose a language to the specification of restrictions rules for system architecture. The rules can be used to automatically check the system development and its initial architecture specification.

Another application of restriction rules involves scenarios containing Web services. A Web service must perform transactions from different organizations. Besides, a business transaction can involve operations using several sources. In the case of failures or exceptions, more dynamic mechanisms of recovery must be applied once several sources of data are involved in the transaction. In [6], authors propose the use of restriction rules in business transactions during failure occurrences. In this case, restriction rules represent business logic more appropriately by improving recovering and consistency maintenance of business transactions.

5. RESTRICTION RULES FOR V&V of KBS

The present work proposes the application of restriction rules as a way to generate test cases to V&V of KBS. Restriction would be constructed based on valid inputs and valid outputs for the rules in the knowledge base. These restrictions would be dependable on the problem domain and on the inference process expected from the KBS.A specific domain could be, for instance, a KBS to help users to choose an appropriate configuration of hardware for their needs. Possible restriction rules could be: 1)if one of the inputs is "user works with 3D modelling" then one of the outputs must be "include a 3Dgraphic card"; 2) if "basic knowledge user" is one of the outputs then "developer user " must not be another input.

Observe that the first example illustrates a restriction rule which express a restriction about an incoherent conclusion according to a given input. In the second example, a restriction rule is used to point out a conflicting input. Considering these two types of restriction, two types of restriction rules are proposed here: restriction rules for input values (*IRR)* and restriction rules for invalid inferences (*InfRR*).

Restriction rules are represented in the same way as *IF-Then* rules of the knowledge base. However, because the consequent part of the rule represents a domain restriction which must be imposed when the antecedent is satisfied, a restriction rule is represented as an*IF - Then Not* rule. The consequent of the rule indicates a knowledge restriction.

For *IRR*, both antecedent and consequent parts refer to input variables; for *InfRR* the antecedent part refers to input values and the consequent part refers to output values. *IRR* and *InfRR* can be respectively represented by: **IF**<inputvariable1>**THEN NOT**<inputvalue2>; **IF** <inputvalue>**THEN NOT**<outputvalue>. Next section presents how the proposed approach can use restriction rules to generate test cases for V&V of *IF-Then* rules in KBS.

6. GENERATION OF TEST CASES USING IF THEN NOT RESTRICTION RULES

In order to perform V&V processes of KBS, this work proposes an approach to generate test cases of*If-Then* rules from a set of *If- Then Not* restriction rules. Restriction rules specify the expected behaviour of the KBS. In KBS, restriction rules can be seen as part of the knowledge representation that is expected from the system. They should be created according to the domain problem restrictions, therefore should be acquired during the knowledge acquisition process. As they express knowledge about the knowledge base, it is possible to classify then as a meta-knowledge base [1].

The approach proposed here allows to generate test cases to KBS which uses forward chaining reasoning. The forward chaining inference method considers an initial set of facts and, by the execution of *IF Then* rules of the knowledge base, fires the rules to which the antecedent is satisfied. New facts are stored in the working memory and the inference motor seeks for new rules to be satisfied according to the new state of the working memory. This process continues until there is no more facts to be inferred or a stop condition is reached.

The algorithm used to generate test cases is based on the inverse of the forward chaining reasoning method. Considering an invalid conclusion, *i. e*., a conclusion that violates an inference restriction rule, the algorithm tries to prove that this conclusion can be obtained from valid input values. Figure 1 presents the main steps of the algorithm.

```
For each rule$R_i$ in {InfRR} /*Inference Restriction Rules Set*/
    Set proof goal $\delta \leftarrow rhs(R_i)$; (right hand side (rhs) )
    $\Upsilon \leftarrow backwardChaining(\delta)$;     /* Υ : set of proofs from backward chaining reasoning executed in
                                                        /* the  knowledge base
    For each $v$ in $\Upsilon$
        $Input \leftarrow lhs(R_i) \cup v$ ; /* left hand side (lhs) of the $R_i$ */
        If$Input$ satisfies ( {IRR} and { $\forall input, input \in Input \rightarrow \neg input \notin Input$} )
    /* input must be a valid input according to the Input Restriction Rules set*/
        /* Input must not contain contradictory inputs  */
            $Output \leftarrow \delta$        /*  proof goal  */
            generate test case $\tau \leftarrow (Input, Output)$;
        End
    End
End
```

FIGURE 1. Algorithm main steps

As the first step, the first *InfRR* is executed. The negation of its consequent is considered as a proof goal and applied to the knowledge base using a backward chaining inference process. The antecedent of this *InfRR* is then included in the Input set to be part of the input values of the test case. This process tries to prove that a violation of an *InfRR* can be supported by input variables in the knowledge base.

All antecedents of successful rules in the knowledge base, which are valid inputs according to the *IRR*, are included in the Input set which will be used to create the first part of the test case. The output of the test case is the proof goal. The result is a test case which contains valid inputs and outputs which can demonstrate a violation or a non-violation of the knowledge base as they were generate considering a violation of an *InfRR*. A test case is a pair of sets which contain valid inputs and valid outputs: $\tau \leftarrow (Input, Output)$.

The idea here is to generate useful test cases by selecting some of the values from the input value set of a KBS. If all possibilities of input values were considered to generate test cases, the test process could have a very high cost because of the combinatory explosion of values [11].

Once test cases were generated, it is important to verify the extension of the knowledge base that was validated. In order to do that a cover metric can be used[14]. A simple way of measuring the cover of a test case can be the ratio between the number of rules tested and the total number of rules in the knowledge base. A low ratio means that the test case set was not able to test most of the knowledge base rules. This indicates that more inference restriction rules should be created.

The following steps summarize the proposed approach of test case generation:

(1) Restriction rule specification: should be acquired during the knowledge acquisition process;
(2) Test case generation: test cases are generated from the execution of the backward reasoning algorithm whose input data are restriction rules;
(3) Test case execution: generated test cases are used as inputs to the knowledge base execution;
(4) Cover measure calculation of the test case set: for the set of executed test cases a measure of how much of the knowledge base was tested must be applied;
(5) Results presentation: a report containing violated restriction rules, cover measure and test cases should be presented.

7. A SAMPLE TRACE OF THE ALGORITHM

In order to illustrate the backward chaining algorithm execution a practical example is presented which uses a fictitious knowledge base. Such knowledge base contains expertise about the decision problem of whether or not to grant credit for a given credit applicant. Figure 2 contains the rules that compose our sample knowledge base.

```
1.  if age < 18 then grant_credit = no : stop
2.  if credit_exp = good and employed = yes then score >= 10
3.  if credit_exp = bad then score <= 10
4.  if age > 18 and age < 21 then score <= 5
5.  if rent = yes and dependants = yes and age > 18 and age < 21 then score <= 30
6.  if rent = yes and dependants = no and age > 18 and age < 21 then score <= 20
7.  if rent = yes and dependants = no and age > 21 then score <= 10
8.  if rent = yes and dependants = yes and age > 21 then score <= 15
9.  if employed = yes and employed_time> 12 then score >= 10
10. if credit_exp = bad and score < 20 then grant_credit = no : stop
11. if score >= 20 grant_credit = yes : stop
```

Figure 2. Sample knowledge base

Our approach requires the knowledge engineer to specify a collection of constraint rules. These rules serve as inputs for the validation algorithm, which is responsible for generating test cases which will be use in the V&V process of the sample knowledge base. Figure 3 presents the set of constraint rules related to the sample knowledge base.

An iteration of the algorithm will be outlined next, in order to illustrate our approach. Since the algorithm iterates through the constraint rules, we begin by selecting *InfRR* number one. This rule specifies that there is a violation of the restriction rule when debtor=yes and grant_credit =yes. The only rule in the knowledge base that supports this conclusion is the rule number eleven (*if score >= 20 grant = yes : stop*).

```
1.  IRR := if credit_exp = good then not age < 18
2.  IRR := if employed_time<= 0 then not employed = yes
3.  IRR := if employed = no then not employed_time> 0

1.  InfRR := if debtor = yes then not grant_credit = yes
2.  InfRR := if employed = no then not grant_credit = yes
3.  InfRR := if age < 18 then notgrant_credit = yes
4.  InfRR := if score < 10 then notgrant_credit = yes
```

FIGURE 3. Restriction rules for the sample knowledge base of Figure 2.

Following the backward chaining strategy, our new goal is to prove the premises in the antecedent portion of rule eleven. In order to do that, rules two and nine need to be fired (R2*if credit_exp = good and employed = yes then score += 10; R9 if employed = yes and employed_time> 12 then score += 10).*

It turns that the conditions in the antecedent of rules two and nine are formed only by input variables (*credit_exp = good; employed = yes and employed_time> 12).*We assign the input set accordingto the antecedent of rules two and nine, respecting the backward chaining procedure. The *InfRR* constraints express that a given output should not occur when a certain input is used. So, we also assign the value *debtor = true* specified in the InfRR number one as part of the input.

At this point we have the following set of values: Input= {credit_exp = good; employed = yes; employed_time> 12; debtor = yes}. For each value in Input it is necessary to verify whether the input value does not violate the *IRR* set and simultaneously does not violate the antecedent of the current *InfRR*. As values satisfies both conditions then the test case would be:

testcase*=<(credit_exp = good; employed = yes; employed_time> 12; debtor=yes); (grant-credit=yes)>*(I).

After this step, the algorithm would continue iterating through the remaining *InfRR* rules. At the end of this process all test cases would be generated. After that they would be submitted to the knowledge base. The forward chaining execution of test case (I) in the knowledge base was successful what indicates a violation. The test case produced a set of valid inputs according to the *RRI* set and produced an invalid output according to the*InfRR*. The test case (I) produced grant_credit = yes when one of the inputs was debtor=yes. This violates the first *InfRR*.To solve this, a new rule such as: " IF *debtor = yes THEN grant_credit = no : stop";* would be necessary to grant the knowledge base consistency.

8. COMPARISON WITH OTHER APPROACHES

In [18], a WEB-based expert system to evaluate suppliers is presented. In this work, knowledge acquisition is guided using interviews with experts and knowledge is represented by if-then production rules. In order to validate the system, a comparison between the solution obtained by the system and the solution proposed by the expert is done. As mentioned by the author, the two solutions were the same.

Although this validation can be seen as satisfactory in some situations, it can be insufficient in others. Time spent by experts can be a considerable cost to the system. Besides, there are no evidences that indicate how to define a set of real cases that can result in a total cover of possible test cases that can guarantee system validation. The author mentions as future works, the need to use other approaches of test to validate the system.

In [12], a tool to V&V process called Veritas is presented. Veritas has a domain-independent modular architecture which is based on production rules to represent knowledge. It is used to detect structural anomalies in knowledge bases as the ones mentioned in section 2.

A direct comparison between Veritas and the proposed approach cannot be done as the methods are conceptually different. In the proposed approach, knowledge behavior and knowledge restrictions are applied to the knowledge base. This allows a V&V process which takes into account the semantic of the domain problem. Although structural anomalies can compromise the performance of an expert system, their detection does not validate the knowledge representation of domain problem.

The Veritas tool also allows the creation of a model with all possible combinations of tracks that can be taken during the inference process. Here, in the proposed approach, the space of combinations is pruned using restriction rules.

The work in [14] presents a tool for V&V process called Valens. The tool is based in meta-rules and in an inference method of hypothesis verification. The verification process is made in steps: 1)a meta-model is constructed according to the knowledge base rules to offer a more convenient structure of inference; 2) possible anomalies are elicited using the meta-rules; 3) using the inference method, candidate anomalies are verified in order to detect and prove the presence of problems.

The proposed approach shares two characteristics with the method applied by the Valens tool: 1) the use of a set of rules to support the V&V process; and 2) the use of an inference mechanism to prove the presence of problems. However, the approach differs from the approach of Valens in the kind of problem to be detected. As previously mentioned, knowledge domain is tested according to requirements of the system behavior. The method proposed by [14] aims to verify consistency, circularity and redundancy.

The approach proposed here is a declarative V&V process to be applied to knowledge bases in expert systems. This declarative characteristic is fundamental for two reasons: 1)knowledge engineer is not necessarily a knowledge engineer, i. e., he/she does cannot detent knowledge about V&V techniques used in tests of conventional software processes; and 2)It is more difficult to apply conventional techniques of V&V to expert systems given their knowledge declarative nature.

9. CONCLUSION

KBS are one of the most known techniques in Artificial Intelligence. However, to use them to solve real practical problems maintaining a trustable knowledge representation, it is necessary to have an efficient V&V process. Because of the declarative form of the knowledge base, which contrasts with the procedural form of classical algorithms and databases, these systems demand more specialized techniques to perform their V&V process.

In this work an approach to support V&V process of KBS was proposed. This approach consists on the generation of test cases from restriction rules. Restriction rules can be seen as meta-knowledge of a specific domain problem.

Problems such as the ones mentioned in section II can be detected through the inspection of the expert systems rules. However, given the fact that the knowledge based systems approach requires a model of expertise, the direct inspection of the rules is not enough to detect problems in the knowledge engineering process. The constraint rules are proposed in order to add knowledge to the validation process, attempting to enable the automation of the V&V process of knowledge based system.

In the same way that V&V techniques are used to validate and verify traditional systems, the use of restriction rules to generate test cases to validate KBS is not a technique able to prove that a system is free of violations, but it is a technique used to detect inconsistencies about the expected system behaviour. In our case, to detect inconsistencies about the domain knowledge.

A practical example showed that using restriction inference rules it is possible to detect inconsistencies in the knowledge base. As inconsistency is detect it is possible to the knowledge engineer proposes the inclusion of new rules in the knowledge base in order to solve the problem.

As future works authors indent to verify whether knowledge engineers can incorporate the acquisition of restriction rules as part of the knowledge modelling process.

REFERENCES

[1] AKERKAR, R.; SAJA, P. Knowledge Based Systems. Jones and Bartlet Publishers. 2010.

[2] COENEN, F. Verification and Validation Issues in Expert and Database Systems: The Expert Systems Perspective. Proceedings. Ninth International Workshop on Database and Expert Systems Applications, 1998.

[3] DEBBABI, M.; HASSAÏNE, V. F.; JARRAYA, Y.; SOEANU, A.; ALAWNEH, L..Verification and validation in Systems Engineering: Assessing UML/SYSML Design Models. Springer, 2010.

[4] EAGLESTONE, B. ; RIDLEY, M.Verification, validation and integrity issues in expert and database systems: the database perspective. Proceedings of the Ninth International Workshop on Database and Expert Systems Applications, 1998.

[5] GIARRATANO, J.; RILEY, G..Expert Systems Principles and Programming. Third ed. PSW Pub., 2005.

[6] JIUXIN C.; BIAO, Z.; BO, M.; BO, L.. Constraint Rules-based Recovery for Business Transaction. In: Grid and Cooperative Computing (GCC), 2010 9th International Conference, Pg. 282 – 289, 2010.

[7] KIPER, J. D..Structural Testing of Rule-Based Expert Systems. In: ACM Transactions on Software Engineering and Methodology, Vol. 1, No. 2, Pg. 168-187, April 1992.

[8] LIU, G.; LIU, Q.; XIE, P.Model-based testing and validation on knowledge-based systems. In: Grey Systems and Intelligent Services, 2007. GSIS 2007. IEEE International, 2007. pp. 1242-1245.

[9] MARINESCU, R.; GANEAIN, G..Code.Rules: An Agile Approach for Defining and Checking Architectural Constraints. In: ICCP '10 Proceedings of the Proceedings of the 2010 IEEE 6th International Conference on Intelligent Computer Communication and Processing,2010.

[10] NEWELL, A.; SHAW, J.C.; SIMON, H. A..Report on a general problem-solving program. In Proceedings of the International Conference on Information Processing. Paris: UNESCO House, Pg. 256-264, 1959.

[11] PALMER, G., & CRAW, S. The Role of Test Cases in Automated Knowledge Refinement. 16th BCS SGES Expert Systems Conference, Cambridge, UK. 1996.

[12] SANTOS, J.; RAMOS, C.; VALE, Z. A.; MARQUES, A.. VERITAS-an application for knowledge verification. In: Tools with Artificial Intelligence, 1999. Proceedings. 11th IEEE International, 1999, pp. 441-444.

[13] SMITH, S.; KANDEL, A..Verification and validation in fuzzy expert systems. In: Systems, Man and Cybernetic. Intelligent Systems for the 21st Century. Vol. 4: Vancouver, BC, 1995.

[14] SPREEUWENBERG, S.;GERRITS, R.;BOEKENOOGEN, M.: VALENS: A Knowledge Based Tool to Validate and Verify an Aion Knowledge Base. In W.Horn (ed.): ECAI2000, Proceedings of the 14th European Conference on Artificial Intelligence, IOS Press, Amsterdam, 2000, pp.731-735.

[15] TERAGUCHI, M.; YOSHIDA, I.; URAMOTO, N.. Rule-based XML Mediation for Data Validation and Privacy Anonymization. In: IEEE International Conference on Services Computing, 2008.

[16] TSAI, W.; VISHNUVAJJALA, R.; ZHANG, D..Verification and Validation of Knowledge-Based Systems. In: IEEE Transactions on Knowledge and Data Engineering, vol. 11, no. 1, January/February 1999.

[17] VERMESAN, A. I..Software Certification for Industry: Verification and Validation Issues in Expert Systems. In: Proceedings of the 9th International Workshop on Database and Expert Systems Applications: Washington, DC, 1998.

[18] WANGPHANICH, P..A Simple Web-based Expert System for a Supplier Assessment: A Case of a JIT production environments. In: Proceedings of 2011 International Conference on System Science and Engineering, Macau, China, 2011.