Applying Reliability Engineering to Expert Systems*

Valerie Barr
Department of Computer Science
Hofstra University
Hempstead, NY 11550
vbarr@magic.hofstra.edu

Abstract

Often a rule-based system is tested by checking its performance on a number of test cases with known solutions, modifying the system until it gives the correct results for all or a sufficiently high proportion of the test cases. However, the performance on the test cases may not accurately predict performance of the system in actual use. In this paper we discuss why this testing method does not give an accurate reliability prediction. We then propose a method for reliability prediction based on coverage data, attained during testing, and an operational profile of the expert system. This extension of software reliability engineering methods into the expert systems realm can lead to better systems as well as a more accurate prediction of system behavior in actual use.

Introduction

Given a rule-base and a set of test cases with expected results, we execute the test cases. The rule-base gives a correct answer for 80% of the test cases. What kind of performance can we predict from the rule-base in actual use? In general, we are interested in the question of how the performance of a system during testing can be used as a predictor of its performance in actual use in the intended problem domain.

Generally the rule-base testing process leads to a statistic indicating the percentage of the test cases for which the system performed correctly. These performance statistics are then presented as if they apply to the entire rule-base, rather than just to the tested sections, which can lead to false predictions of system reliability in actual use. In reality the system reliability indicated by a comparison of actual and expected results is relevant only for the tested sections, while reliability of the untested sections and overall reliability of the system under conditions of general use cannot be predicted by this testing method.

The fundamental weakness of this kind of functional testing is that it does not guarantee that all parts of the system are actually tested. We, in fact, have no information about a section of the rule-base that is not exercised during the functional test and whether it is correct or contains errors. Furthermore, many reliability problems for rule-bases are the result of unforeseen interactions between rules (O'Keefe & O'Leary 1993). A test suite of known cases may never trigger these interactions, though it is important that they be identified in order to correct them before a system is put into actual use.

Accurate reliability prediction must take into account several factors:

- identification of the portions of the rule-base which are actually exercised, or covered, during the testing process;
- the behavior of the system on the test cases;
- the likelihood that each kind of test case will occur in actual use (the operational profile with occurrence probabilities (Lyu 1996));
- a measure of how representative the test set is of the operational profile.

In this paper we briefly discuss a testing approach which tracks rule-base coverage, and then discuss how the different factors cited above can be used to produce a quantitative reliability prediction value for the system under test when used in its intended application domain.

Testing with Rule-Base Coverage

Identification of un-exercised portions of a rule-base can be accomplished by enhancing the functional analysis of the rule-based system with a *rule-base coverage* assessment. This determines how extensively possible combinations of inference relations are exercised during test data evaluation. In the trivial case, with a correct

rule-base and a complete test suite, the test data would completely cover the rule-base, all actual results would agree with expected results, and we could predict that the system would be completely reliable in actual use. In the more usual situation there are errors and incompleteness in the rule-base, as well as inadequacies in the test data. If we judge the system based only on a comparison of actual and expected results, the rule-base could perform well on test data, but contain errors which are not identified due to incompleteness of the test data. This could lead to a incorrect prediction of high reliability in actual use, when in fact this approach does not allow us to make any accurate prediction about the reliability of the rule-base in those areas for which there is an absence of test data.

Frequently, when testing classification systems, a large population of cases is available. However, many of these cases may represent situations which are easy to classify and similar to each other. Furthermore, running all available cases may be extremely time consuming for a large classification system. A random selection of test cases may give statistical confirmation that the system works properly for the tested situations, but may not cover all types of situations.

Our approach carries out structural analysis of the rule-base using five rule-base coverage measures to identify sections not exercised by the test data. This testing approach allows for clear identification of both incompleteness in the test data and potential errors in the rule-base through identification of sections of the rule-base that have not been exercised during functional test and may not be necessary or are incorrect. An incomplete test set can be supplemented with additional cases chosen from the available population, guided by a series of heuristics and the coverage analysis information (Barr 1997). This makes it possible to improve completeness of the test suite, thereby increasing the kinds of cases on which the rule-base has been tested. Alternatively, if there is no test data which covers certain parts of the system, it is possible that those sections should not remain a part of the system at all.

Rule-base testing with coverage measures is based on a graph representation of the rule-base, using a directed acyclic graph (DAG) representation. During construction of the DAG, pair-wise redundant rules, pair-wise simple contradictory rules and potential contradictions (ambiguities) are identified. After the DAG is constructed, static analysis reports dangling conditions, useless conclusions, and cycles in the rule-base. The rule-base can then be modified to eliminate or correct any static problems.

Next dynamic analysis of the rule-base is done using test cases. As test cases are processed, one or more of several rule-base coverage measures (RBCMs) can be applied in order to determine the quality of the test data supplied. Additional information about the rule-base and its testing can also be used by the system tester to guide the selection of additional test data. The tester would start by providing sufficient test data to satisfy the simplest functional measure (conclude each class of the system) and proceed to the more difficult structural measures. Finally, if the user is not able to provide sufficient data to attain the desired degree of rule-base coverage (according to the selected criterion), the tester can use the DAG representation to synthesize data, which can then be reviewed by an expert to determine if the data represents a valid case in the problem domain.

This testing approach, described more fully in (Barr 1996), has been implemented in the TRUBAC tool (Testing with RUle-BAse Coverage) (Barr 1996; 1995).

A Metric for Rule-Based Systems

The graph representation employed serves as a suitable foundation for a path metric to measure the complexity of the system and the success of the testing process. The graph imposes no particular execution order on the rules, and it represents all the logical relations that are inherent within the rule-base. However, graph-based metrics such as McCabe's cyclomatic complexity metric (McCabe 1976) cannot adequately determine the number of execution paths in a rule-base. The actual number of execution paths is based on the logical relationships in the rule-base (full details can be found in (Barr 1999)).

The total number of execution paths represents the maximum number of test cases needed for complete coverage of the rule-base according to the strongest rule-base coverage measure (**All-edges**). However, usually the actual number of data sets needed will be less than the number of execution paths, since often, particularly in diagnosis systems, one test set may cover a number of execution paths to different goals. For reliability prediction, the path metric allows us to quantify how much of the system has and has not been covered during the testing process.

Reliability Prediction

Reliability prediction for a rule-based system must be based on

- the behavior of the system on test cases,
- a measure of how well the test data covers the rule-base, based on the coverage analysis and the path metric

- the likelihood that each kind of test case will occur in actual use (this is a subset of the operational profile (Lyu 1996) for the system, with the occurrence probability of each operation)
- how representative the test set is of the operational profile

Assumptions

In order to clarify, we first make some assumptions about the system under test, the test data, and the general population. We distinguish between

- I, the input space which represents the entire population for which the rule-base might be run
- I_D, the pool of data that is available as potential test cases
- I_T, the set of test cases that are actually run through the system during testing

Note that $I_T \subset I_D$, where I_D is generally large and repetitive, so that it is not feasible to use all the cases in I_D as test data. We then assume that

1. it is possible to group the cases in I_D into operations such that each operation contains one kind of case, and all cases in an operation will execute the same inference path within the rule-base. (Here we use the term *operation* in the reliability engineering sense, meaning a grouping of runs that have identical input states, so that only one run of the type is sufficient to test the corresponding portion of the system (Lyu 1996)).
2. I_T is created by selecting one test case from each of the operations found in I_D
3. Therefore, each test case in I_T corresponds to precisely one path in the rule-base.

We note that the third assumption implies that if the rule-base incorporates logic for all possible scenarios in the general population, then the degree of representativeness of the test data and the degree of coverage of the rule-base should be the same.

We also will use the following notation

- O represents the percentage of distinct types of operations in I which are represented in I_D and I_T
- B represents the performance of the rule-base on the test data (percentage correct behavior)
- R represents the reliability predicted, which quantifies the percentage of correct runs that will be achieved in actual use of the system
- C represents the degree of path coverage achieved by the test data (in percentage of paths covered) relative to the path metric
- t_i represents the i^{th} test case
- l_i represents the likelihood of occurrence in I (i.e. in the general population) of an operation that will use the same inference chain (execution path) as is used by test case t_i

General Problem

Consider the situation in which we run N test cases through a rule-base with the following results:

- the test cases achieve 100% coverage of the rule-base (C=100%)
- the actual results generated by the rule-base are 80% correct (B=80%).
- given our assumption of a 1-1 relationship between test cases and paths in the rule-base, this implies that 20% of the rule-base handles cases incorrectly.

However, this situation does not allow us to predict 80% correct performance of the system in actual use. The full path coverage implies that the test data was fully representative of the actual population. However, we still must consider the likelihood of occurrence in the population of the cases that were handled correctly by the rule-base and those that were handled incorrectly.

If 95% of cases in the actual population will be handled by the part of the system that works correctly, then we could predict reliability which will be better than the 80% accuracy achieved by the test data. On the other hand, if only 50% of cases in the actual population will be handled by the part of the system that works correctly, then we could predict reliability that will be much worse than the 80% accuracy achieved by the test data.

In general we expect that, while we will not achieve complete coverage of the system, the section that is covered will correspond to the most likely situations in the population. The portions of the system that are not covered during testing will generally correspond to the least likely situations in the population precisely because it is much more difficult to find test data for cases which are rare in the general population.

Next we consider a somewhat more complicated case. Assume we still have 80% correct performance of the system on the test data (B=80%). However the test data covers only 75% of the paths in the rule-base (C=75%, O=75%). However, those cases are likely to occur 90% of the time in the general population. (That

is to say, our test cases represent 75% of the distinct situations in the operational profile. However, if you selected 100 random cases from the general population, 90% of them would correspond to the operations represented by our test cases). This implies that, in actual use, 10% of the cases that will be presented to the system are from the pool of possible cases (25% of the possible cases) that were not represented by our test data. If we want to generate a safe lower bound on expected reliability, then we have to assume that these cases will operate incorrectly in actual use of the system, since the system was never tested on them. We would like to be able to generate a reliability prediction for this kind of situation, which will be a function of the behavior on the test data, the coverage of the rule-base, the representativeness of the test data and the likelihood of the cases represented by the test data.

Total Path Coverage

We return to the simple situation, with $B = 80\%$ correct behavior, $C = 100\%$ coverage of the rule-base, and $O = 100\%$ of the kinds of cases in I represented by the test data. If we assume that each operation has the same occurrence probability in I then we predict reliability in actual use that will be the same as the behavior of the system on the test data.

If we instead assume that the test cases do not have the same occurrence probability, then the reliability prediction changes. Assume there are 10 test cases ($t_1 \ldots t_{10}$), of which 8 are handled correctly by the system (B=80%). If t_1 and t_2 have occurrence probability of .15 each, t_9 and t_{10} have occurrence probability of .5 each, and all other cases have occurrence probability of .10, we can compute the reliability prediction by computing the sum

$$\sum_{i=1}^{10} l_i * c_i$$

where l_i is the occurrence probability of the ith test case and c_i is 1 if the actual result agreed with the expected result for the ith test case and is 0 otherwise. If the system behaves incorrectly on t_9 and t_{10} then we predict reliability of 90% although only 80% of the test cases were handled correctly. However if the system's correct behavior is on all cases but t_1 and t_2 then we predict reliability of 70%, lower than the 80% correct behavior on the test data.

The assumption of a one-to-one correspondence between operations and paths in the rule-base allows us to shift the occurrence probability figures onto the paths. Then, in the simple scenario in which all paths are executed, we simply sum the occurrence probability values for all paths for which the answer given by the system was correct and use the resulting value as the reliability prediction. Usually, however, we expect that not all paths will be executed during testing.

Incomplete path coverage

If not all paths are executed by the test data, assume that any path not executed during testing will give an incorrect result in actual use of the system. Consider a scenario in which I_T represents 75% of the kinds of cases possible in I ($O = 75\%$). Given the assumptions above, we expect coverage of 75% of the paths during testing (C=75%). Assume that the answers generated during testing are correct for 80% of the test cases (B=80%)[1]. If all cases in I_T have equal occurrence probability then we predict reliability of .6 (60% correct behavior), since of 100 random cases selected from I, 25 may be wrong because they use the untested portion of the rule-base, and an additional 15 will be wrong because they utilize the portion of the rule-base which gave incorrect results during testing.

Next we consider that not all cases have equal occurrence probability. Assume that there are 100 operations (types of cases) in I, and a corresponding 100 paths in the rule-base. Further assume we have only 75 test cases representing 75 of these 100 operations[2], which have high occurrence probability and are found in the population 95% of the time. That is, out of 100 random cases from I, 95 of them will fall into the 75 operations represented by our test cases, with multiple cases falling into some operations. Only 5 of the 100 random cases will be examples of some of the remaining 25 operations.

The rule-base is basically divided as follows:

- RB_{NC} is the portion of the rule-base not covered during testing. For safe reliability prediction, we expect this portion to fail in actual use.
- RB_C represents the portion of the rule-base that handles cases that are included in the test set, divided into:
  - RB_{CC}, the portion of the rule-base that behaves correctly on a subset of the test cases
  - RB_{CI}, the portion of the rule-base that behaves incorrectly on a subset of the test cases

Out of 100 cases drawn from I, 5 would be handled by RB_{NC}, and 95 would be handled by RB_C. We assume the existence of an oracle that determines if the

[1] Partial coverage with complete correctness within the covered section is equivalent to complete coverage with partial correctness.

[2] Information about the makeup of the population may come from the expert, who may not be able to provide test cases for all operations

result given by the rule-base for a test case is correct or that we have *a priori* knowledge of the expected result for each test case. We also assume that we know the occurrence probability value for each of the paths in RB_C (possibly provided by the expert). Given our assumption of a 1-to-1 correspondence between test cases and paths, this is really a probability l_i that each path i will be executed. Therefore our reliability prediction for RB_C (and prediction of correct behavior overall by the system) is

$$\sum_{i=1}^{\#cases} l_i * c_i$$

where, as before, c_i is 1 if the actual result equals the expected result along path i and is 0 otherwise.

To demonstrate how this computes a reliability prediction, we consider two scenarios.

Scenario 1 We first consider a situation in which O=75%, C=75% and B=80%. Furthermore, we let the total likelihood that a case will be handled by RB_{NC} be .5, with .95 likelihood that a case will be handled by RB_C. Assume that there are 100 paths total in the rule-base and that one path represents an extremely popular case, say 10% of the entire population. That is, 10% of all cases run through the system will execute this particular path. We also assume that the system gives the correct answer for the popular case (it is in RB_{CC}).

Out of 100 random cases from I, 5 run through RB_{NC} (which contains 25 paths) and, presumably, give a wrong answer. The remaining 95 cases run through RB_C (made up of 75 paths). The test cases gave 80% correct answers and were run only through RB_C. Given our assumption of a one-to-one correspondence of test cases and paths, this means that, of all paths in RB_C, 60 will give a correct answer (80% of 75 paths). Therefore, of the 95 random cases handled by RB_C, 77.85 will actually be handled correctly, and we predict .7785 reliability for the rule-base in actual use.

We can see this as follows: The 75 paths in RB_C have a total occurrence probability of execution of 95% of 100 test cases, of which 10% belongs to one path and 85% is divided equally among the 74 remaining paths. So the likelihood of execution of each of the 74 paths is .0115. Another way of viewing this is that, of the 60 paths that give a correct result during testing, one path will handle 10% of cases during actual use and each of the remaining 59 paths will handle 1.15% of cases. Therefore, we expect a correct result in $.0115 * 59 + .10 * 1$ cases, or 77.85 out of 100 cases, for a reliability prediction of .7785.

Scenario 2 In this case we leave the above scenario unchanged except that we assume that the popular case is tested but is handled incorrectly by the rule-base (the path for the popular case is in RB_{CI}). Therefore we predict correct performance along 60 paths, each of which has a likelihood of execution of .0115, for a reliability prediction of approximately .69, significantly lower than the .8 that might be inferred if we simply looked at the 80% correct behavior on the test cases.

Conclusions

The reliability we can expect from a rule-based system in actual use can be significantly different than its behavior on a set of test data. The method discussed allows us to compute a reliability prediction without running repetitive test cases, avoiding the approach of running similar cases in order to get statistical confirmation. If a case executes correctly, then its occurrence probability in the operational profile of the general population contributes to the performance prediction figure. Using coverage information, number of paths, and representativeness in the fashion described allows us to use limited data from the population and the system under test to compute a safe reliability prediction figure.

References

Barr, V. 1995. TRUBAC: A tool for testing expert systems with rule-base coverage measures. In *Proceedings of the Thirteenth Annual Pacific Northwest Software Quality Conference.*

Barr, V. 1996. *Applications of Rule-Base Coverage Measures to Expert System Evaluation.* Ph.D. Dissertation, Rutgers University.

Barr, V. 1997. Rule-base coverage analysis applied to test case selection. *Annals of Software Engineering* 4.

Barr, V. 1999. Applications of rule-base coverage measures to expert system evaluation. *Journal of Knowledge Based Systems to appear.*

Lyu, M., ed. 1996. *Handbook of Software Reliability Engineering.* Los Alamitos,CA: IEEE Computer Society Press. chapter 5, 167–218.

McCabe, T. 1976. A complexity measure. *IEEE Transactions on Software Engineering* SE-2(4):308–320.

O'Keefe, R., and O'Leary, D. 1993. Expert system verification and validation: a survey and tutorial. *Artificial Intelligence Review* 7:3–42.