A Graph-based Approach for Rule Base Maintenance

Kunihiro Higa¹ and Ho Geun Lee²
Department of Information and Systems Management
School of Business and Management
Hong Kong University of Science and Technology
Clear Water Bay, Kowloon, Hong Kong
fax: 852-2358-2421

¹ tel: 852-2358-7639
email: k-higa@usthk.ust.hk

² tel: 852-2358-7649
email: hlee@uxmail.ust.hk

Abstract

As conventional software systems have maintenance costs far exceeding development costs, so do rule-based expert systems. Any rule base for decision support evolves over time. Most research works in expert systems have focused on building and validating rule bases, leaving the maintenance issue unexplored. We propose a graph-based approach, called Object Classification Model (OCM), as a methodology for the rule base (RB) maintenance. An experiment is conducted to compare the OCM with traditional rule base maintenance methods. The results verify that the OCM helps knowledge engineers to keep the RB integrity and to increase the RB maintainability.

1. Introduction

Software maintenance has become an important task as maintenance accounts for a considerable amount of the time and cost of the life-cycle of a software system. For traditional software systems, maintenance costs exceed the development costs by a factor of between two and four [1, 7]. There is no reason to assume that this should be any different for rule-based expert systems, provided they have a reasonable operational life. Any knowledge base for decision support is doomed to evolve. For instance, the knowledge base of the DEC's XCON had grown from 700 rules to over 6200 rules since its operation, with 50% of its rules changing every year [10]. As a rule base changes over time, it can lose integrity in the same fashion as conventional programs. This is magnified by the fact that most systems might be maintained by people who did not develop the initial system.

Despite its importance, the maintenance of rule-based systems has not been well explored: most research works in rule based systems so far have focused on the construction of the rule base, emphasizing knowledge acquisition, representation and validation. There is no established methodology for the maintenance of rule bases (RB), thus making knowledge updates challenging tasks. We propose a graph-based approach called Object Classification Model (OCM) as a methodology for maintaining rule bases. This paper reports on an experiment which compares the rule base maintenance by a traditional method with that by the OCM. Our hypothesis is that the OCM facilitates updates of knowledge by increasing consistency and maintainability of rule bases.

The paper is organized as follows. First, related works are discussed to differentiate our approach from existing methods. Then two key requirements for the rule base maintenance, integrity and maintainability, are defined and the OCM is proposed as a methodology for the RB maintenance. This is followed by the discussion of the experiment. Our concluding remark is contained in the last section.
2. Related Work

The rule-based system development process typically includes knowledge acquisition, representation, implementation, validation and maintenance. Knowledge maintenance is not independent from other phases of the development. It is highly dependent on the representation: good representations make rule bases much easier to understand and hence maintain. Knowledge maintenance has also obvious parallels with the knowledge validation since the defects that may result from maintenance can equally result from the initial construction of the rule base. The difference is that validation addresses the entire rule base, while the error checking associated with the maintenance activities need only impinge on the section of the rule base.

Several methods have been proposed to verify the knowledge. These approaches are designed to detect inconsistency, missing rules and conflicts in rule bases. Examples include MELODIA [2], Expert System Checker [4], Rule Checking Program [11, 12] and CHECK [8, 9]. However, our approach to the rule base maintenance is different from these knowledge validation techniques. First, the OCM is designed for a preventive maintenance rather than a curative maintenance. Knowledge validation techniques take already established knowledge and detect errors and anomalies resident in the rules. We view the preventive approach as important as the curative one, since the former reduces many human errors in advance as rules are inserted, deleted and modified. Second, the maintenance is concerned with maintainability as well as integrity of rule bases. The advantages of writing readable and understandable code are well appreciated for traditional software maintenance. The graph-based approach helps knowledge engineers eliminate a priori redundancy and subsumption, thus keeping rule bases structured for easy changes.

3. Rule Base Integrity and Maintainability

Expert systems (ES) evolve over time. The maintenance of rule bases may result in inconsistency as well as unstructuredness of rules. In this section, various illegal rule structures that make an RB unreliable, and complex rule structures that make RB maintenance difficult, are discussed.

3.1 Integrity

Logical inconsistency, dead-end-IF, and incompleteness are addressed in this paper as sources of the integrity problem. Inconsistency results from conflicting rules, while incompleteness is caused by missing rules.

Conflicting Rules:
When two (or more) rules have logically equivalent antecedents (or one subsumes the other) but conclude differently, they are conflicting rules.

Dead-end-IF:
A rule (or a set of rules) that has no connection to a goal is called a "dead-end-IF". A rule R_i is dead-end-IF when:
G_i, a conclusion of a rule R_i, is not a final goal and
G_i is not used as a condition by any other rules.
Consider, for example, an RB with two rules, R_1: IF A=true AND B=true THEN G=g_1 and R_2: IF A=false THEN B=false where G is the final goal. R_2's consequence (B=false) is not the final goal value, and it is not connected to R_1; therefore, R_2 is a dead-end-IF.

Missing Rules:
A gap exists in the RB when a valid set of antecedent values is not represented in the RB. For example, suppose that an RB contains only two rules, R_1 and R_2, such that the antecedent part of two rules are identical except for values of one common attribute (a_{11} for R_1 and a_{21} for R_2). Then a gap exists between these two rules if the union of these values (a_{11} \cup a_{21}) does not cover its attribute value domain (thus there are missing rules).
3.2 Maintainability

The maintainability of RB is concerned with redundancy and unstability of rules. In contrast to the integrity, these rules are logically sound but increase the potential of inconsistency in the future. Thus, the RB maintainability problem can be tested by checking the existence of either redundant rules or unstable rules. Redundancy in an RB is typically caused by the presence of subsumption or duplication of rules. An RB is considered to be unstable when a simple change in one rule easily creates inconsistency in the RB.

Duplication:
R_i is a duplicate of R_j if R_i has a logically equivalent antecedent part with R_j and shares a common consequent value with R_j. Duplication is a special case of subsumption which is explained below.

Subsumption:
R_j subsumes R_i if the antecedent of R_i logically dominates the antecedent of R_j, and both rules share a common consequent value.

Adjacency:
R_i is adjacent to R_j if the union of both antecedent parts contains no gap, and both rules share a common consequent value (Note: The "unnecessary IF condition" discussed in [8] is one type of adjacency).

4. The Object Classification Model

The Object Classification Model (OCM) is a tree-shaped graphical model for knowledge representation. It is created based on the Structured Object Model (SOM) that was originally developed as an analysis, design, and navigation tool for database applications [6]. The SOM itself is a derivative of the System Entity Structure (SES) that was initially developed as a model representation scheme for simulation modeling [13].

4.1 Components and Constructs of OCM

The OCM represents data semantics using objects, attributes, and two types of relationships: aspects and specialization. An object is an event or an activity about which users wish to collect and store information. Attributes are used to describe objects by providing them with descriptive properties. Relationships represent associations among objects. Aspects describe owner-member relationships, while the specialization expresses the classification/categorization relations of objects. The use of the OCM for building rules is well documented in [5].

Translating a set of rules into the OCM diagram is a quite simple mechanical process. Each rule becomes a path of the tree. Each attribute in the antecedent part of the rule becomes a classification of a path. Naturally the consequence, or goal value, of a rule becomes the leaf. This process is completely automatible since it requires no human interaction. Figure 1 shows the OCM diagram translated from the rule base used for the experiment in this paper. The production rules, listed in Appendix, are designed to evaluate applicants to a research institute and to determine their appropriate rank (G, P1 to P4) based on applicant age, education, past experience and foreign language fluency. In the OCM diagram, multiple classification out of an object typically indicates a reasoning chain. The diagram in Figure 1 has three classifications out of the root object, QC (rank qualification class), I_Q (initial qualification) and Age; therefore the three subtrees form a reasoning chain.

Although there are some systems [3, 8] that automatically check for logical completeness and consistency of a rule set, it is a good practice to check for these problems manually before using such systems. It is generally known that the use of manual tracing (including structured walk through) reduces the cost of programming. Similarly the proposed manual checking of the knowledge structure will reduce the cost of knowledge base development.
Figure 1. The Experiment Rule Set OCM.
The proposed OCM is similar to a decision tree. However, a decision tree in its current form cannot be used for designing and maintaining rule bases in the same manner as the OCM does because there is no distinction between aspects and specialization of objects in a decision tree. Therefore, a decision tree is inappropriate for the design and analysis of rule bases that contain multiple related objects with multiple specializations.

5. Experiment

5.1 Hypotheses

In order to determine the effectiveness of the OCM for RB maintenance, an experiment is conducted. The following three hypotheses are tested in a controlled lab experiment with MBA students as subjects.

Between the traditional approach and the graph-based approach in modifying the rule set:

H1) The subjects perform significantly better using the graph-based approach.
H2) The subjects make significantly less ill-structured rules using the graph-based approach.
H3) The subjects make significantly less un-reliable rules using the graph-based approach.

The three hypotheses (H1 to H3) are formed to test the positive effect of using the OCM diagram on the maintenance of the RB. For H2, rules with the aforementioned structures that cause the maintainability problem are considered as ill-structured rules. Similarly for H3, rules with the aforementioned structures that cause the integrity problem are considered as un-reliable rules.

For the experiment, three insertions of rules were designed so that the insertion of one rule affects the existing complex rule structure: that is, each insertion required modification of more than one existing rule in the set. The rule set contained 33 rules with complex structures. Therefore, the maintenance task was a non-trivial one.

5.2 Experiment Design

Independent Variables
Two approaches for the RB maintenance, traditional and graph-based, are used for the study. There is no established method available for the RB maintenance. Therefore, the traditional approach relies solely on the skill of a knowledge engineer. On the other hand, the graph-based approach uses the OCM to help the knowledge engineer maintain the RB.

Dependent Variables
Integrity and maintainability of rules after maintenance are employed as dependent variables. Since logical correctness is the most important for the maintenance, higher priority is given to the integrity than to the maintainability. No restriction is given on the number of correct solutions as long as they are semantically equivalent and cause no logical problem.

Subjects
Forty-eight MBA students enrolled in the introduction Expert Systems class were recruited and trained to perform the RB modification tasks. Subjects had received limited credit for the course based on their performance in the modification tasks. The average age of the subjects was 28. The subjects had average work experience (full time) of 4.8 years, and none had previous experience in maintenance of RBs.

Procedures
The subjects were divided into two groups, 24 subjects for each group, based on their age and work experience. As pre-test training, one lecture (90 minutes each), one homework, and one in-class exercise (both were similar to the task in the experiment) were given to the subjects one week before the experiment. For one group, the OCM diagram was taught and used throughout the pre-test training. Then the subjects were asked to perform the RB modification task in the experiment.

Prior to the experimental task, subjects were told the following two points:
(1) There are three insertions of rules for the maintenance task, and there is no time limit for the task.

(2) Although 30 minutes are suggested for the task, subjects' performance (for their course credit) is strictly based on the correctness of the modification, and the time taken is irrelevant to the evaluation of their performance. The 30-minute suggestion was based on the result from a pilot test.

The experiment was conducted as follows:

(1) First, the two-page instruction was distributed to subjects. The instruction included requirements for modification, abbreviations used in the tasks, and domain definitions for the rule sets (see Appendix). Similar instruction had been used in pre-test training; therefore, subjects were already familiar with the instruction.

(2) The rule set and the task sheet were distributed to the subjects. They were not allowed to start until they were told so. Then the subjects were asked to perform the maintenance job for the rule set. The job included insertions of three new rules into the current rule set.

(3) As soon as a subject completed the task, the subject brought the rule set and the task sheet to the nearest proctor (there were four proctors in the experiment room). The proctor then time-stamped the subject's task sheet. The subject was asked to leave the room upon the completion of the task.

Grading
A score of 1 was given to each insertion if the rule set was logically sound and contained no ill-structured rules after the insertion. If the rule set became unreliable after the insertion, that is, it contained dead-end-IF rules, conflicting rules, or missing rules, a score of 0 was given. Ill-structured rules in a rule set do not immediately make the set unreliable; however, they make the set inefficient and expensive to maintain. Therefore, 0.5 point was taken off for each ill-structured rule created by the insertion.

The evaluation criteria are described in the following:
(1) Completeness of the insertion - A score of 0 was given if the insertion process was not completed.

(2) Logical consistency - A score of 0 was given if the insertion caused any logical inconsistency between rules. Number of conflicts (inconsistency between rules) and dead-end-IF rules were also recorded for each subject.

(3) Completeness of the domain coverage - A score of 0 was given if the insertion resulted in any incomplete domain coverage. Since each original rule set has complete domain coverage (all attribute and goal values are covered by the rule set), each incomplete coverage resulting from the insertion can be considered as a missing rule. Thus, the number of missing rules was also recorded for each subject.

(4) Ill-structured rule - 0.5 point was taken off from the score if the insertion created any adjacency, duplication and subsumption in the rule set. The original rule set contained some ill-structured rules. Therefore, only new ill-structured rules after the insertion was counted, and the number of ill-structured rules were recorded for each subject.

5.3 Results and Interpretations

The study used a critical significance level, alpha of 0.05, for testing differences in means. All three hypotheses were supported with significance below this alpha level. The summary of the experimental results is provided in Table 1.

H1 tested the difference in overall quality of modification correctness between the traditional approach and the graph-based approach.
The OCM group performed 23.6 percent more correctly (the mean score was 2.333) than the traditional group (the mean score was 1.625). In fact, 10 out of 24 subjects scored the perfect 3 points in the OCM group, and only three subjects scored the perfect 3 points in the traditional group. The difference was statistically significant (p=0.003).

H2 tested the difference in number of ill-structured rules in modifying the rule set by two groups. In this experiment, three types of ill-structured rules (adjacency, subsumption, and duplication) were actually produced by the subjects. Therefore, the number of these rules produced was counted for each subject. Using the OCM diagram, ill-structured rules are easy to detect; therefore, it was expected that the subjects using OCM created few ill-structured rules in modifying the rule set. As expected, only eight subjects made eight ill-structured rules in the OCM group. On the other hand, fifteen subjects made twenty ill-structured rules in the traditional group. The difference in means between two groups was statistically significant (p=0.015).

H3 tested the difference in number of unreliable rules in modifying the rule set by two groups. In this experiment, three types of unreliable rules (dead-end-IF, conflicting rules, and missing rule) were actually produced by the subjects. Therefore, the number of these rules produced was counted for each subject. The rule set used in this experiment had thirty-three rules and had a rather complex rule structures. Because all three insertions were designed such that multiple rules had to be modified for each insertion, the potential for unreliable rules was high. In fact, fourteen subjects in the traditional group made thirty-six unreliable rules as compared to eight subjects with thirteen unreliable rules in the OCM group. The difference in means between two groups was statistically significant (p=0.005).

### 5.4 Discussion

In this experiment, subjects in the OCM group were asked to draw their own OCM diagram based on the rule set. Because of this arrangement, the OCM group took 10 minutes longer (37 minutes on average) to complete the task than the traditional group (27 minutes on average). After the evaluation of the experiment results, we found that some incorrect modifications were caused by incorrectly drawn OCM diagrams. Because the drawing of an OCM diagram from a rule set can be automated, a machine-drawn OCM diagram should be provided in the future experiment. This will allow us to measure the effect of OCM on RB maintenance more accurately.

Despite these problems, the OCM group still performed significantly better than the traditional group. The result of this experiment clearly validates the potential of the graph-based approach for the RB maintenance.

### 6. Conclusion

The degree of maintainability of a software product is a function of its understandability and consequently its adaptability. This in turn is dependent on the software development techniques adopted. Good software development techniques make software much easier to understand and maintain. Much effort in expert systems research has been
focused on initial building of rule bases, leaving its maintenance unexplored. This is partly due to the fact only a few published systems have reached the maintenance phase. However, even these systems under operations prove that the maintenance of rule bases is not a simple matter originally expected: the system eventually requires an overhaul, and extensive re-write for the maintenance unless it is equipped with a well-structured maintenance method.

The rule base maintenance is challenging when the expert systems are intended to support decision makers in organizations. Unlike conventional transaction systems, decision support systems developers in general adopt a prototyping methodology where new knowledge augments continuously during the system evolution. The experiment in this study indicates that a graph-based approach increases understandability and adaptability of rule base. This preventive methodology will lead toward a better rule base maintenance by complementing existing knowledge validation techniques.

References

APPENDIX: Experiment Rule Set

This system evaluates applicants for a research institute and determines their appropriate rank (G, P1-4) for their qualification.

ABBREVIATION:
NQ: not qualified; H: high school; A: associate degree;
B: bachelor degree; M: master degree; P: Ph.D.;
LNG: number of languages; LQ: low qualification; HQ: high qualification;
I_Q: initial qualification; QC: rank qualification class; NA: no rank appropriate;
G: general rank; P1-4: professional I-IV level rank;
EXP: international work experience; PUB: number of professional publication;
RECOM: final recommendation.

ATTRIBUTE DOMAINS:
AGE=[0..100]: integer; LNG=[1..10]: integer.
EXP=[0..25]: integer; PUB=[0..100]: integer;
QC={G,P2, P2,P3, P3,P4}; DEGREE={H, B, M, P, Others};
I_Q={LQ, HQ, NQ}; RECOM={G, P1, P2, P3, P4, NA}.

RULE SET:

r1: if AGE<25 then I_Q=NQ
r2: if AGE>50 then I_Q=NQ
r3: if 24<AGE<51 and LNG=1 then I_Q=NQ
r4: if 24<AGE<51 and LNG>1 and DGREE=Others then I_Q=NQ
r5: if 24<AGE<51 and LNG>1 and DGREE=H then I_Q=LQ
r6: if 24<AGE<51 and LNG>1 and DGREE=A then I_Q=LQ
r7: if 24<AGE<51 and LNG>1 and DGREE=B then I_Q=LQ
r8: if 24<AGE<51 and LNG>1 and DGREE=M then I_Q=HQ
r9: if 24<AGE<51 and LNG>1 and DGREE=P then I_Q=HQ
r10: if I_Q=NQ then RECOM=NA
r11: if I_Q=LQ and DEGREE=H and AGE<36 then RECOM=G
r12: if I_Q=LQ and DEGREE=H and AGE>35 then RECOM=NA
r13: if I_Q=LQ and DEGREE=A and AGE<36 then RECOM=G
r14: if I_Q=LQ and DEGREE=A and AGE>35 then RECOM=NA
r15: if I_Q=LQ and DEGREE=B and AGE<36 then QC=G.P2
r16: if I_Q=LQ and DEGREE=B and AGE>35 then RECOM=NA
r17: if I_Q=HQ and DEGREE=M and AGE<40 then QC=P2.P3
r18: if I_Q=HQ and DEGREE=M and 39<AGE<46 then QC=P3.P4
r19: if I_Q=HQ and DEGREE=M and AGE>45 then RECOM=NA
r20: if I_Q=HQ and DEGREE=P and AGE<46 then QC=P3.P4
r21: if I_Q=HQ and DEGREE=P and AGE>45 then RECOM=NA
APPENDIX: Experiment Rule Set (continued)

r22: if QC=G.P2 and PUB<2 and EXP<2 then RECOM=G
r23: if QC=G.P2 and PUB<2 and EXP>1 then RECOM=P1
r24: if QC=G.P2 and PUB>1 and EXP<2 then RECOM=G
r25: if QC=G.P2 and PUB>1 and EXP>1 then RECOM=P2

r26: if QC=P2.P3 and EXP<3 then RECOM=P2
r27: if QC=P2.P3 and PUB>2 and EXP>2 then RECOM=P3
r28: if QC=P2.P3 and PUB<3 and EXP>2 then RECOM=P2

r29: if QC=P3.P4 EXP<5 then RECOM=NA
r30: if QC=P3.P4 and 4<EXP<7 and PUB<6 then RECOM=p3
r31: if QC=P3.P4 and 4<EXP<7 and PUB>5 then RECOM=P4
r32: if QC=P3.P4 and EXP>6 and PUB<7 then RECOM=P3
r33: if QC=P3.P4 and EXP>6 and PUB>6 then RECOM=P4