

# Utilizing Validation Experience for System Validation

## Rainer Knauf

Faculty of Computer Science  
and Automation  
Ilmenau Technical University  
PO Box 100565, 98684 Ilmenau  
rainer.knauf@tu-ilmenau.de  
Germany

## Avelino J. Gonzalez

School of Electrical Engineering  
and Computer Science  
University of Central Florida  
Orlando, FL 32816-2450  
USA

## Setsuo Tsuruta

School of Information Environment  
Tokyo Denki University  
2-1200 Musai-gakuendai, Inzai  
Chiba, 270-1382  
Japan

## Abstract

This paper adopts the idea of using knowledge gained by various validation sessions over time with a validation technology developed previously. The work is designed to reduce the human involvement needed to apply this technology. It introduces the reuse of test cases with the "best solution", discovered in previous validation sessions. This reduces the number of test cases to be solved and rated by the experts within the validation process. By reducing the workload of the involved experts, the costs of validation can be reduced. Moreover, this approach may compensate for possible shortages of expertise available for the validation process.

## Introduction

Validation and refinement of an intelligent system has turned out to be a key issue for using these systems in practice. Here we refrain from sketching former works and results in this field and refer to (Knauf 2000) for a comprehensive overview.

Here we present an idea that arose while critically considering the methodology of KNAUF et.al. ((Knauf, Gonzalez, and Abel 2002)) from a practical point of view. Obviously, useful validation knowledge delivered by merely using an AI system is wasted. Instead, a very expensive process to re-acquire this knowledge by computing and solving test cases has been proposed.

Validation should be considered an on-going process. For one thing, what is currently considered "gospel truth", could one day be the subject of historical amusement (e.g., flat earth theory prior to 1492). Intelligent systems must be continually or periodically validated to ensure correctness vis-a-vis the latest thinking.

However, under practical conditions after a long-term usage of an AI system, very little is likely to change from one validation session to the next. Therefore, a full-fledged validation effort, including a panel of validation experts, is not constantly required. However, this implies that the knowledge used in validation, namely the set of test cases, must persist from one validation exercise to the next. Therefore, a way to store, manage, and maintain this knowledge is required for any practical approach to validation.

The concept of maintaining a permanent database of test case files could provide a vehicle for long-term management and improvement of the validation process for an intelligent system. This validation knowledge base (*VKB*) has been suggested by Tsuruta (Tsuruta et.al. 2002). When combined with the validation framework developed by Knauf et.al. (Knauf, Gonzalez, and Abel 2002), it clearly supports the rule base refinement process.

The afore-mentioned validation procedure, as developed so far, covers five steps: (1) test case generation, (2) test case execution, (3) evaluation of results, (4) validity assessment, and (5) system refinement. These steps can be performed iteratively, where the process can be executed again after initial improvements have been made.

Within this procedure, the experiences of the validation process are recorded in a new knowledge base that is the result of the system refinement process. See (Knauf et.al. 2002), for a discussion of the refinement technique. This experience is thereby recorded

- rather implicitly, since it is compiled into the language of the rules and
- therefore, the original test cases with their best solutions are not available any more after the refinement.

The test case generation procedure consists of a first part that is based on a logical analysis of the rule base structure and the input/output behavior it performs. This part leads to a so-called *quasi-exhaustive set of test cases* (*QuEST*). Thus, the set *QuEST* doesn't depend on the circumstances of the system's application. The second part, on the other hand, uses validation criteria to reduce this (usually huge) set *QuEST* down to a so-called *reasonable set of test cases* *ReST* that is used for the test case experimentation. Finally, *ReST* determines the resulting validity statements.

Different system users might have different validation criteria and thus, different test case sets will be generated. Nevertheless, the knowledge gained by the validation of a system could and should be reused for the validation of another system's copy.

This would have the effect of limiting the workload of the human experts involved in the validation process. Since this is the most costly factor within the entire procedure, this issue might well become the key to making the validation procedure practical.

One approach to keep the knowledge gained in a validation process explicit and reusable for further validation sessions has been introduced by Tsuruta (Tsuruta et.al. 2002). Here, the authors propose a *Validation Knowledge Base*, which is basically a library of test cases used in validation sessions so far.

The basic idea is to keep the *best solution* provided in a time-consuming Turing Test – like interrogation of both the system and the human expert panel (Knauf, Gonzalez, and Abel 2002), for test cases and to reuse them for upcoming validation sessions.

## Brief Description of the Turing Test Methodology

The process of intelligent system validation can be said to be composed of the following related steps (Knauf 2000; Knauf, Gonzalez, and Abel 2002), which are illustrated in figure 1:

### 1. Test case generation

Generate an appropriate set of test data that will simulate the inputs to be seen by the system in actual operation. We refer to the pairs [*TestData*, *ExpectedOutput*] as test cases. There are two competing requirements in this step:

- *Coverage* of all possible combinations of inputs – thus expanding the number of test cases to ensure completeness in coverage, and
- *efficiency* – minimizing the number of test cases to make the process practical.

This step is performed in two sub-steps:

- (a) First a *quasi-exhaustive set of test cases (QuEST)* is computed by just analyzing the rules and their input/output behavior.
- (b) Secondly, the huge amount of test cases is limited by utilizing so-called validation criteria. Test cases that don't reach a certain validation necessity degree by considering these criteria will be removed from *QuEST* resulting in of a *reasonable size set of test cases ReST*.

A workable compromise between these constraints is central to both the technique developed so far and the improvements proposed here.

### 2. Test case experimentation

Since intelligent systems emulate human expertise, human opinion needs to be considered when evaluating the correctness of the system's response. Here, we propose a fair evaluation of the correctness of the system's outputs given by imperfect human expertise. This step consists of

- (a) exercising the resulting set of test data (from step 1 above) by the intelligent system as well as by the validating experts and
- (b) anonymously rating all upcoming results, i.e. those provided by the system as well as those provided by the human experts.

in order to obtain and document the responses to each test data by the various sources.

### 3. Evaluation

This step interprets the results of the experimentation step and determines errors attributed to the system and reports it in an informal way. As a side effect of this previous TURING Test – like experimentation, a test case associated competence assessment of the validators can be computed, which is utilized for more objective validity statements.

### 4. Validity assessment

This step analyzes the results reported above and reaches conclusions about the validity of the system. Depending on the purpose of validation, the validity is expressed as

- validity degrees associated to test cases,
- validity degrees associated to system's outputs,
- validity degrees associated to system's rules, and finally
- as a validity degree associated to the entire system.

### 5. System refinement

In order to improve the final system, this step provides guidance on how to correct the errors detected in the system as a result of the previous four steps. Since the validity assessment points out the rules that are "guilty" of some invalidity, and the TURING test experimentation reveals a so-called optimal solution to each test case, we are able to refine these rules with the objective to infer the optimal solution for each examined test case. This, naturally, leads to an improved system.

These steps are iterative in nature, where the process can be conducted again after the improvements have been made. Figure 1 illustrates the steps outlined above.

The benefit of this standardized validation framework is that developers of knowledge-based systems can reference it when describing the validation process to the end user. This may enhance the acceptability of the system. Secondly, this framework attempts to minimize the effort involved in validation of the expert system. This may reduce the cost of system validation to the developer. Lastly, it may help the developer to accurately predict the amount of effort and cost that the validation process will require.

The methodology to implement these steps and its application to a frequently-used kind of rule-based systems is sketched here. A detailed description of all steps as well as the research behind this work can be found in (Knauf 2000) and in (Knauf, Gonzalez, and Abel 2002).

## Brief Description of the Validation Knowledge Base Approach

In (Tsuruta et.al. 2000a), a bi-directional, many-sided explanation typed multi-step validation method (*MMBV*) was proposed. Using this method, knowledge engineers (*KE*) and computers can share validation loads with experts. Thus, the validation load on busy experts is expected to decrease.

However, in order for *KEs* and computers to share more validation load with experts, it is important for *KEs* and

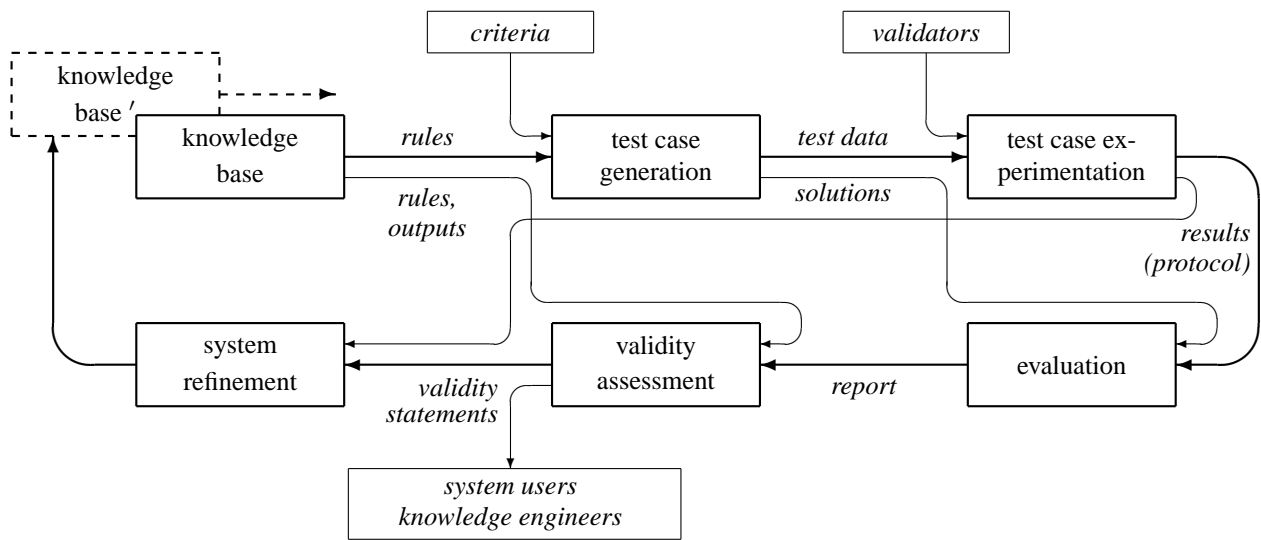


Figure 1: Steps in the Proposed Validation Process (Knauf, Gonzalez, and Abel 2002)

computers to share more validation knowledge with experts, through incorporating validation knowledge into computers. Thus, the concept of Validation Knowledge Base (*VKB*) and a validation approach based on *VKB* has been suggested (Tsuruta et.al. 2000b), (Tsuruta et.al. 2002), which can reuse experts' validation experiences and has the effect of limiting the validation load on busy experts. However, there is a serious problem called the "knowledge acquisition bottleneck". It seems even more difficult to acquire validation knowledge than to acquire domain knowledge because validation knowledge is a kind of meta-knowledge used for validating domain knowledge.

In order to solve this problem, the following approach was suggested in (Tsuruta et.al. 2000b) and (Tsuruta et.al. 2002). This approach is based on the concept, that computers, supported by *KEs* and experts, acquire, validate, and refine validation knowledge (*VKB*), based on the experts' validation data stored in the validation data base (*VDB*) of the validation system. An implementation detail of this concept is described in (Tsuruta et.al. 2000a). Thus, correct and consistent validation knowledge can be easily acquired and incorporated as a Validation Knowledge Base (*VKB*), though such knowledge is difficult to acquire, also because experts are too busy to teach such validation expertise for various kinds of situations. Furthermore, this knowledge is often different or inconsistent depending on experts. This is explained more concretely as follows.

### Experts' validation data base: *VDB*

In the above-mentioned *VKB* approach, the validation knowledge is acquired through experts' validation data in the *VDB* of the validation system such as the one introduced in (Tsuruta et.al. 2000a).

*VDB* includes test cases. A test case consists of test data, test process data, and test results. The test process data include the test schedule and delay status. Test results consist of solutions, explanations, validation results, and comments.

Validation results include evaluators and evaluation values such as *OK* (valid), *NG* (invalid). Comments include any thoughts or explanations with a particular test case.

Thus, validation knowledge is automatically constructed and stored in the *VKB* as described below.

### Validation knowledge base: *VKB*

As mentioned above, experts' validation data in the *VDB* includes test data (problems), solutions, and experts' validation results. They are considered to be experiences or examples of experts' validation knowledge. Therefore, the validation knowledge is acquired from the *VDB* and represented as a case library (Tsuruta et.al. 2002) or as a rule-base (Tsuruta et.al. 2000b). That is, Validation Knowledge Base (*VKB*) can be constructed from *VDB*, by putting the test cases (problems with solutions), into a case-condition part (rule's condition part), and the experts' validation results (expert's evaluation value with comments) into a case-solution part (rule's conclusion part).

For example, as to a Travelling Salesman Problem, a case-condition part (rule's condition part) is a problem (test data) such as a list of visited cities and constraints, accompanied with its solution such as an optimally ordered sequence of visited cities. A case-solution part (rule's conclusion part) is the expert's evaluation value such as *OK* (valid), *NG* (invalid) or as grade of 1 to 5.

Each knowledge piece of the *VKB* has various properties, such as the confidence value (*CV*), many-sided explanation, expert's comment, etc. Further, in order to confirm the correctness of the acquired *VKB*, it has also a property called *Supporter*, which is the list of expert supporters who have accepted the knowledge piece, to trace back from where the validation knowledge originated (Tsuruta et.al. 2002).

The validation and refinement of the acquired validation knowledge is necessary and important for correct validation. In the proposed approach, an acquired new validation

knowledge piece (a new case or a new rule), for example, is checked in comparison with the existing ones in the *VKB*. If an identical one is found, its confidence value (*CV*) is increased, and they are integrated into one knowledge piece. However, if inconsistency exists, the *CV* is decreased (Tsuruta et.al. 2000b), and the experts' validation is retried to check validation knowledge by the responsible experts to be traced back. Other experts can be involved to assist if needed. That is, each piece of validation knowledge is validated and refined by the persons described in its *Supporter* property indicating the persons responsible for the knowledge, namely indicating experts who made or accepted the validation results (Tsuruta et.al. 2002). And, the wrong rule is removed or ignored under the control of *CV* or as a result of the above retrial.

Experts' validation knowledge can be easily acquired and incorporated as correct and consistent Validation Knowledge Base (*VKB*), though experts are too busy to teach or to validate such validation knowledge.

Thus, computers can automatically infer the validation results, utilizing the *VKB*, and can further share the validation load of busy experts, with the help of *KEs* who check and modify the automatic validation results. As a result, the validation load of busy experts is lightened.

### **Incorporation of the Validation Knowledge Base into the Turing Test Methodology**

The objective of the approach is to utilize the experience made in a validation session for future sessions. Therefore, we use a validation knowledge base (*VKB*) to store this experience.

This experience has to be involved in the *test case generation* step as well as into the *test case experimentation* step of the validation framework. The incorporation of the *VKB* into these steps is illustrated in figure 2 and described below.

### **The Content of VKB**

In the context of the *test case experimentation* step, there can be several solutions for each test data. Some of them are provided by the system and others by human experts.

The *VKB* should collect all test data with the solution that obtained the best rating by the expert panel. This solution has to be computed anyway in step 3 of the framework, because it is needed for the formal system refinement (step 5). We refer to this solution as simply the *best solution*.

### **The Involvement of VKB into the Technology**

Since the main objective of introducing the *VKB* is limiting the humans' workload, not all *VKB*'s test cases should be included in the test case experimentation. This would lead to a heavier workload corresponding to the gained experience.

A reasonable way to utilize the *VKB*'s experience is to consider *VKB*'s test cases in the test case generation step of the present session by adding them to the *quasi-exhaustive set of test cases (QuEST)* before computing the *reasonable set of test cases (ReST)*. According to whether or not they meet the validation criteria, they will or will not become

member of the *reasonable set of test cases (ReST)* within the criteria-based reduction process.

### **The Evaluation of VKB's Knowledge**

Test cases of *ReST* that originate from the *VKB*, don't need to be solved again. In the *test case experimentation step* of the framework described in section , a solution is already associated to these test data. It is even the *best solution* of all the ones that have been produced (by humans or the system itself) in a former validation cycle.

The issue here is how to determine the rating of these solutions. There are two ways to do that:

1. Adopt the rating of the test cases obtained in the session before they became a part of *VKB*, i.e. include them along with the rating that is stored in the *VKB*, or
2. ignore the rating in the *VKB* and include these solutions into the regular rating process of the actual session and thus, determining a new rating.

The first of these ways supports the basic objective of this incorporation: Limiting humans' workload in the validation technology. The second way respects the dynamic character of the topical knowledge that is typical for AI systems' application domains.

Since we believe that experience changes over time, as well as by the changing application conditions of the examined systems, we prefer the second way. This is, at least, the actual state of our ongoing discussion on this issue.

We are aware that this has the drawback of reconsidering each test case of the *VKB* within the rating procedure, although it has already been rated before.

On the other hand, the preferred second way has additionally a social satisfaction: For any validation statement that is communicated and used for decisions, there must be a clear responsibility. In other words, it should be possible to trace back where the *validation knowledge* came from.

The used topical validation knowledge in the framework proposed in (Knauf, Gonzalez, and Abel 2002) and (Knauf 2000) is provided by a particular expert panel. Since these experts are responsible for the topical knowledge, they must have an opportunity to provide the test case solutions from the *VKB* with an individual own rating instead of just adopting the rating of former expert panels.

Nevertheless, this reconsideration is still a reduction of the expert's workload, because they do not have to solve these test cases before they rate the different solutions.

### **The Maintenance of VKB**

To ensure the *VKB* really gains experience while being used, it has to be updated during each validation session. Updating, in this context, might mean replacing test cases of *VKB* as well as adding new test cases to it.

In fact, the *experience* of a session worth being kept is the best rated solution to each test data that has been solved, regardless of whether it is originated from the *VKB* or from the actual process of solving and rating test cases. Thus, each test data and the associated *best solution* will be added to *VKB*.

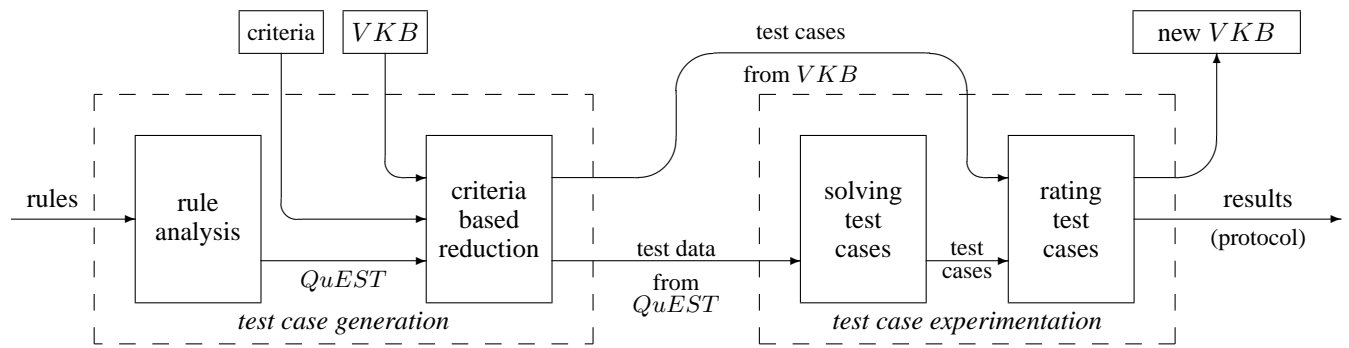


Figure 2: Incorporation of the Validation Knowledge base (*VKB*)

For test cases with inputs that are already part of *VKB*, the corresponding test case in the *VKB* will be over-written with the new corresponding (and higher rated!) solution. Thus, there is no risk of killing experience by removing test cases, because the former test case solution was also considered in the present rating process: Only when a different solution obtained better marks, this *experience* will be over-written by the *new experience*. This allows the *VKB* to keep itself updated with respect to the latest thinking on the domain of interest.

### Summary

This paper has presented a synergistic combination of two research programs to achieve advantages for both.

The historical validation knowledge in the form of test cases (test inputs and expected correct results) can be used to keep an ever-improving benchmark for use in periodic validation of an intelligent system.

This has several significant advantages.

- For one, it lightens the burden on the human experts who are called upon to serve as validators. Such individuals are typically very busy and generally unavailable, not to mention expensive.
- Secondly, it provides a mechanism to continually update and upgrade the test case set to reflect the latest thinking about the domain.
- Thirdly, and somewhat less importantly, it provides with a link to the expert(s) who supported the knowledge used in each test case. This can be done automatically via the Turing Test.

As of this time, no tests have been carried out to validate the concept. We hope to undertake such an effort in the near future.

### References

- Knauf, R. 2000. *Validating Rule-Based Systems – A Complete Methodology*. Habilitation Thesis, Ilmenau Technical University, Faculty of Computer Science and Automation, ISBN 3-8265-8293-4 Aachen: Shaker.
- Knauf, R.; Philippow, I.; Gonzalez, A.J.; Jantke, K.P.; Salecker, D. 2002. System Refinement in Practice – Using

a Formal Method to Modify Real Life Knowledge. Kohlen (ed): *Proc. of the 15<sup>th</sup> Internat. Florida Artificial Intelligence Research Society Conference 2002 (FLAIRS-02)*, Pensacola Beach, FL, USA, pp. 216–220, Menlo Park, CA: AAAI Press.

Knauf, R.; Gonzalez, A.J.; Abel, T. 2002. A Framework for Validation of Rule-Based Systems. *IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics*, vol. 32, # 3, pp. 281–295.

Tsuruta, S.; Onoyama, T.; Kubota, S; Oyanagi, K. 2000a. Validation Method for Intelligent Systems. Etheredge / Manaris (eds): *Proc. of the 13<sup>th</sup> International Florida Artificial Intelligence Research Society Conference (FLAIRS-00)*, Orlando, FL, USA, pp. 361–365.

Tsuruta, S.; Onoyama, T.; Kubota, S; Oyanagi, K. 2000b. Knowledge-based Approach for Validating Intelligent Systems. Kern (ed): *Proc. of 45<sup>th</sup> Internat. Scientific Colloquium (IWK-00)*, Ilmenau, Germany, Technical Univ. of Ilmenau, pp. 769–774.

Tsuruta, S.; Onoyama, T.; Taniguchi, Y. 2002. Knowledge-Based Validation Method for Validating Intelligent Systems. Kohlen (ed.): *Proc. of the 15<sup>th</sup> Internat. Florida Artificial Intelligence Research Society Conference 2002 (FLAIRS-02)*, Pensacola Beach, FL, USA, pp. 226–230, Menlo Park, CA: AAAI Press.