How the Testing Techniques for a Decision Support System Changed Over Nine Years

A. Terry Bahill, *Fellow, IEEE*, K. Bharathan, and Richard F. Curlee

***Abstract*—A decision support system has been under development since 1985 to help speech clinicians diagnose small children who have begun to stutter. This paper describes how testing of the system evolved during these nine years. Testing included: (1) having an expert use and evaluate it, (2) running test cases, (3) developing a program to detect redundant rules, (4) using the Analytic Hierarchy Process, (5) running a program that checks a knowledge base for consistency and completeness, (6) having five experts independently critique the system, (7) obtaining diagnoses of stuttering from these five experts derived from reports of children who had been evaluated for possible stuttering problems, (8) using the system to expose missing and ambiguous information in 30 clinical reports, and (9) analyzing the dispersion and bias of six experts and the decision support system in diagnosing stuttering. When using the final system, three clinicians with widely differing backgrounds produced diagnostic opinions that evidence little variability and were indistinguishable from those of a panel of five experienced clinicians.**

I. Introduction

Development of a decision support system to help speech clinicians diagnose and manage preschool children at risk for a stuttering disability began in 1985. Following eight years of testing and modification *Childhood Stuttering: A Second Opinion*™ was commercially released in 1993 as a diagnostic decision aid. Speech-language clinicians may vary widely in their diagnoses of stuttering in young children, but discussing such cases with experienced clinicians greatly reduces this variability. When using Childhood Stuttering: A Second Opinion, clinicians with different training and experiences arrived at diagnostic opinions that were indistinguishable from those of a panel of five experienced clinicians. In effect, using Second Opinion allows inexperienced clinicians to "discuss" cases of incipient stuttering with a panel of experts, a process that should increase the reliability of their diagnoses in the real world.

Manuscript received August 1, 1993; revised April 24, 1994, and September 17, 1994. An earlier version of this paper was published in thr *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, October 17–20, 1993, Le Touquet, France, Vol. 5, pp. 12–17. This work was supported by the National Institute of Child Health and Human Development Grant R44 HD26209 and by Bahill Intelligent Computer Systems. The contents of this paper are solely the responsibility of the authors and do not necessarily represent the official views of the National Institute of Child Health and Human Development.

A. T. Bahill is with the Department of Systems and Industrial Engineering, University of Arizona, Tucson, AZ 85721 USA.

K. Bharathan is with Bahill Intelligent Computer Systems, Tucson, AZ 85704-1822 USA.

R. F. Curlee is with the Department of Speech and Hearing Sciences, University of Arizona, Tucson, AZ 85721 USA.

IEEE Log Number 9414488.

Previous papers [1], [2] have presented a history of the system and discussed the use of knowledge engineers, shells, techniques for extracting and representing knowledge, outputs of the system, and methods for dealing with uncertainty. They also described how an expert's thought processes changed as he became more aware of the implicit algorithms, heuristics and strategies that he uses in manipulating his knowledge. It was suggested that the changes described for this expert could be generalized to other experts. This paper will discuss a variety of testing techniques, some simple and some complex and the changes in testing techniques that occurred over nine years.

Publications about validating general software systems appeared in the late 1980's [3], while those relating to decision support systems [4] began to appear in the early 1990's. A number of works on verification, validation and testing of knowledge-based systems [5]–[11] have appeared since.

The term *testing* is used in this paper to refer to all facets of determining whether or not the system is functioning as it should. (Adelman [5] used *evaluating* in a similar manner.) It includes both verification and validation. Verification procedures involve checking to make certain that a system performs according to its specifications; that there are no mistakes in the coding. Validation checks consist of ensuring that a system does what it is supposed to do; that its specifications are correct.

There are several types of validation, and the following discussion of them is based on the work of Adelman [5]. Face validity refers to qualitative assessments that a system seems to do what it is supposed to do. Such assessments are based on the judgements of one or more experts in the domain who have used the system. External validity indicates how well a system agrees with established ways for doing the same task. It is usually established by comparing the output of a system with the opinions or decisions made by an external human expert. In assessing predictive validation, a system is used to predict conditions or outcomes that are later verified. A system that evidences high inter-expert validity should help foster a consensus among its users. Confidence validity refers to the confidence that users express in the system. In Turing validity, an expert diagnoses a problem and produces some output advice and a knowledge-based system diagnoses the same problem and produces its output advice. Then a human tries to differentiate between the two outputs. If they are indistinguishable the system has passed a Turing Test [11]. This paper describes techniques that were used for evaluating these and other validity criteria.

The testing techniques to be discussed are organized chronologically, like a case study, instead of following traditional Introduction, Methods, Results, Discussion format. At the time of our system's initial development in the mid 1980's, such knowledge-based systems were commonly called *expert systems*, because they seemed to emulate the input-output behavior of an expert. As people gained experience with knowledge-based systems they realized that these systems did not behave like experts: e.g., such systems do not know what they do not know. For example, if a chicken were presented to Mycin, it would not have known that its knowledge was inappropriate for a chicken and would have proceeded in using its knowledge to diagnose the chicken's disease. Within a few years, therefore, the preferred name and preferred role for knowledge-based systems became *expert advisory systems*. This change explicitly acknowledged that such systems were supposed to give advice, just as an expert would, within a limited domain. The term expert advisory system is used in this paper to refer to the systems we evaluated during this period. During the 1990's, as people gained experience, it became apparent that these systems were not very good at acting like expert advisors. Rather they were best suited to help humans solve problems. Hence their name was again changed to reflect their role as *decision support systems*. Therefore decision support system is used to refer to the systems tested during this later period.

II. Testing the First System

In 1985, the first system was tested by having the expert use it and tell us how he felt about it, what seemed right or wrong. This method was unsatisfactory in many ways, not the least of which was the time it required of the expert. It was not worthwhile, for example, to have him run through the entire system, answering all the questions, every time a minor error was corrected in the knowledge base. As a result, a testing technique that used test cases was developed.

III. Testing the Second System

In 1987, a series of test cases were used to assess a second expert advisory system named Stutter. To create a test case the expert ran the system and answered all the questions as they applied to a child he had evaluated for a suspected stuttering problem. After the expert advisory system presented its diagnostic conclusions and advice, the intermediate knowledge was saved into a disk file. Then, using a normal text editor, those values that were derived by the inference engine were eliminated, leaving only the human's answers to the system's questions. Subsequent files were similarly generated and their sets of answers were saved under different names. Each such file became a test case, because Stutter could use the data in a file as responses to its questions.

In time, many test cases were accumulated and used for testing over the life of the project. Every time a rule was added or modified, each test case was loaded into the system and used to look for run-time errors. If no new questions had been added to the knowledge base, the system's advice was immediately available. If there were new questions, appropriate answers were added to update each test case. These modifications often required more knowledge from the expert but sometimes this could be obtained through a telephone call. Thus, use of test cases that could be updated as needed lowered the demands on the expert's time and usually allowed testing of the system without further input from the domain expert.

Using real test cases provided by an expert is but one way to exercise a knowledge base. Test cases can also be fabricated by knowledge engineers or provided by intended end users. Those created by a knowledge engineer are often more parsimonious because more rules can be exercised by fewer test cases. Typically, an expert, a knowledge engineer and an end user are like three blind men describing an elephant: each focuses on different aspects of the system. The test cases of each will exercise different but overlapping portions of the knowledge base; therefore it may be wise to always gather test cases from all three sources.

Next a computer program was created that used these test cases to test the expert advisory system more systematically [12]. The firing of each rule was recorded as each test case was run. A rule was said to succeed if all of its premises evaluated true and the action specified in its conclusion was performed. Rules that were never found to succeed for any test case were flagged as probable mistakes and were brought to the attention of the human expert. Of course, a rule that never succeeds during such testing may not be an error. Some rules may be intended for unusual or infrequently occurring cases that were not exercised by the test cases that were used. Conversely, rules that succeed for every test case may also be mistakes. If a rule is always true it may be better to replace it with a fact; however, there are control rules that must always succeed. Consequently, this technique is useful in screening a system's rules for potential errors; but a human must decide whether the rule in question is appropriate or not.

The best way to test an expert advisory system, in our opinion, is to have the domain expert run the system and create a representative sample of test cases. Next, determine which rules never succeed and which always do. Our experience indicates that a test case for every five rules in the knowledge base may be sufficient (i.e., 20 test cases for a 100 rule system) even though clearly more test cases are better and having many more test cases than rules would be best. For those rules that do not succeed, the expert should provide test cases that he thinks will make them succeed until most do succeed. Then, the knowledge engineer should try to determine the reason that the remaining rules are not succeeding. It should also be noted that some test cases are not real. The expert for our system characterized some test cases as cartoons that had idealized, exaggerated or stereotyped signs and symptoms of stuttering. To increase the number of test cases, he took a number of existing test cases and systematically changed those answers that should not change the system's output and looked to see if they did. Thus these types of test cases reflect an expert's conceptualization of a domain rather than real cases from the domain.

The knowledge base of our first system was uneven in its coverage. There were, in fact, both redundant questions and areas of knowledge that were not covered. In retrospect, these

shortcomings likely resulted from the knowledge extraction techniques that had been used. Therefore we used the Analytic Hierarchy Process (AHP) to elicit and organize the knowledge of our expert [13], [14]. This process arranges the knowledge into a hierarchy. In our case the knowledge was first divided into rules that dealt with information obtained from an examination of a child and rules that dealt with information acquired from a case history interview. Next, the rules pertaining to each of these sources of information were divided into areas, which were then divided into subareas. Eventually, knowledge was broken down to individual questions, which were further decomposed into their possible answers. An example of how the tool is used will be illustrated with Question 11 from the system called Stutter.

Please describe the within-word repetitions that were observed during your examination of the child.

1) smooth, evenly paced, effortless
2) louder, or increasing in pitch
3) faster in tempo, or unevenly spaced and stressed
4) marked by interruption in voicing or airflow
5) vowels in the syllables being repeated sound like a "schwa" vowel instead of normal vowels.

The answer has five possible alternatives, and it is necessary to know which are the most important. So the expert has to make comparisons of each pair. The AHP chart of Table I shows the comparisons our expert made for this question. For example in completing the entry for row "Part-2" and column "Part-5", the expert indicated that Part-2 is moderately more important, **2.6**, than is Part-5 for diagnosing stuttering. Parentheses around numbers indicate that the item at the top of the chart was rated more important than the item to the left. The expert for our system said that the AHP process led him to organize and systematically reconsider the relative importance of the knowledge that he uses in diagnosing stuttering. He believes that using this process helped to focus his attention during the knowledge extraction process and enabled him to formalize patterns of diagnostic signs and symptoms into relevant object-attribute-value relationships. With traditional knowledge extraction techniques, an expert often thinks about one area for a while and then about another. With nothing to guide the thought processes, lacunae can develop. The analytic hierarchy process forces an expert to make exhaustive comparisons of attributes so that every attribute is compared to every other attribute at each level. This process helps to prevent areas of knowledge from being omitted from a decision support system.

Expert Choice, a software package that incorporates the analytic hierarchy process, was used to build our system; however, any of several other multiobjective decision analysis techniques would likely have worked just as well [4], [15]. Expert Choice forced the expert to *systematically* evaluate the relative importance of each object, attribute and value in the knowledge base, both independently and in all possible combinations. This process forced him to clarify, explicitly, the significance or weight he placed on each specific element of diagnostic information he gathers in arriving at a diagnosis. After an AHP matrix has been filled out, Expert Choice computes the priority vector. This vector shows the relative importance of each item in the matrix. These vectors were used in assigning certainty factors for rules in the knowledge base [13]. Expert Choice also provides (for free as it were) a measure of inconsistency to detect violations of numerical and transitive consistency. If the computed inconsistency index was less than 10 percent, then the pairwise comparisons were considered consistent; i.e. the priority vector was deemed insensitive to slight variations among the elements of the comparison matrix. If a computed inconsistency index exceeded 10 percent, the pairwise comparisons were considered inconsistent, and the task was repeated by the expert. The inconsistency index is important in pointing out inter and intrasubject differences. For example, our expert normally had inconsistency indicies much less than 10 percent. If the inconsistency index of a particular matrix jumped to 15 or 20 percent, we asked, "Should we give up or redo it?" Sometimes he would redo the comparisons; other times he would say he was too distracted and would quit for the day. Although this expert was usually quite consistent providing inconsistency indicies of 1 to 3%, others have often had inconsistency indicies greater than 10 percent. One value of this measure is identifying inconsistencies in an expert's knowledge at an early stage in the knowledge acquisition process, thereby saving substantial time in the testing phase.

TABLE I
AHP MATRIX FOR QUESTION 11

	Part-2	Part-3	Part-4	Part-5	Vector
Part-1	(5.8)	(7.4)	(9.0)	(4.2)	0.031
Part-2		(2.6)	(4.2)	**2.6**	0.143
Part-3			(2.6)	4.2	0.265
Part-4				5.8	0.481
Part-5					0.079

Scale of Relative Importance	
1	Equally Important
3	Moderately more important
5	Strongly more important
7	Very Strongly more important
9	Extremely more important

IV. TESTING THE THIRD SYSTEM

The third expert advisory system, which was named Expert Stuttering Program (ESP), was ready for testing in 1989. It was the first to be tested with a special program, Validator, that we wrote to help find mistakes in knowledge bases [16], [17]. By this time, of course, there were many publications about verification and validation of particular knowledge-based systems, which Jafar and Bahill referenced [16]. *Validator* assesses the consistency and completeness of a knowledge base. It checks for syntactic errors, unused rules, unused facts, unused questions, incorrectly used legal values, redundant constructs, rules that use illegal values, and multiple methods for obtaining values for expressions and systematically indicates potential errors to the knowledge engineer.

A. Verification With Validator

Verification, or building the system right, ensures that a system correctly implements specifications and determines

how well each prototype conforms to design requirements. It guarantees product consistency at the end of each phase (with itself and with previous prototypes) and ensures a smooth transition from one prototype to another.

Validator checks both the syntax and semantics of a knowledge base and brings potential errors to the attention of the knowledge engineer, who has the task of fixing such errors. Validator has four verification modules: a preprocessor, a syntax analyzer, a syntactic error checker, and a debugger.

1) *The Preprocessor:* Syntactic errors can cause a production language (which is used here to refer to AI languages as well as expert-system shells) to misinterpret a knowledge base and consequently to alter its syntactic structure, leading to semantic errors. Validator's preprocessor performs a low-level syntax check. Detecting such errors saves knowledge engineers and experts much time, frustration, and grief. This type of checking is dependent on a system's production language and ameliorates the shortcomings of the language's compiler. Validator builds internal representation structures of the knowledge base, which its other three modules analyze for syntactic compliance.

2) *The Syntax Analyzer:* Many syntactic errors are caused by misspellings, typographical errors, or ill-formed knowledge-base structures. Therefore, Validator's syntax analyzer creates alphabetical listings of the expressions in the knowledge base according to their categories: goals, rule premises, rule conclusions, questions (with legal values), and facts (with values). This list comprises a knowledge base dictionary that makes it easier for a knowledge engineer to find mistakes. For example, a knowledge engineer, forced to read an entire knowledge base, may be hard pressed to remember if a hyphen or an underscore is to be used to tie two words together. But if the two constructs are adjacent in a list it is easy to see an inconsistency.

3) *The Syntactic Error Checker:* People can easily detect inconsistent expressions if they are presented in pairs rather than in large collections, but they need to rely on machines for detecting global and indirect inconsistencies. The syntactic error checker looks for syntax that, while legal, produces unspecified behavior by the production language's compiler. For example, "out-of-range values" (such as incorrect usage of reserved words) usually escape detection by the complier and the knowledge engineer. Validator detects these errors in the contexts in which they occur.

Expressions get their values from facts, from user responses to questions, or from the conclusions of rules. There are four basic types of values. (1) Legal values are acceptable answers to questions. (2) Utilized values appear in rule premises and allow the rule premise in which they appear to be evaluated to true. (3) Concluded values appear in rule conclusions and are set for an expression when a rule using that expression succeeds. (4) Assigned values are assigned to expressions with facts or certain commands specific to the production language.

Utilized Versus Legal Values: Legal values guard against typographical errors and help in abbreviating long answers. If no legal values are provided, a system accepts any response as a valid answer, so typographical errors and wrong responses may escape detection. Even if errors are detected, effective error-recovery procedures are time-consuming and increase the size of the knowledge base. It should be noted that providing legal values will not prevent a knowledge engineer from using out-of-range values in a rule. For example, Mycin-derived production languages do not check a knowledge base to ensure that only legal values have been used; these languages only check user responses to questions to see if they match the legal values specified by the knowledge engineer. Legal values are related to questions, not to rules or facts. Illegal values are common in knowledge bases and often result in the failure of rules using such values.

Unused Legal Values: The syntactic error checker searches the premises of rules, looking for declared but unused legal values, which it flags as potential errors. It also lists all unasked questions. Unused legal values are common in knowledge bases. Many result from errors, others are remnants of old constructs that were put into the knowledge base by mistake or were incompletely removed. Deleting such constructs reduces the size of the knowledge base and speeds inferences during the use of a system. If Validator searched the following knowledge base, it would note that the legal values for coat of animal are {hair, feathers, beads} but the utilized values are {hair, feathers, scales}. Validator would point out that it is not possible to get a value of beads for coat of animal and would indicate that this is likely a mistake. It would also point out that one legal value, scales, has never been used, and would suggest that this may also be a mistake.

```
goal = identity of animal.

if coat of animal = scales
then type of animal = fish.

if coat of animal = hair
then type of animal = mammal.

if coat of animal = feathers
then type of animal = bird.

if type of animal = lizard
then identity of animal = Gila monster.

question(coat of animal)=
'What is the coat of animal?'.

legalvalues(coat of animal)=
[hair, feathers, beads].
```

Utilized Versus Concluded Values: If the values used in the premises of rules do not match the values used in the conclusions of those rules, the rules will fail. In the above knowledge base, the utilized values for type of animal are {fish, mammal, bird, lizard}, whereas the concluded values are {fish, mammal, bird}, which indicates that it would be impossible for this system to conclude that a lizard is a type of animal.

4) *The Debugger:* Debugging is a tedious, difficult, time-consuming, and costly process of finding and correcting errors

in the knowledge base. On many occasions, the presence of errors is discovered during developmental testing but finding them depends on the knowledge engineer's intuition, experience, and common sense. However, computer-aided debugging tools are more reliable because they reduce the possibility of human errors. Validator's debugger checks for rules that use variables, negations, and unknowns. It regards the knowledge base as a closed world, and assumes that all the information about the domain is captured in the knowledge base. Thus, all possible rules and axioms should be either implied or implicitly modeled in the knowledge base.

Variable-Unsafe Rules: Variables can be used in both the premises and the conclusions of rules. They act as symbolic place holders. Each construct that uses variables is logically equivalent to the large set of constructs that could be obtained by replacing those variables with suitable terms. A backward-chaining rule that has variables is variable-safe (closed) if: (1) Its conclusion is homomorphic to a fact or a consequence in the knowledge base; (2) Variables that appear as attributes of an expression in a conclusion also appear as attributes of an expression in the rule's premise; (3) Variables that appear as attributes of an expression in a premise also appear as utilized values in previous premises of the same rule or as attributes of an expression in the conclusion; (4) Variables that appear as concluded values of a rule also appear as utilized values in the rule's premise. A knowledge base is variable-safe if its set of rules evaluate to true, no matter what values are substituted for its variables. Variable-unsafe rules usually lead to infinite loops that violate the closed-world assumptions.

Illegal Use of Negations: Negations can be used only in the premises of rules. A rule that has a negation in its conclusion violates closed-world assumptions, because a false fact is not explicitly declared in the knowledge base, but is inferred from not being able to conclude a given value for an expression.

Illegal Use of Unknowns: The most common origin of unknown is the result of a user's response to a question. Unknown can also be concluded as a result of the unsuccessful firing of all the rules that concern an expression. However, like negations, unknown can be used only in the premises of rules. A rule that concludes unknown for an expression violates the closed-world assumption, because an unknown fact is not explicitly declared in the knowledge base; rather, it is inferred from not being able to conclude a value for an expression.

B. Validation With Validator

Validation, building the right system, ensures the consistency and completeness of the whole system. The validation part of Validator has two modules: a chaining thread tracer and a knowledge base completeness module. The chaining thread tracer determines if rules can fire by tracing their connectivity back to the goal. Rules that cannot fire are flagged as dead rules and are brought to the attention of the knowledge engineer. A rule is also flagged as dead if it is the root of a dead tree. Thus, flagging a dead rule may uncover a whole set of dead rules.

One aspect of validation is checking the knowledge base for completeness, that is, attempting to determine if something is missing from the knowledge base. This checking usually proceeds in two ways. The first is direct and involves a knowledge engineer and an expert working together reviewing, refining and analyzing each item in the knowledge base. They check the completeness of every module. They also check the consistency, effectiveness and efficiency of every knowledge base item. Then they review each rule and decide whether to split it, modify it, or delete it from the knowledge base.

The second approach requires a knowledge engineer to work on the structure and representation of the knowledge base. Knowledge base items are analyzed, compared for redundancy, completeness, and correctness of usage. This approach is structured, algorithmic, and more exhaustive than the direct approach. It also uses heuristics that can be automated to produce fast and effective results. Validator allowed us to use this second approach. After all potential errors flagged by Validator had been resolved, we continued testing this expert advisory system using test cases and additional domain experts.

V. Testing the Fourth System

The fourth version of this evolving decision support system, Second Opinion®, was ready for field testing in 1991. It had already undergone testing with all the techniques mentioned above when our knowledge engineer traveled to the universities of four other experts and spent a day with each running the system on real cases. The experts were generally pleased with the performance of the system, but each offered suggestions for changes, which were implemented. Next, the clinical records of 30 anonymous children whose parents suspected them of stuttering were collected and rewritten after deleting any information that might identify the child or his family. Each expert, at a workshop in Tucson, provided a diagnosis for each of the 30 children after reading the rewritten clinical reports. Next, they discussed these rewritten reports and their diagnosis. The latter diagnoses were used to derive a consensus diagnosis for each child and Second Opinion was changed to match the consensus opinions. Unfortunately, the performance of the system was not evaluated quantitatively before its modification.

If there is a collection of input-output data that is known to be correct (sometimes called a gold standard), then a variety of quantitative techniques can be used to help validate a system, many of these are described by Adelman [5]. In effect, the expert panel's diagnoses reflected our effort to develop such a gold standard.

VI. Testing the Fifth System

While testing a fifth version of this decision support system, Childhood Stuttering: A Second Opinion™ in 1992, we discovered that its outputs differed when different experts used it. We asked two experts to use the system based on information each obtained from reading the 30 clinical reports. The output of the system was the same for 10 of these reports even though there were different answers to, on average, one-third of the questions. This suggests that the system is robust to differences attributable to users. The output of the system was highly similar for another 10 reports but differed

significantly on the remaining 10. These findings suggested that some differences might be attributable to the clinical reports. For example, one report stated "Jose felt frustrated." Some experts understood this statement to mean that the child was showing frustration about his disfluency, while others understood it to reflect his frustration with English and Spanish word-finding difficulties. It was decided, therefore, to resolve such ambiguities in these clinical reports before testing the decision support system further.

First, 15 of the 30 clinical reports were carefully edited so that any lacunae, ambiguities, and conflicting statements were removed. Next, 10 additional clinical reports were obtained and edited in a similar manner. These 25 edited reports were then used to test the latest version of the decision support system. It would have been better, of course, to use a few hundred validated clinical reports, but developing these clinical reports was expensive. In 1992 and 1993 more time was spent acquiring and preparing these reports than in developing and refining the rules in the system's knowledge base.

A. New Tools

In Table II diagnoses of four apocryphal experts and that of a decision support system for four clinical reports are displayed. These data are not real but were created to illustrate the use of several statistical tools. Diagnoses of the experts are denoted with lower case letters a, b, c, and d, and that of the decision support system with dss. The four clinical reports are designated by the upper case letters A, B, C, and D.

The numbers in the Diagnostic Score column represent five diagnostic opinions, which correspond to the following descriptions:

1) Little cause for concern about stuttering. There is little reason to suspect that a stuttering problem may be emerging in this child at this time. The types and frequencies of disfluencies that were observed and described are characteristic of those of nonstuttering children.
2) Some cause for concern about stuttering. This child is evidencing some danger signs of incipient stuttering as well as some that are characteristic of nonstuttering children. This pattern of equivocal findings suggests that the child may be at risk for an emerging stuttering problem.
3) Mild concern about stuttering. This child is evidencing relatively consistent signs of early stuttering.
4) Moderate concern about stuttering. This child presents speech and behavioral signs which suggest that stuttering may not be a transient problem.
5) Severe concern about stuttering. This child evidences speech and behavioral signs that may signal the evolution of a severe stuttering problem.

Two quantitative measures of the agreement between the experts and the decision support system, which were originally suggested by Lucien Duckstein, can be used to characterize the dispersion and bias of these diagnostic data. Each diagnosis is denoted with a capital R with a subscript identifying the individual expert. The term dispersion shows how much the diagnosis of each expert varied from those of every other expert. It is computed with

TABLE II
HEURISTIC DATA SET

Diagnostic Score	Name of Clinical Report A	B	C	D
5			b	c d
4			d	dss
3		b d	c dss	a
2	b d dss	dss	a	b
1	a c	a c		

$$\text{Dispersion}_k = \frac{1}{J}\sum_{j=A}^{D}\left(\sqrt{\frac{1}{I-1}\sum_{i=a}^{dss}(R_k - R_i)^2}\right)_j$$

where i runs over the set $\{a, b, c, d, dss\}$, k runs over the set $\{a, b, c, d, dss\}$, j runs over the set $\{A, B, C, D\}$, $I = 5$ (the number of evaluators) and $J = 4$ (the number of clinical reports).

For the data in Table II the dispersions are:

"Expert"	Dispersion
a	1.47
b	1.68
c	1.37
d	1.35
dss	1.06

The term bias shows inclinations of the individuals. It is computed with

$$\text{Bias}_k = \frac{1}{J}\sum_{j=A}^{D}\left(\frac{1}{I-1}\sum_{i=a}^{dss}(R_k - R_i)\right)_j$$

For the data in Table II the biases are:

"Expert"	Bias
a	-1.19
b	0.38
c	-0.25
d	1.00
dss	0.06

These measures can be used to help validate a decision support system. For the data in Table II all the "experts" performed less consistently than did the decision support system, because four the experts had dispersion scores greater than that of the decision support system. The opinions of expert "b", for example, were the least consistent. The data also illustrate that experts may be biased. The diagnosis of expert "a" suggest that he viewed each child's fluency problem as less severe than the other experts, while those of expert "d" were at the other extreme. The decision support system, on the other hand, showed little bias. These data were contrived to illustrate such ideal behavior by a decision support system.

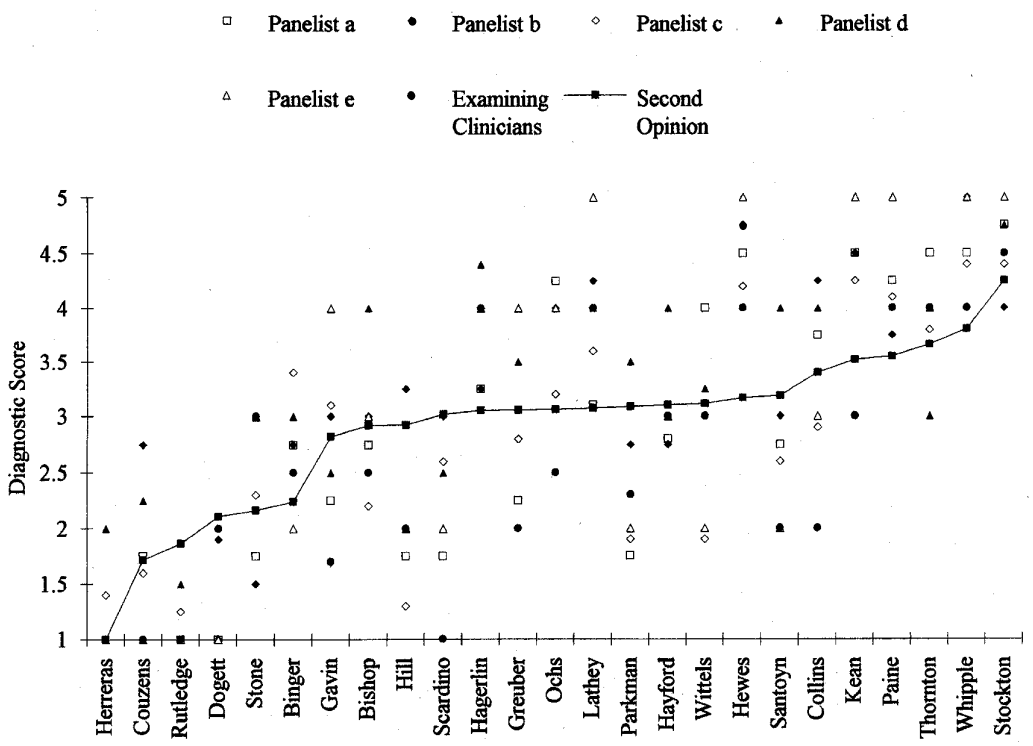


Fig. 1. Diagnostic scores from clinicians and the decision support system for 25 clinical reports. The outputs of the decision support system are indistinguishable from those of the experienced clinicians.

TABLE III
CLINICAL RESULTS

Expert	Bias	Dispersion
a	-0.11	0.76
b	0.17	0.79
c	-0.17	0.71
d	0.50	0.88
e	0.14	0.86
ec	-0.42	0.85
dss	-0.10	0.80

B. The Final System Test

The 25 new clinical reports were mailed to five highly experienced stuttering clinicians who were asked to evaluate the likelihood that each child was stuttering. Fig. 1 displays the diagnostic scores of these five experienced clinicians, of the examining clinicians who prepared the original reports, and of the decision support system for each of the 25 reports. Table III summarizes the statistical results. The five experienced clinicians are represented by the letters *a*, *b*, *c*, *d*, and *e*. The examining clinicians who evaluated each child are designated by *ec*, and the decision support system by *dss*.

Based on the data in Table III, the performance of the decision support system is indistinguishable from that of experienced clinicians. Its dispersion is higher than some clinicians' but lower than others. Its bias is very small. In contrast, the examining clinicians' diagnostic scores do stand out from the panel of clinicians, as is indicated by the large negative bias score. This seems reasonable, because the clinicians who conducted these evaluations had access to data that the other clinicians lacked: they examined the actual children. Furthermore, they likely reviewed video tapes of their evaluation and discussed difficult cases with other clinicians before arriving at a diagnosis and writing their clinical reports. Therefore their diagnostic opinions may be the most valid. The decision support system's diagnoses were designed to fall between those of the examining clinicians and the mean of the panel of experienced clinicians. The low bias and dispersion of the decision support system indicates its robustness.

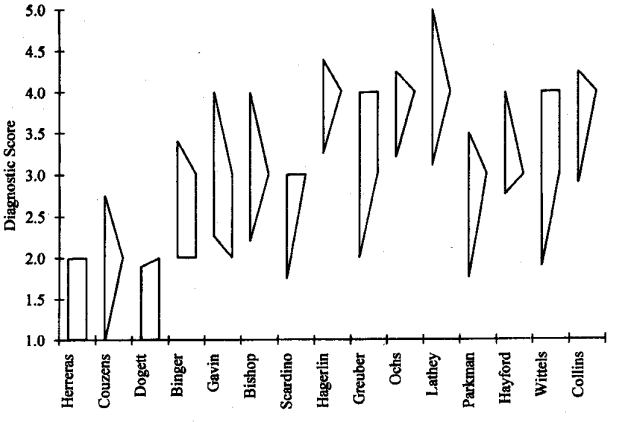


Fig. 2. Range of diagnostic scores for 15 clinical reports before (the left side of each polygon) and after (the right side of each polygon) their discussion by a panel of experienced clinicians. Discussion typically reduced variability in diagnostic scores.

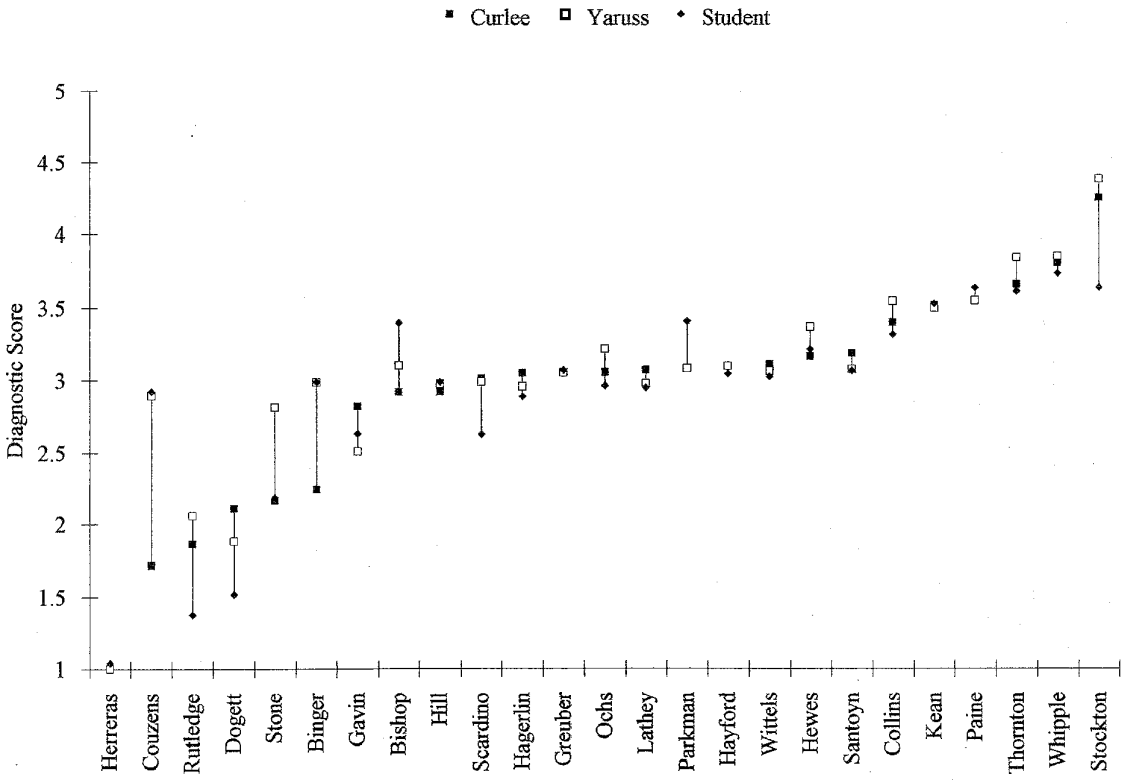


Fig. 3. Diagnostic scores from the decision support system when used by three clinicians having vastly different backgrounds. The variability in scores is small.

The Kendall coefficient of concordance for our five experts is 0.83. This suggests good agreement among the clinicians given that their diagnostic decisions were based solely on written clinical reports. Clinicians would never diagnose an actual child solely on the basis of a written report. Thus, the performance of these five clinicians after reading descriptions of a child's evaluation provides indirect support for their use as experienced, expert clinicians.

When experienced clinicians have an opportunity to confer and discuss their diagnostic opinions, differences in opinion that they may have usually decrease. Fig. 2 shows the range of diagnostic scores the panel of experts assigned to 15 clinical reports before (the left side of each polygon) and after (the right side of each polygon) they discussed these cases. At the time of their discussions, panel members were shown both the mean and range of diagnostic scores for each case. The intended purpose of these discussions was to clarify information about a case, not to develop a consensus, but the range of diagnostic scores was smaller after the discussions. This reduction in range may be due to panel members considering additional diagnostic factors during their discussions than were considered when their initial diagnoses were given. It may also reflect, in part, some panel members' decisions to modify their initial diagnostic scores to more closely approximate the panel's mean score. One purpose of Second Opinion is to allow an inexperienced clinician to "discuss" findings obtained from a diagnostic evaluation with a panel of experts and to learn the diagnostic opinions of the panel of experts. Such "discussions" are intended to increase the reliability of inexperienced clinicians' diagnoses.

The output of Second Opinion is robust. Three people used it with the same 25 clinical reports described above. One was the chief scientist for our decision support systems, who has had over 25 years of experience evaluating children with fluency problems. The second was a doctoral candidate in Communication Sciences and Disorders at Syracuse University who had acquired substantial previous experience with the decision support system The third was a master's degree student at the University of Arizona who had little experience evaluating young stutterers and no previous experience with any decision support system. Their diagnostic opinions, shown in Fig. 3, evidence little difference. The case about which there was the most disagreement, Couzens, led to further review of that clinical report and the identification of ambiguities that may have caused this discrepancy. Use of Childhood Stuttering: A Second Opinion, appears to promote diagnostic opinions that are indistinguishable from those of a panel of five experienced clinicians regardless of a clinician's training and background. One significant finding is that the diagnosis of three people with widely differing backgrounds evidenced less variability when using the decision support system than did those of five experienced clinicians rating the same clinical reports without the decision support system.

As a penultimate test, the system was given to 30 students in an Expert Systems class at the University of Arizona. It is now undergoing its ultimate test as it is used by customers. Most of the concerns of both students and customers involve the installation procedure. No one yet has pointed out errors or suggested improvements in its knowledge base.

A number of factors are likely to affect the performance of clinicians on tasks that have been described above. For example, if they complete 25 evaluations at one sitting, they may produce occasional spurious outputs. Three of the 225 data points in Figs. 1 and 3 were obtained from re-evaluations. The original data points were statistical outliers, so the clinical reports were returned to the clinicians with a request to repeat their evaluation. No one was told why the request was made, and in all three cases the revised diagnosis was markedly different and was no longer an outlier.

This decision support system is meant to assist, not replace, inexperienced speech-language clinicians. One thing a human does that a computer system does not do is track a child's behavior over time, looking for improvement or a worsening of signs and symptoms. In the future we intend to build a version of this system for teaching that will include explanations and pertinent references for each question asked. Such systems would appear to be useful in training clinical skills as well as in supporting the skills of inexperienced clinicians.

VII. Conclusion

It is difficult to detect errors in a decision support system. Therefore, much effort was devoted to the knowledge extraction process to insure that errors did not creep into the knowledge base. Because all such knowledge came from and was captured by fallible humans, it is unreasonable to expect a knowledge base to be free of errors. Therefore, several tools and procedures were developed to help detect errors in a decision support system's knowledge base. Each was designed to work with existing tools, and each added additional complexity to testing procedures, and, we think, credibility to the decision support systems' performance. Finally we compared the output of the system to the evaluations of human experts. When using Childhood Stuttering: A Second Opinion, three clinicians with widely differing backgrounds in stuttering produced diagnostic opinions that evidence little variability and were indistinguishable from those of a panel of five experienced clinicians.

Acknowledgment

The authors thank S. D. Bellaire and S. Yaruss for helping us test our system. Our panel of experts includes: R. F. Curlee, University of Arizona, M. R. Adams, University of Houston, E. G. Conture, Syracuse University, H. H. Gregory, Northwestern University, and J. C. Ingham, University of California, Santa Barbara.

References

[1] A. T. Bahill, K. Bharathan and R. F. Curlee, "Knowledge extraction changes the way an expert thinks," in *Proc. IEEE Int. Conf. Systems, Man, and Cybernetics*, Chicago, Oct. 18–21, 1992, pp. 917–921.

[2] A. T. Bahill, K. Bharathan and R. F. Curlee, "Making an expert system changes the expert," *Recent Trends in Research, Education, and Applications*, M. Jamshidi, R. Lumia, J. Mullins and M. Shahinpoor, Eds. New York: ASME Press, vol. 4; also in *Proc. 4th Int. Symp. Robotics and Manufacturing*, Santa Fe, NM, Nov. 11–13, 1992, pp. 711–716.

[3] S. J. Andriole, *Software Validation: Verification, Testing and Documentation*. Princeton, NJ: Petrocelli, 1986.

[4] A. P. Sage, *Decision Support Systems Engineering*. New York: Wile, 1991.

[5] L. Adelman, *Evaluating Decision Support and Expert Systems*, New York: John Wiley & Sons, 1992.

[6] A. T. Bahill, *Verifying and Validating Personal Computer-Based Expert Systems*, Englewood Cliffs: Prentice Hall, 1991.

[7] U. G. Gupta, *Validating and Verifying Knowledge-Based Systems*. Los Alamitos, CA: IEEE Computer Society Press, 1991.

[8] M. Ayel and J. P. Laurent, *Validation, Verification and Tests of Knowledge-Based Systems*. New York: Wiley, 1991.

[9] S. Smith and A. Kandel, *Verification and Validation of Rule-Based Expert Systems*. Boca Raton, FL: CRC Press, 1993.

[10] K. Parsaye and M. Chignell, "Intelligent Database Tools and Applications: Hyperinformation Access," *Data Quality, Visualization, Automatic Discovery*. New York: Wiley, 1993.

[11] R. Agarwal, R. K. Kannan and M. Tanniru, "Formal validation of a knowledge-based system using a variation of the Turing test," *Expert Systems with Applications*, vol. 6, pp. 181–192, 1993.

[12] Y. Kang and A. T. Bahill, "A tool for detecting expert system errors," *AI Expert*, vol. 5, no. 2, pp. 46–51, 1990.

[13] R. F. Moller and A. T. Bahill, "A knowledge extraction technique," Chapter 2 in *Verifying and Validating Personal Computer-Based Expert Systems*, A. T. Bahill, ed. Englewood Cliffs, NJ: Prentice Hall, 1991.

[14] T. L. Saaty, *The Analytic Hierarchy Process*. New York: McGraw-Hill, 1980.

[15] K. Parsaye, "Acquiring & verifying knowledge automatically," *AI Expert*, vol. 3, no. 5, pp. 48–63, 1988.

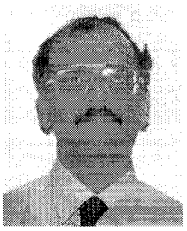
[16] M. Jafar and A. T. Bahill, "Interactive verification and validation with validator," Ch. 4, in *Verifying and Validating Personal Computer-Based Expert Systems*. A. T. Bahill, Ed. Englewood Cliffs: Prentice Hall, 1991.

[17] ——, "Interactive verification of knowledge-based systems," *IEEE Expert*, vol. 8, no. 1, pp. 25–32, Feb. 1993.

A. Terry Bahill (S'66–M'68–SM'81–F'92) was born in Washington, PA, on Jan. 31, 1946. He received the B.S. in electrical engineering from the University of Arizona, Tucson, in 1967, the M.S. in electrical engineering from San Jose State University, San Jose, CA, in 1970, and the Ph.D. in electrical engineering and computer science from the University of California, Berkeley, in 1975.

He served as a Lieutenant in the U.S. Navy, and as an Assistant and Associate Professor in the Departments of Electrical and Biomedical Engineering, Carnegie Mellon University, Pittsburgh, PA, and Neurology at the University of Pittsburgh. Since 1984, has been a Professor of Systems and Industrial Engineering at the University of Arizona. His research interests include systems engineering theory, modeling physiological systems, eye-head-arm coordination, the science of baseball, the system design process, validating expert systems, concurrent engineering, quality function deployment, and total quality management. He has published over 100 papers and has lectured in a dozen countries. His research has appeared in *Scientific American, The American Scientist, The Sporting News, The New York Times, Sports Illustrated, Newsweek, Popular Mechanics, Science85, Highlights for Children, Rolling Stone*, and on the NBC Nightly News. He is the author of *Bioengineering: Biomedical, Medical, and Clinical Engineering*, Prentice-Hall, 1981, *Keep Your Eye on the Ball*: The Science and Folklore of Baseball (with Bob Watts), W. H. Freeman, 1990, *Verifying and Validating Personal Computer-Based Expert Systems*, Prentice-Hall, 1991, *Linear Systems Theory*, (with F. Szidarovszky), CRC Press, 1992, and *Engineering Modeling and Design*, (with Bill Chapman and Wayne Wymore), CRC Press, 1992.

Dr. Bahill is a member of the following IEEE societies: Systems, Man, and Cybernetics, Engineering in Medicine and Biology, and Professional Communications. For the Systems, Man, and Cybernetics Society he has served three terms as vice president, six years as associate editor, as Program Chairman for the 1985 conference in Tucson, and as Co-chairman for the 1988 conference in Beijing and Shenyang, China. He is a Registered Professional Engineer and a member of Tau Beta Pi, Sigma Xi and Psi Chi. He is the Editor of the CRC Press Series on Systems Engineering.



K. Bharathan (M'93) was born in Madras, India, on January 22, 1951. He received the B.A. (Honors) in economics from the University of Delhi, India, in 1973, the M.A. in economics from Jawaharlal Nehru University, New Delhi in 1975, the M.Phil. in economics from the University of Madras, India, in 1979, the Ph.D. in economics from the University of Madras in 1990, and the M.S. in systems engineering from the University of Arizona, Tucson, in 1991.

He was on the faculty of the Madras Christian College, India, from 1976 to 1978, and thereafter on the faculty of the Madras Institute of Development Studies until 1988. He has been a Systems Engineer at Bahill Intelligent Computer Systems, Tucson, AZ, since 1992. His research interests include systems engineering theory, the design, verification and validation of expert systems, and the knowledge extraction process.

Dr. Bharathan is a member of the IEEE Systems, Man, and Cybernetics and Computer Societies, and the American Medical Informatics Association.

Richard F. Curlee received the B.A. in speech from Wake Forest College, Winston-Salem, NC, in 1961, the M.A. in communicative disorders from the University of Southern California, Los Angeles, in 1965, and the Ph.D. in communicative disorders from the University of Southern California in 1967. He has been a Professor of Speech and Hearing Sciences, University of Arizona, Tucson, since 1980.

He has served as a Speech-Language Pathology Clinician at the Children's Speech and Hearing Center, Van Nuys, CA, an Adjunct Assistant Professor at the University of Southern California, Associate Secretary for Research and Scientific Affairs for the American Speech-Language-Hearing Association, and Associate Dean of the Graduate College at the University of Arizona. He is also the Chief Scientist of Bahill Intelligent Computer Systems. His research interests are in the fields of neurophysiological correlates of fluency, disfluency and stuttering and the clinical management of speech-language disorders, especially that of stuttering.

Dr. Curlee is an associate editor for the *Journal of Fluency Disorders*, and was editor of *Nature and Treatment of Stuttering*, College-Hill Press, San Diego, CA, 1984, *Stuttering and Related Disorders of Fluency*, New York: Thieme Medical Publishers, Inc., 1993, and of the journal *Seminars in Speech and Hearing*. He is a member of Phi Beta Kappa, Phi Kappa Phi, Sigma Xi, and is a Fellow of American Speech and Hearing Association.