PERFORMING AND MANAGING EXPERT SYSTEM VALIDATION

Robert M. O'Keefe and Daniel E. O'Leary

ABSTRACT

Assuring the quality of an expert system is critical. A poor quality system may make costly errors resulting in considerable damage to the user or owner of the system, such as financial loss or human suffering. Hence validation is fundamentally important. This paper reviews the issues, methods, and techniques for validating expert systems in presenting a survey and integration of the existing literature. Approaches to defining the quality of a system are discussed, drawing upon work in both computing and the model building disciplines, which leads to definitions of verification and validation and the associated concepts of credibility, assessment, and evaluation. Considerable attention is then given to the issues in structuring the validation process, particularly the establishment of criteria by which the system is judged, the need to maintain objectivity, and the concept of reliability. This is followed by a review of methods

Advances in Expert Systems for Management,
Volume 1, pages 141–176.

ISBN: 1-55938-390-9

for validating both the components of a system and the system as a whole, and includes examples of some useful statistical methods. The issue of managing the validation process is then discussed.

I. INTRODUCTION

Expert systems incorporate human expertise in computer programs to allow these programs to perform tasks normally requiring a human expert. More formally, an expert system has been defined as "a computing system capable of representing and reasoning about some knowledge-rich domain with a view to solving problems and giving advice" (Jackson 1986) and "a computer model of expert human reasoning reaching the same conclusions the expert would reach if faced with a comparable problem" (Weiss and Kulikowski 1984). Examples of developed and implemented systems include R1/XCON (Bachant and McDermott 1983), which configures VAX computers for Digital; ExperTAX (Shpilberg and Graham 1989), developed by Coopers & Lybrand to give advice on corporate tax planning; ONCOCIN (Langlotz, and Shortliffe 1983), which helps doctors determine appropriate treatments for chemotherapy patients; and CLASS (Duchessi, Shawky, and Seagle 1988), a system that supports commercial loan decisions in a bank. Reviews of other systems can be found in Waterman (1986) and Ernst (1988).

Showing that an expert system is in some sense correct is a critical task. An incorrect system may make costly errors or may not perform up to expectations. In either case the decisions generated by the system may be inappropriate or wrong, and if relied upon considerable damage such as financial loss or human suffering may result to the user or owner of the system. For example, expert medical diagnosis systems and income tax systems have encountered implementation difficulties due to concerns over liability of the system's diagnosis.

The purpose of this paper is to survey and integrate the literature on validating expert systems. Accordingly, this paper will discuss definitions for verifying and validating expert systems, provide a basic structure to meet those concerns, and discuss some of the primary approaches. Throughout a fairly liberal interpretation is made of the term *expert system.* Often, expert systems generates notions of if . . . then . . . rule-based systems, but in this paper it will also be used to refer to a broader set of systems that employ other forms of knowledge repre-

sentation and hybrid representation schemes or embed analytic tools in symbolic-based knowledge representations.

II. VERIFICATION, VALIDATION, AND SYSTEM QUALITY

A. A Hierarchic View of System Quality

Verification is defined by Adrion, Branstad, and Cherniavsky (1982) as "the demonstration of the consistency, completeness and correctness of the software." As noted by O'Keefe, Balci, and Smith (1987), "verification means building the system right." Verification is aimed at eliminating errors in the system and making sure that it corresponds to the specification. Adrion et al. (1982) indicate that "validation is the determination of the correctness of the final program or software produced from a development project with respect to the user needs and requirements," O'Keefe et al. (1987) note that "validation means building the right system." Validation is more concerned with the quality of the decisions made by the system.

Computer professionals concerned with development rarely deal with more than verification and validation, so called V&V. However this is really just the first stage in ensuring the quality of the system. An issue that determines the use of the system is that of credibility: the extent to which a system is believable or the extent to which users can put credence in the system. That is, to what extent is the system credible to its users? (Balci 1987). Other issues concern the fit between the system and the user beyond the correctness of the decisions that the system makes. These issues (e.g., Buchanan and Shortliffe 1985) are summarized as assessment (O'Leary 1987) or evaluation (Liebowitz 1986) and include the nature of the discourse between the system and the user, the adequacy and efficiency of the hardware, the quality of the implementation, and the security and documentation of the system. In addition, development typically involves the analysis of questions such as What benefits does the system have? or Does the system do what is required of it? Since the validity of an expert system is often the key to its worth, evaluation is frequently mistaken for validation (e.g., Chandrasekaran 1983).

The relationship among these five aspects of the quality of a system form a hierarchy, shown in Figure 1. Essentially, each depends upon the level below it, and in many cases problems at a lower level will result in

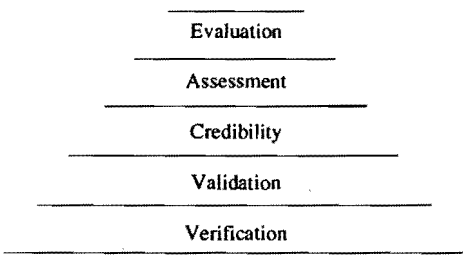

Figure 1. A Hierarchical View of Aspects of Verification and Validation.

problems at a higher level. For example, a system that cannot be shown to be valid may have little chance of being evaluated as worthwhile. Traditionally, different communities have concentrated on different parts of the hierarchy. Contrasted with V&V in computing, model-building professionals, such as operations research scientists, have concentrated on validation and credibility. Often managers, project sponsors, and end users, however, are concerned only with assessment and evaluation of the end product.

B. Verification, Validation, and Credibility

The types of errors that can occur in software can be regarded, at least partially, as a function of the technology used to implement the system. For example, in a rule-based expert system it is possible to produce a knowledge base containing a reasoning cycle (where a chain of inference can lead back to the original premise), and these should normally be eliminated. The technology, in particular the knowledge representation scheme and method of handling uncertainty, establishes much of the basis for verification.

Validation generally is regarded as a more complex task than verification, which is not as dependent on the particular technology. Where a piece of software must perform in a predefined manner, such as a compiler developed for a programming language, measuring validity is relatively straightforward. A compiler that incorrectly compiles any legal code is invalid. A different view of validation stems from the process of model building, particularly the construction of statistical, econometric, and operations research models. A model is a representation of reality that will never be perfect.

Landry, Malouin, and Oral (1983) provide a conceptual framework for model validation. They define five types of validity:

1. *Conceptual:* "the degree of relevance of the assumptions and theories underlying the conceptual model . . . for the intended users and use of the model."
2. *Logical:* "the capacity of the formal model to describe correctly and accurately the problem situation."
3. *Experimental:* "the quality and efficiency of the solution mechanism."
4. *Operational:* "the quality and applicability of the solutions and recommendations."
5. *Data* "the sufficiency, accuracy, appropriateness, and availability of the data."

Often model validation is an attempt to measure the generality of the model (for example, can it perform with different data sets or under differing assumptions?). Typically, model validation is often equated with operational validity—a model is judged by the quality of its solutions—yet as Landry et al. point out, absolute validity based upon accuracy of solutions is a myth. The intended use of a model must be taken into account—a model may be valid for one application but not for another. Hence conceptual validation, and the criteria by which a model is judged, are both very important.

Since a model is never absolutely valid (or for that matter, absolutely invalid), its acceptance is very dependent upon the intended use and user. Hence model builders have recently started to focus more on credibility (for instance, Balci 1987 and Gruhl 1982). It would be expected that the more valid a model the more credible it is, but there is no guarantee that a valid model will be perceived as credible. Although credibility is a useful concept, there is at present no way to measure it. It is known, however, that some of the techniques that can be used for validation have a positive impact on credibility, as discussed later.

C. Assessment and Evaluation

Although the focus of this paper is on validation it is important to understand the rest of the hierarchy. Verity, validity, and credibility do not guarantee a usable, let alone useful, system. A system may be

unusable, for example, because its response time is too slow or the interface is too complex. Such issues are certainly relevant when assessing an expert system, but in general validation should be considered separately from assessment, not just as part of the assessment process (for instance, as done by Liebowitz 1986), since it is a prerequisite for adequate performance. A poorly validated system rapidly giving wrong advice through a beautifully designed interface may be dangerous: the superficial quality of the system may induce unwarranted credibility.

Traditionally, it is promulgated that data-processing systems be evaluated purely on a financial basis, such as return on investment. A feasibility study specifies the costs and benefits of a system, and development is initiated or otherwise based upon these projections. Increasingly for many information system, particularly decision support systems, the benefits are intangible. The value to an organization of improving the decision-making process is often very difficult to quantify (O'Keefe 1989), and hence many systems are accepted for development, and subsequently evaluated, in an ad hoc manner (Hamilton and Chervany 1981, Keen 1981).

Similarly, the benefits of an expert system are often intangible and almost impossible to quantify. Hence formal evaluations that address the costs and worth of an expert system are rare, although this may happen implicitly. Further, to this point in time one of the benefits of many developed systems has been increased familiarity with the technology for both the user group and the developers. As expert systems become more pervasive it is likely that evaluation will become more stringent, and this in turn will create a need for more formal verification and validation.

III. CHARACTERISTICS OF EXPERT SYSTEMS

Different classes of software have particular characteristics that shape the way verification and validation are done. The dominant characteristic of expert systems that makes verification and validation difficult is that an expert system is simultaneously a piece of software *and* a model. Like any software, bugs can cause unwanted and unpredicted consequences, but an expert system is also a model of human knowledge and reasoning, and like all models, as mentioned above, will never be perfect. Even if the software is completely verified and reliable, the embodied model may be in error. Divorcing the software and the model is difficult, if not

impossible, although verification will normally concentrate on software aspects, whereas validation will concentrate on modeling aspects.

Expert system validity will typically depend upon the extent that an expert's activity and knowledge have been captured in the expert system: the extent to which the system is a good model. Thus, as noted by O'Leary (1987), validation is normally concerned with "what the system knows, does not know, or knows incorrectly." A critical issue is the criteria by which the system is judged as valid, and this is discussed in detail later.

Viewing an expert system as a model leads to a number of problems that affect verification and validation. First, one of the findings in the development of expert systems is that understanding the domain is critical, as with most model building activities (Waterman 1986). In many other types of software it is assumed that the program specifications need only be turned over to a programmer, who would then produce the code. However, generally, in order to produce the computer code that is used as an expert system the programmer must become a near-expert in the particular domain (Lethan and Jacobson 1987). Second, the domain defines what are the critical aspects of an expert system developed in that domain: reasoning and knowledge in one domain may not be the same as that in other domains. Third, the expertise that is modeled in an expert system is generally in short supply, is expensive, or is not readily available. This is in contrast to other types of software, where there is substantial expertise available, such as some accounting systems, but again is fairly common in model-building activities.

In addition to the problems generated by the model characteristic of expert systems, three other characteristics pose problems for verification and validation. First, expert systems employ both numeric and symbolic information rather than just numeric information. Because symbolic information is used, techniques typically used for the validation of numeric information may be infeasible, and this leads to a need for new validation methods. A purely quantitative model may produce a number that can be compared against an actual observation—the difference between the two is thus an estimate of accuracy; contrast this with, for example, a page of textual advice representing a tax plan—comparison against an observed plan would require considerable expertise, and even then assessing the difference may be difficult. Kulikowski and Weiss (1982) discuss this problem in the context of the medical diagnosis system Casnet.

Second, expert systems generally are developed using a middle-out design rather than a traditional top-down or bottom-up approach. A middle-out approach starts with a prototype and gradually expands to meet the needs of the decision. Further, expert systems have a tendency to evolve over time as the decisions they model become better understood. Hence, as discussed later, knowing when to actually validate an expert system can be difficult.

Third, expert systems are often used to model tasks for which computer programs have not been previously developed. Therefore, it is likely the task is not well understood prior to the development effort (at least not by the developers), and thus there is no preestablished understanding of the problem.

IV. THE STRUCTURE OF VALIDATION

Validation is thus a complex process dependent upon the criteria on which the system is to be judged and its intended use. Hence the validation process should be properly structured: the means whereby the system will be declared as valid or otherwise should be established at the outset of the process. This paper uses a framework, based on the theory of research methods, to investigate structuring validation efforts. That framework includes *establishing criteria* for validation, *criterion vs. construct* validity; *maintaining objectivity;* and *reliability.*

A. Establishing Criteria

1. Level of Expertise

The simplest approach to establishing a criterion for validating a system is to define the output level of expertise at which the system should perform. It may be required that a system performs at the level of an expert, better than an expert, or at the level of a good trainee. For example, the performance of both ONCOCIN and the infectious disease diagnosis and therapy selection system MYCIN were shown to be reasonably close to experts at the Stanford Medical Center, where the systems were developed (Hickam et al. 1985, Yu et al. 1979a,b); it is sometimes stated that the mass spectrometer analysis program DENDRAL, also developed at Stanford, performed at the level of an organic chemistry Ph.D. (Buchanan, Sutherland, and Feigenbaum 1969).

In some instances it will be required that the system perform better than an expert. This may be achievable, for example, by the system's being more consistent than an expert, not tiring, being able to deal with more data, and being able to solve problems more quickly. Hence, criteria concerning consistency, speed, and other factors may dominate criteria concerning, for instance, quality of solutions. A good example of this is the R1/XCON system (Bachant and McDermott 1983), where previous human performance was very mixed, and introduction of the system established some much needed consistency.

Finally, in other situations, in order for the system to be usable it must function at least at some minimal level of performance. One view of that minimal level is that of a trainee or some other nonexpert level of performance.

Two suggested methods for defining the level of expertise of a system require mention if only so that they can be subsequently disregarded. The first method is to get the expert on whose knowledge the system is based to sign-off on the system, stating that it performs at their level. Basden (1983) discusses some applications in an organization in which this has happened. This is obviously convenient from the viewpoint of legal liability: whether it validates a system is very debatable. The second is to get the system to sit an examination since many professions, such as accounting and medicine, have entrance exams that test basic competence. Such exams, however, generally test breadth, whereas most systems developed so far are narrow in their specialty domain. In addition, typically, such exams are minimal levels of performance rather than measure of expertise within those professions.

2. *Performance Range*

A convenient approach to measuring the level of expertise of a system is to judge its success in solving problems. The system will not necessarily perform at the expert level, although it might, but will perform within some range. The level of performance acceptable to users and sponsors is called the *acceptable performance range.*

Many developed expert systems have explicitly used performance range as an evaluation criterion. Typically, a number of case studies are presented to the system, and the number of correct answers, compared to those of an expert, are tallied. The system is then determined to be, for example, "90% correct," or "95% perfect." Such figures are meaningless:

		State of the expert system	
		System is valid	*System is invalid*
Action	*Accept as valid*	Correct decision	User's risk (Type II error)
	Declare invalid	Builder's risk (Type I error)	Correct decision

Figure 2. Type I and Type II Errors in Validation.

they are simply a function of the cases presented to the system. What is more, in many instances a large number of cases dealt with by an expert are standard, and the leverage of the expert is determined by the difficult and obscure problems that are faced.

As an example, consider the case of a system designed to evaluate firms for whether they will go bankrupt. Typically the success of such a system should be in its ability to find those relatively few firms that will go bankrupt. In any given year roughly 95–98% do not go bankrupt, and hence it is conceivable that a system may, correctly categorize 90–95% of the total without being able to determine more than 25–50% of the firms that actually went bankrupt.

3. Builder's/User's Risk

A more formal way to specify criteria for a performance range developed by Balci and Sargent (1981) is based upon the standard statistical concept of type I and type II errors, as shown in Figure 2. A type I error results if a system is rejected as valid when it is in fact valid; type II error results when an invalid system is accepted as valid. The probability of the first is *builder's risk,* since the effect is to prolong development of an acceptable system, and perhaps even abandon development. The probability of the second is *user's risk,* since the effect may be dramatic for the user who accepts incorrect results, such as loss of life or large sums of money.

For many expert systems it will be impossible to quantify either risk. However, for all systems it should be possible to consider the *relative* importance of each risk. For example, with R1/XCON a high user's risk was acceptable, due to the relatively poor performance of the human

predecessor, but with medical systems user's risk must be shown to be virtually zero. This is, of course, very difficult to do.

For systems with a limited set of outcomes, such as many straightforward classification systems, relative risks can be expressed at the level of actual outcomes. Suppose that an expert system produces a classification as A, B, or C. It may be that certain variations in performance are acceptable but others are not. For example, an incorrect classification of A as B may be acceptable, but A as C may not. Hence in validation it must be shown that prob($A|C$) is zero, but prob($A|B$) can be greater than zero.

B. Criterion vs. Construct Validity

All the various approaches to establishing criteria discussed above are in fact variations of what social science model builders call criterion validity. "Criterion validity is studied by comparing test or scale scores with one or more external variables or criteria, known or believed to measure the attribute under study" (Kerlinger 1973). The attribute is expertise; the criteria are some variation of performance range. An alternative type of validity is construct validity: validation against the theory on which the system is based. As noted by Kerlinger (1973) "the significant point about construct validity that sets it apart from other types of validity is its preoccupation with theory and theoretical constructs."

With the majority of expert systems the developer elicits the knowledge base in a purely empirical manner, where knowledge is treated as something to be iteratively discovered without reference to any underlying theory to guide the investigation. There is no explicit theory against which the system can be validated, and hence criterion validity has dominated expert system validation and will likely continue to do so.

Construct validity may have an increasing role to play, however. Recent developments in qualitative and causal reasoning have resulted in systems based upon first principles derived from an understanding of the causality in the domain being examined (for example, Davis 1984).

There are at least three potential validation comparisons associated with construct validity: first, the comparison between the system and the first principles; second, the comparison of the first principles to a decision maker; third, the comparison of the system to the human expert. Thus, from one perspective, when validating such systems the developer is in

actuality trying to validate the underlying theory, the decision maker, and/or the system.

Construct validity can be investigated by either focusing on the subjective process of the existence of a theory (Did the development of the system include reference to theory?) or investigating the expertise on which the system is based (Does the expert reference theory?). In the case of a system that generates its knowledge from multiple experts, another issue needs to be examined: Are the experts using the same or different underlying principles?

C. Objectivity

Since verification investigates aspects that are not open to subjective appraisal (for instance, a cycle does or does not exist in the knowledge base) objectivity is generally not an issue. On the other hand, in validation, objectivity is critical because the measurement of validity against established criteria may be open to interpretation. Objectivity is really a who issue: who will do the validation? In addition, it is an attitudinal issue. For example, what is the attitude of the validator if it is known that there is a computer giving expert advice.

Ideally, validation tests are built into the software, similar to automated verification tests, and particular procedures are implemented without intervention. For all but the simplest of validation criteria, this is impossible to do. So it is necessary to check the objectivity of the human validator.

1. Programmer Validation

Typically the knowledge engineer is continually verifying and validating the system, based on skills that are brought to the system and gained during system development. However, those efforts may be less than objective due to a number of factors. First, if the developer is short on time or budget, the validation effort may be cut, since it may be seen as an overhead function. Second, if the developer has a vested interest in the system, then letting the programmer instigate the only verification and validation procedures is somewhat analogous to letting the fox guard the henhouse: there is a potential for substantial violations of objectivity.

In order to mitigate such violations, typically research methods emphasize the importance of the independence of the validator. If the model builder is the only validator, there can be conflicts of interest.

2. *Sponsor or End-User Validation*

With conventional software, software engineers often use an acceptance test by the sponsor or end user as the final step in the validation process. If validation criteria are well established, as discussed earlier, then a validation by the sponsor or end user provides evidence that at least the system meets those criteria.

Unfortunately, if a system changes substantially, the notion of an acceptance test may be difficult to implement. Further, as noted in O'Leary (1987), in some cases the end user or sponsor may have insufficient expertise to validate the system. Hence the developers have to establish validity and then build system credibility by professionally reporting the details of the validation process.

3. *Third-Party Experts*

An attractive approach to the establishment of objectivity is to get third-party experts, that is, experts not involved with the development effort, or not even part of the sponsoring organization, to validate the system.

Some researchers, for example Buchanan and Shortliffe (1985), have reported that in some cases external validators may be biased in their validation of a system if they know that the problem solution was generated by a computer. They may expect less from the system or more likely have unreasonable expectations of performance (this is called "the super-human fallacy" by Chandrasekaran 1983). Other biases may also influence evaluators. If a human investigator finds that a problem solution is from a rival expert, or simply one from another school of thought, then that could impact the quality they attribute to the solution. Clearly, this means that blinding techniques must often be used to mitigate these types of violations of objectivity.

D. Reliability

In addition to performing within an acceptable range, a system must be a *reliable* representation of the expert's knowledge. The expert's

knowledge is captured by the knowledge engineer, who then casts his or her representation of that knowledge into a computer program. There are two possible interpretations of reliability here. In the first, total reliability occurs when the knowledge reported by the expert and the actual knowledge of the expert are the same. In the second, total reliability occurs when the knowledge reported by the expert and the knowledge in the computer program are the same. One perspective on this notion of reliability is that the uncertainty of capturing knowledge cascades from one representation to another.

1. An Analytic Model of Reliability in Weights

Based on this, O'Leary (1988b) developed an analytic Bayesian model to investigate the impact of cascaded reliability on expert system weights. In this case the concern is with the relationship between an actual event, a report of the event, and the corresponding weights in an expert system that relate to the evidence. For convenience and specificity the discussion employs the uncertainty representation used in Prospector and AL/X (Duda, Gaschnig, and Hart 1979).

Knowledge in AL/X is represented as a set of rules and weights on the rules of the form:

$$\text{if } E, \text{ then } H \text{ (to degree } S, N),$$

where S and N are numeric values that represent the strength of association between E and H. S and N can be specified from the following likelihood ratios:

$$S = S(E,H) = P(E|H)/P(E|H')$$

$$N = N(E',H) = P(E'|H)/P(E'|H'),$$

where E' and S' are not E and not S and $P(.)$ is the probability measure. S and N are used as logarithms, to the base 10, as, respectively, NW and PW.

Let $E\#$ be the report of the evidence rather than the actual state of the evidence. Since we have $E\#$, we are interested in $S(E\#,H)$ and $N(E\#',H)$. Using the report of information and Bayes' theorem, we can determine that

$$S(E\#|H) = [P(E\#|H \text{ and } E)P(E|H) + P(E\#|H \text{ and } E')P(E'|H)]/$$

$$[P(E\#|H \text{ and } E)P(E|H) + P(E\#|H \text{ and } E')P(E'|H)]$$

$$N(E\#'|H) = [P(E\#'|H \text{ and } E')P(E'|H) + P(E\#'|H \text{ and } E)P(E'|H)]/$$

$$[P(E\#'|H' \text{ and } E')P(E'|H') + P(E\#'|H' \text{ and } E)P(E'|H')].$$

The factors $P(E\#|.)$ and $P(E\#'|.)$ characterize the uncertainty associated with the report of the evidence, that is, the reliability. Denote $S(E\#|H)$ as RS and $N(E\#'|H)$ as RN, where the R indicates the version that includes reliability. Denote PW and NW as RPW and NPW. O'Leary (1989b) provides a detailed analysis of RS and RN. A special case of that model that provides insight into the impact of reliability on the weights will be considered.

Consider the case where $P(E\#|.)$ is not dependent on H and $P(E\#|E) = P(E\#'|E') = r$. In that case,

$$RS = [r^*P(E|H) + (1-r)^*P(E'|H)]/[r^*P(E|H') + (1-r)^*P(E'|H')]$$

$$RN = [r^*P(E'|H) + (1-r)^*P(E|H)]/[r^*P(E'|H') + (1-r)^*P(E|H')].$$

The factor r has a substantial impact on the weights that suggests that reliability be accounted for appropriately. The impact of the reliability factors on the *RPW's* and *RNW's* is illustrated in the following example:

RPW's and *RNW's* for selected $PW = -1.0$

$P(E\|H)$	0.075	0.05	0.025	
$P(E\|H')$	0.75	0.50	0.25	
Reliability	*RPW*	*RPW*	*RPW*	
1.0	−1.0	−1.0	−1.0	*PW* Value
0.9	−0.641	−0.553	−0.398	
0.8	−0.424	−0.337	−0.212	
0.7	−0.260	−0.193	−0.111	
0.6	−0.122	−0.086	−0.046	
0.5	0.0	0.0	0.0	
0.4	0.114	0.072	0.034	
0.3	0.224	0.134	0.061	
0.2	0.334	0.188	0.082	
0.1	0.447	0.235	0.099	
0.0	0.568	0.279	0.114	*NW* Value

Owing to a duality theorem, RPW at reliability r yields an RPW at level $1 - r$. Other results include $RPW = PW$ and $RNW = NW$ when $r = 1$.

V. VALIDATION METHODS

Once criteria have been established, the goals of validation determined, and the various problems addressed, appropriate methods have to be chosen. The criteria, existence of a theory, and pragmatic issues such as the availability of experts and case studies will determine the appropriate methods.

Expert systems can be validated by either examining the individual components, such as the weights or the knowledge base, or by examining the operation of the system as a whole. When testing the system as a whole the system can be treated as a black box simply to determine if it is making the right decisions, or it can be opened up to determine if the line of reasoning is correct, that is, it is making the right decisions for the right reasons (e.g., O'Leary 1988a).

A. Component Validation

1. Rule Validation

A common method of component validation is to directly examine the knowledge base so as to assess the accuracy, representativeness, and validity of individual rules. The process of direct examination is facilitated by the manner in which some expert system shells allow the user to access the knowledge.

Where the knowledge base is too large to fully directly investigate, approaches can be used to choose which rules are more important to examine (O'Leary 1988a). For example, those rules that have the most costly consequences or largest profit generally should be investigated. When using an uncertainty measure, those rules with either the larger or smaller weights should be examined because they have the greatest impact on the solution generated by the system. Another approach is to determine which rules fire the most (or the least) in simulated paths through a rule base. Then those rules can be examined for their quality since they either are frequently or rarely in solutions generated.

2. Weights

O'Leary and Kandelin (1988) develop a number of approaches to validate the actual weights in an expert system. The basic assumption

underlying their investigation is that the behavior of the weights can be investigated using statistical approaches. One approach is designed to investigate for the presence of outliers. Finding weights that are substantially different than the other weights is likely to indicate that those weights should be investigated further.

Another issue concerns the existence of the relationship between different versions of the system. The relationship between the weights in, say, the first and second versions of the system can be used to measure that relationship. If there is some reason to suspect that there is a relationship, say, that there is no change in the way the weights were generated in subsequent versions of the system, then this can be tested statistically, for example, by examining the relationship between the means in the set of weights. Such a relationship might exist because the expert may change the approach to assigning the weights in order to change the behavior of the system.

Still another issue relates to the relationship between multiple weights assigned to the same rule. For example, as discussed previously, in the system of weights developed for Prospector and AL/X there are two interlinking weights associated with each rule, *PW* and *NW*. Since the weights are interconnected, a change in one weight typically should result in a corresponding change in the other weight. The relationships between these weights can also be tested statistically by doing a similar test of means of weights. If the means are different for one weight but not for the other in subsequent versions of the system, this may indicate that the development of the system was not appropriate.

3. *Heuristics*

An expert system, particularly a rule-based system, can be viewed as a collection of heuristics or a single large heuristic. A single rule may be a sensible heuristic in its own right, or, more likely, a combination of rules can be considered as a complete separate heuristic producing a solution given input. A modular system may be composed of a number of such heuristics.

Previous work in validating mathematical heuristics, reviewed by Eglese (1986), provides a method for validating knowledge-based heuristics. Many mathematical heuristics are used where existing optimization methods are too inefficient for solving problems on a regular basis or in real time. In these cases the results from a heuristic can be compared

against the optimum solution, and deviation from the optimum can provide a measure of the goodness of the heuristic. Similarly, where an expert systems works on a problem where complete enumeration of the state space or an optimization method can provide an optimum result, this comparison can be made. Some constraint reasoning systems, such as those developed to produce production plans and schedules (for example, the ISIS system developed at Carnegie-Mellon [Fox 1984]), fall into this category.

Heuristic validation also uses a concept called *worst case analysis,* which is the maximum deviation from the optimum that can ever occur. For some mathematic heuristics the worst case can be proved; for knowledge-based heuristics this is unlikely to be possible, but experimentation with numerous problems, or a detailed analysis of the search mechanisms, may supply an estimate of the worst case.

A complicating factor in knowledge-based heuristics is uncertainty measures, where an uncertain estimate input by the user or produced by another heuristic affects the outcome. If the outcome itself is a measure of uncertainty, this will vary over some range depending upon the estimates input. For example, as the input varies from −1 to +1, the output itself will vary over all or some part of the range −1 to +1. Langlotz, Shortliffe, and Fagan (1986) show how this situation can be regarded as a distribution sampling simulation, where samples of the input produce a distribution for the output. This can then be used to aid assessment of the validity of the heuristic.

4. *Metamodels*

A metamodel expresses the relationships between the elements of a model: it is a model of a model. Where a knowledge base becomes large it can be useful to have a model of the constructs and concepts present: this can then be used to determine conceptual validity.

Rushby (1988) concludes that any rule base should be accompanied by a causal model. Expressed in diagrammatic form, a causal model can be used by developers and experts to check for completeness of the knowledge base. Unlike the previously discussed contingency table, which relates rules to each other, a causal diagram relates the concepts that are expressed as rules. For example, Figure 3 shows two rules concerning automobile fault diagnosis and an associated causal link.

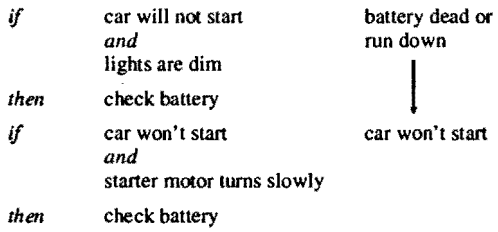

Figure 3. Use of a Causal Diagram to Express Concepts in Rules.

B. System Validation

1. Test Cases

Based on a survey of expert system developers (O'Leary and Watkins 1989), using test cases seems to be the present dominant method for the systemic validation of expert systems. Cases previously solved by an expert are run through the system, or new cases are presented to both expert and system, and the solutions are compared.

There are at least four guidelines that should be followed when selecting test cases. First, the problems to be encountered by the system should be reflected in the cases (Chandrasekaran 1983). If we use terminology from software verification, this implies that there should be a *prescribed input domain:* the boundaries of the input that the system will receive should be specified. Second, a sufficient number of test cases is necessary to elicit the range of parameters necessary to test the system and to be able to establish some statistical measures of significance. However, as noted by O'Keefe et al. (1987) "the issue is the coverage of the test data—that is, how well they reflect the input domain," not the *number* of cases that are used. A sufficient variation in the test problems is necessary to test the range of parameters in the system.

Third, the nature of the problems investigated by the system should help establish the characteristics of the cases. As for the example of a system designed to investigate bankruptcy, in any one year roughly 3–5% of United States firms go bankrupt. Thus, if a system was given test data in proportion to the occurrence of bankruptcy in the actual population (approx 96% not bankrupt and 4% bankrupt) the nonbankrupt firms would flood the system to result in a high success rate.

Fourth, in some domains expert decisions may precipitate actual outcome. O'Keefe et al. (1987) give an example: "Suppose that a bank uses a performance prediction when deciding whether or not to support company X financially. If an expert had decided a year ago that the financial position of company X would be poor in a year's time and thus implemented withdrawal of financial support, the present poor financial position of company X might be due in part to that previous expert position." Choice of test cases is thus difficult, if not impossible, in such domains.

In using test cases there is an assumption that the expert against whom the system is being compared is *always correct,* i.e. if the system differs from the expert then it is wrong. This, quite obviously, is not always the case. One of the authors was involved in the development of a personnel selection system, which underwent extensive validation, where it was realized that in a number of the test cases the previous expert had missed or misinterpreted something, and hence had made an incorrect decision. When the system made a correct decision in these cases, the credibility of the system was greatly enhanced.

In many instances test cases will not be available. Synthetic cases can be produced, but this is dangerous and demands considerable objectivity on behalf of the validators. There is a temptation to make the cases reflect the known strengths of the system.

2. *Turing Tests*

In the classic Turing test, a third-party has access to the output from both machine and human and has to determine which is which. As a validation tool, a Turing test refers to a third-party expert comparing the results from an expert system with those from a human expert. To overcome the objectivity problems discussed earlier, the process should be blinded so that it is not clear which is the computer's and which is the human's.

Test cases are necessary for Turing tests, and the discussion above equally applies. Now, however, *there is no assumption that the human expert is correct:* the third-party expert can compare, rank, or criticize as deemed appropriate. For many expert systems, Turing tests are the most appropriate validation method. They are particularly useful when (a) it is difficult for the developer to assess output on a case study as correct, or otherwise, or make judgments about how it differs from a

human expert (this is often the case when output is holistic and difficult to quantify), or (b) the system must be validated against multiple experts and there is variation among the performance of the experts.

MYCIN was validated using a Turing test methodology (Buchanan and Shortliffe 1985, Yu et al. 1979b). Ten cases were developed and analyzed by 10 experts, including the system. These 100 case results were then evaluated by 8 evaluators, using one of three alternatives on a rating system, to establish a level of performance for each expert. The rating system alternatives were "equivalent" (i.e., the evaluator would have done the same thing), "acceptable alternative," or "not acceptable." Hickam et al. (1985) discuss a similar Turing test validation of the chemotherapy adviser ONCOCIN.

Hansen and Messier (1986) discuss a test of the auditing expert system EDP-XPERT, and a second more extensive test is reported in Messier and Hansen (1992). EDP-XPERT provides a certainty measure for the electronic data processing controls in computerized accounting systems. It produces three measures representing confidence in the supervisory, data-base management, and application controls. In each test, expert auditors and the system produced a measure, and these were compared by the developers. Interestingly, the experts then had the opportunity to produce a second measure given the output from the system. In the first test, where the experts were low-level computer audit specialists (in the second test they were senior experienced specialists), a significant number changed their answer. This may be a useful approach to validating systems that will be used in a supporting role.

Despite its power, the Turing test methodology may be difficult to implement in practice. First, it requires more expert time, and ideally these experts should not have been involved in the development of the original system. Second, comparison against multiple experts can be difficult to measure. Third, blinding the outputs from computer and human expert, which normally has to be done by putting them into a common form, can be very time consuming.

3. *Simulation*

In some instances, particularly real-time control expert systems, the analogy to test cases is connecting the system to a simulation model. Each simulation run is a test case, and different scenarios with various parameter settings can produce a number of different runs. For simple determi-

nistic simulation models, validation via simulation is very powerful. For complex domains, however, a major problem arises: the simulation itself is a model, not perfect, that performs within an acceptable range. Hence modeling problems, such as accuracy and reliability, cascade. An expert system that performs well with a simulation cannot be guaranteed to perform as well with the real system.

4. *Control Groups*

Many expert systems rely upon the combination of human user and system to solve problems, and so the system cannot be validated alone. Where this is the case, a Turing test can be combined with a control group methodology. Cases are presented to two separate groups: those with the system, and those without. The validation process then proceeds as before, although now it is hoped that the group with the system outperforms the control group.

This approach, unfortunately, contains many pitfalls. The two groups may have performed differently irrespective of one group having access to the system, and a small number of case studies may not show up this inherent difference. Complexities in the system may mean that performance with it may only improve over time, or that its effect on performance is negligible until fully institutionalized. Hence a control group approach is normally seen as an evaluation tool used after implementation. For an example of using a control group as an evaluation tool see Hamilton and Chervany (1981).

5. *Sensitivity Analysis*

Where no case studies are available, the validation process is far more difficult. Often developers will verify the system and then simply use credibility as a complete surrogate measure: Is the system credible to the expert, the developers, and the potential users? Yet a few validation methods are applicable where no or few case studies exist, in particular *sensitivity analysis*.

Assume there exists a single case C where intermediate results, the line of reasoning, and the final results are all known to be perfect (or, more likely, are judged by an expert to be reasonable). If C uses inputs $i_1, i_2 \ldots, i_n$, then each can be systematically altered (either individually or in sensible combinations), and the change in output from the system

assessed as reasonable or otherwise by an expert. In many instances, it is easy to generate tests where the output should *not* change given changes in input. For example, if in a financial analysis expert system it is known that a particular figure should have no effect on the results (perhaps since the situation is dominated by other concerns), then the system should be run with this figure set at its extreme values.

6. Comparison against Other Models

In some instances a different type of model, such as an optimization or statistical model, may already exist. Comparison of the system against this model can provide useful insights, for example, Moninger, Stewart, and McIntosh (1988) compare a weather forecasting expert system against a regression model to assess the comparative accuracy of the system. Typically it might be expected that a knowledge-based approach would be able to handle odd or different cases better due to use of specific knowledge.

The availability of induction algorithms in many shells, such as Quinlan's ID3 (1979), means that induction of a rule set may be quite easy to do. Although the induced set may be very limited compared to the rule set crafted by hand following knowledge acquisition, due to lack of examples or limitations in the algorithm used, it does provide an alternative that can give insights into the validity of the acquired set.

7. Line of Reasoning

When validating many systems it is insufficient to validate just the results from the system: it is also necessary to show that the line of reasoning is correct. There are two reasons for this. First, if the user investigates the line of reasoning via facilities built into the shell, then it must be credible, otherwise the result will be disregarded. Second, if the system is to be developed further, the reasoning process must be capable of being scaled up to a larger domain (Chandrasekaran 1983). Line of reasoning can be used as evidence in a Turing test. However, this requires that human experts articulate their reasoning and that it can be presented to third-party experts in a form similar to the explanation facilities of the shell being used.

A more complex approach is to compare aspects of the reasoning process, such as the relative time taken to reason, the amount of data

used, or the number of hypotheses established and rejected. Meservy, Bailey, and Johnson (1986) derived knowledge from experts using protocol analysis, and then the expert's percentage of time spent performing specific processes, such as cognitive processes (e.g., assuming, conjecturing, evaluating, and questioning) was compared to that of the system.

C. Statistical Methods

Although many validations will be entirely qualitative in nature, in other instances a quantitative approach using a statistical model is warranted. Such an approach will almost certainly be necessary when comparing the system to one or more human experts.

Essentially, the validation process can be viewed as the following hypothesis test (O'Keefe et al. 1987):

> H_0: the expert system is valid for the acceptable performance range under the prescribed input domain,

where the alternative hypothesis is that the system is invalid. Hence, until the acceptable performance range has been established and the type of problems the system will handle defined (and thus the future input domain prescribed), then statistical tests cannot be formally employed.

This section gives something of the flavor of the use of statistics, covering confidence interval and consistency measure approaches. In any given situation, one particular approach from the vast array of statistical techniques may be better than others, and nonparametric approaches not discussed here, for example rank correlation, may be appropriate. Mosteller and Rourke (1973) cover many methods. If detailed statistic analysis is used, the validation team should include a statistician: misuse of statistical measures can be worse than no analysis.

1. Confidence Intervals

Where a system produces a single result (for example, a certainty factor representing an estimate of the financial state of a company), comparison against an expert is quite straightforward. If the system's result is X_i, and that of the human expert is Y_i, then the difference between them will be $D_i = X_i - Y_i$. For n case studies, there will be n observed

Table 1. Final Certainty Factors for 10 Case Studies from a System and an Expert

Case	*System*	*Expert*	*Diff*
1	4.5	4.0	0.5
2	3.2	4.5	−1.3
3	−1.4	−2.0	0.6
4	−3.0	−1.5	−1.5
5	2.1	1.0	1.1
6	3.5	3.0	0.5
7	2.0	3.5	−1.5
8	−1.0	1.5	−2.5
9	3.5	4.0	−0.5
10	−4.5	−4.0	−0.5
		mean	−0.51
		sd	1.114

differences D_1 to D_n. The confidence interval for the difference between system and expert is thus:

$$d - t_{n-1,\alpha/2}\, Sd/n, d + t_{n-1,\alpha/2}\, Sd/n,$$

where d is the mean difference, Sd the standard deviation, and $t_{n-1,\alpha/2}$ the value from the distribution with n degrees of freedom. If zero lies within the interval, there is no significant difference between the system and the expert. Note that the acceptable performance range will dictate the specification of α. Table 1 shows an example where 10 case studies have been given to a system and an expert, and the result is a certainty factor ranging from −5 to +5. The mean difference d is −0.51, and so for α = 0.1 and 10 degrees of freedom we have a confidence interval:

$$-1.1557 < d < +0.1357.$$

Since zero lies in the range, the null hypothesis can be accepted. However, a small number of case studies will generally give rise to a large confidence interval half-width, due to the small number of degrees of freedom, so any conclusions have to be carefully interpreted. Certainly in this example d is only just in the range, and the system's results tend to be lower than those of the expert.

Where a single result is not produced, this method can still be used if the output can be assessed as a whole. For example, in a Turing test, if the output of both system and expert are assessed by a third-part expert on a scale of 1 to 10, then X_i and Y_i will be the absolute assessed performance measure.

Where multiple results are produced, simultaneously applying a paired *t*-test to each result is inappropriate since the results may be correlated. O'Keefe et al. (1987) give an example of the correlated multiple-response problem: "In a medical diagnosis system prescribing drug treatment . . . two types of drug can be validly prescribed if each is separately considered as independent—yet that combination of drugs may be unacceptable." Hence it is necessary to produce *simultaneous confidence intervals*. How this can be done is shown in Balci and Sargent (1984).

2. Consistency Measures

Originally developed to determine the consistency among raters, such as legal judges, consistency measures have a number of uses in expert system validation. First, they provide an alternative to confidence interval approaches—an expert and the system can be viewed as two independent raters. More important, they do not require the underlying distribution assumptions that confidence interval approaches do. Second, when comparing a system against multiple experts, the consistency between experts is of concern. Third, they have the advantage over other statistical techniques in that they can deal with categorical scales and are hence appropriate when dealing with classification systems or expert assessments of performance on a discrete scale. Consistency measures are covered in detail in Fleiss (1981) and the discussion here uses the same notation.

The most useful consistency measure is the *kappa statistic*, originally developed by Cohen (1960), which measures agreement on a single category. The weighted kappa (Cohen 1968) measures overall agreement across all categories and was used by Hickham et al. (1985) in the ONCOCIN validation. To explain the use of the weighted kappa, Table 2 shows the agreement between an expert and a system on 20 case studies with three result categories called A, B, and C. Each entry shows agreement as a proportion of the total cases; for example, both the expert and the system classified 50% of the cases as A.

The weighted kappa k is defined as

$$k = \frac{p_o - p_e}{1 - p_e},$$

Table 2. Agreement between System and Expert on 20 Cases with Three Categories

	Expert			
System	*A*	*B*	*C*	*Total*
A	0.5	0.1	0.05	0.65
B	0.1	0.1	0.05	0.25
C	0.05	0	0.05	0.1
Total	0.65	0.2	0.15	1.0

where p_o is the overall proportion of observed agreement, and p_e is the overall proportion of chance expected agreement (the agreement that would be expected to occur at random). A value of +1 for k indicates perfect agreement, and a value of 0 indicates that agreement occurs no more than would be expected by chance; a value less than 0 obviously indicates disagreement. p_o is the sum of the agreement proportions, so here

$$p_o = 0.5 + 0.1 + 0.05 = 0.65,$$

and p_e is the product of the total classifications by rater; thus

$$p_e = (0.65 \times 0.65) + (0.2 \times 0.25) + (0.15 \times 0.1) = 0.4875.$$

Hence $k = 0.3171$, which suggests that agreement does not vary much from what it would be by chance.

This can be formally tested. Fleiss (1981) gives an expression for the standard error of the kappa, $s.e._o(k)$. This example gives $s.e._o(k) = 0.1962$, and thus a z statistic can be calculated as

$$z = \frac{k}{s.e._o(k)} = \frac{0.3171}{0.1962} = 1.6163$$

and compared against a standard normal distribution. This is not significant, given that the z value for $\alpha = 0.05$ is 1.96.

3. *Multiple Experts*

A further variation of the kappa can give some insight into multiple experts where each rates the solution of a set of cases, by other experts and the system, in a Turing test. Suppose we have five experts who rate 15 test cases. Using a three-way rating of 'good', 'acceptable' or 'poor', Table 3 gives an example.

Table 3. Classification of Expert Performance into Three Categories on 15 Cases by Five Experts

	Rating			
Expert	*Good*	*Acceptable*	*Poor*	$\sum_{j=1}^{k} x_{ij}^2$
1	10	4	1	117
2	8	4	3	89
3	12	3	2	157
4	7	7	1	89
5	10	5	0	125
Total	47	23	5	377

Again, using the notation of Fleiss (1981), there are $n = 5$ raters, $k = 3$ categories, and $m = 15$ ratings, and x_{ij} is the number of ratings in the jth category for the ith rater. The proportions in each rating category are

$$p_1 = 47 / 75 = 0.6267$$

$$p_2 = 23 / 75 = 0.3067$$

$$p_3 = 5/ 75 = 0.0667.$$

The weighted kappa k is now given by

$$k = 1 - \frac{nm^2 - \sum_{i=1}^{n} \sum_{j=1}^{k} x_{ij}^2}{nm(m-1) \sum_{j=1}^{k} p_j(1 - p_j)}$$

and in this example has a value of 0.2872. Again, an estimate of the standard error can be used to give a z value, and for this example z is not significant. This implies that the experts are not consistent: decisions would have to be made about who or what the system is actually trying to model (this may mean narrowing or broadening the available range of expertise).

In this example one could remove the cases solved by the expert system and produce another kappa value, and then compare the two. This

would show if the disagreement is due to the output from the system or in any case existed between the human experts. Producing a kappa for multiple experts prior to development of the system is a useful measure. If agreement between human experts is good, the system can be compared against the joint agreement of all other experts using a statistic developed by Williams (1976).

VI. MANAGING VALIDATION

Whatever methods are used for validation, success may depend as much upon how these methods are managed as on the methods themselves. Three issues are particularly important: the location of validation in the development life cycle, the amount of money spent on validation, and the formality of the process.

A. Location in the Life Cycle

The position of validation in the life cycle has been discussed in detail in Gaschnig et al. (1983). Recently, Benbasat and Dhaliwal (1989) have expanded the impact of location of life cycle as a substantial portion of the basis of the choice of validation methods. The impact of the life cycle is basically analogous to the when issue in validation as discussed in O'Keefe et al. (1987): When do you do validation?

There are a number of reasons that the location of validation in the life cycle has an impact on determining the type and extent of validation. First, it is critical to verify the programmed knowledge base before substantial validation efforts are begun. If validators find errors in the knowledge base due to inappropriate implementation of particular technologies, substantial doubt may be cast on the ability of the developers to produce a correct system (let alone a system that can perform the work of an expert).

Second, some approaches to the validation of expert systems compare portions of the knowledge base at different stages of the life cycle (for example, O'Leary and Kandelin 1988). The expected relationships between information is compared to the actual relationships in order to gauge the extent of further validation efforts. Third, verification and validation efforts required in the initial stages of development are likely to be different from those required later in the process. Initial efforts may concentrate on

direct examination of components of the knowledge, while later efforts are likely to be aimed at evaluation of the system as a whole.

Validation will be pursued at numerous stages of the life cycle. Recently, there has been a move to include expert knowledge validation as part of the knowledge acquisition process (Shaw and Woodward 1988). As with other software, catching problems early will save considerable effort downstream. For systems that have a very low user's risk, it may be possible to field test the system: put it in place, and let the users find errors. This is possible, however, only if a user can determine when an error has occurred (for example, a piece of equipment still does not work after being mended under advice). For many systems, this will not be possible. When user's risk is high, a formal validation of the system prior to implementation is virtually obligatory (Hickam et al., 1985).

1. Knowledge Acquisition

One of the initial processes in the development of an expert system is knowledge acquisition. There has been increasing emphasis on validation during knowledge acquisition. This is particularly the case with automated knowledge acquisition processes, for example, tools such as ACQUINAS (Boose and Bradshaw 1987). Such approaches are suggested as validation techniques when supplemented with another knowledge acquisition technique, such as protocol analysis (Benbasat and Dhaliwal 1989).

2. Prototyping

Typically, a prototyping methodology is used in the development of an expert system. Some of the validation effort can be done at the prototype stage; for example, selected test data can provide insight into the initial validity of the system. As noted in O'Leary (1988c), prototypes can be an important validation tool, since "prototypes provide an opportunity to test assumptions about the knowledge base, inference strategies of the expert and other characteristics of the system." As noted earlier, prototypes also can provide for a test of underlying first principles.

3. Knowledge-Base Maintenance

If an expert systems evolves over time there is a need to build the validation process into the maintenance of the system. This often means that it must be the organizational responsibility of someone to validate new knowledge, alterations to existing knowledge, enhancements to the system, and so on. In the O'Leary and Watkins (1989) survey, at least one company employed a knowledge-base manager. That manager was responsible for a particular expert system and for ensuring that any new knowledge added to the system was verified and validated. Typically, a knowledge-base manager will be someone with less technical understanding of the software but possibly more domain expertise.

Expert system maintenance systems have become increasingly important as systems grow in size and complexity. Such systems have varying capabilities, but verification is normally one of the primary concerns, in order to ensure that the knowledge added to the system is consistent. Shatz, Strahs, and Campbell (1987) present one such system.

If case studies have been previously used to validate a system, they may be used to revalidate a system. Often, unfortunately, this is not possible, since changes to knowledge, information, or organization operating procedures will make them outdated.

B. Costs and Benefits of Validation

Implicitly and explicitly, cost assessments permeate virtually all system development and validation efforts since there are always resource or time constraints. Hence the attention given to any part of the life cyle is to some extent dependent upon the perceived benefit of validation in that part of the process.

The survey summarized in O'Leary and Watkins (1989) found that validation efforts rarely exceed budget and generally are allocated significantly less of the total system budget than is normally planned. Validation efforts are often driven out of the life cycle by the production process of developing the expert system. Further development and enhancements are often perceived as more important than validation. Additionally, the cost and benefits of different validation methodologies are likely to depend on their location in the development

process. For a system at the prototype level there may not be a detailed user interface. As a result it may prove quite costly to have the expert, end user, or sponsor involved directly in any validation efforts.

C. Formality

Virtually all the factors discussed in this paper impact the cost of validation of an expert system. However, generally the formality of the validation effort will be a major factor in the determination of the cost. The formality has an impact on who performs the validation, when the validation is performed, and of what the validation consists.

In informal validation, the process is left to the developers and programmers, and possibly subsumed into other parts of the life cycle. In formal validation, the cost of the validation will include the time of experts (perhaps third party) spent on validation and the time of the sponsor of the project, and the cost of acquiring or generating case studies or a simulation. A formal validation of the system is likely to take place at the conclusion of one of the major prototypes, and as a result, substantial up-front effort is likely to be made to ensure that the system meets the demands placed on it. Finally, in a formal validation, the validation process is likely to require a substantial amount of interaction with the system and include a specified sequence of tests, such as a Turing test.

VII. GUIDELINES FOR MANAGEMENT

This paper has presented a host of methods for validating expert systems and emphasized the importance of structuring the validation effort. Unfortunately, as concluded by O'Keefe et al. (1987), what we know about validation of expert systems does not easily translate into guidelines. Differences in application domains, the development life cycle, and the technology used make generalization difficult. Further, there are few published accounts of validation efforts, handicapping the generation of common experience.

Despite this, our research and experience leads us to believe that there are three crucial components of the validation process, each of which should be followed in any validation effort.

Plan

Plan in detail the validation effort: where it will occur in the development process, and how much money and time will be spent. If validation is to be effective, it likely should be formal. If only informal validation is to be performed, understand the associated problems. Focus carefully on how the system is to be judged and establish clear criteria for validation.

Perform

Choose and use appropriate methods. Validate the components and then the system as a whole. Test cases, Turing tests, and simulation are three case-oriented methods; other methods discussed above may also be appropriate. A field test or control group experiment may also be viable. If possible, use statistical tests rather than subjective evaluation.

Act

The results of validation are not good unless they are acted upon. This may require canceling the further development of a system, or investing more money into its development, but often validation effort will result in better systems that can be implemented more quickly.

Validation is a key part of the methodology of expert systems. Improving validation will greatly improve the quality of the resulting systems.

REFERENCES

Adrion, W., M. Branstad, and J. Cherniavsky. "Validation, Verification and Testing of Computer Software." *ACM Computing Surveys* 14, No. 2 (1982): 159–192.

Bachant, J. and J. McDermott. "R1 Revisited: Four Years in the Trenches." *AI Magazine* 5, No. 3 (1983): 21–32.

Balci, O. "Credibility Assessment." In O. Balci, ed., *Proceedings of the 1987 Eastern Simulation Conference.* La Jolla, CA: Society for Computer Simulation, 1987.

Balci, O. and R. Sargent. "A Methodology for Cost Risk Analysis in the Statistical Validation of n Simulation Models." *Communications of the ACM* 24, No. 4 (1981): 190–197.

———. "Validation of Simulation Models Via Simultaneous Confidence Intervals." *American Journal of Mathematics and Management Sciences* 4, Nos. 3, 4 (1984): 375–406.

Basden, A. "On the Application of Expert Systems." *International Journal of Man-Machine Systems* 19 (1983): 461–477.

Benbasat, I. and J. Dhaliwal. "A Framework for the Validation of Knowledge Acquisition." *Knowledge Acquisition* 1 (1989): 215–233.

Boose, J. and J. Bradshaw. "Expertise Transfer and Complex Problems Using Aquinas as a Knowledge Acquisition Workbench for Expert Systems." *Internation Journal of Man-Machine Systems* 26 (1987): 3–28.

Buchanan, B. and E. Shortliffe. *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Project.* Reading, MA: Addison-Wesley, 1985.

Buchanan, B., G. Sutherland, and E. A. Feigenbaum. "Heuristic DENDRAL: A Program for Generating Explanatory Hypotheses in Organic Chemistry." In D. Michie, ed., *Machine Intelligence 4*. New York: Elsevier, 1969.

Chandrasekaran, B. "On Evaluating AI Systems for Medical Diagnosis." *AI Magazine* 4, No. 2 (1983): 34–37.

Cohen, J. "A Coefficient of Agreement for Nominal Scales." *Educational and Psychological Measurement* 20 (1960): 37–46.

———. "Weighted Kappa: Nominal Scale Agreement with Provision for Scaled Disagreement or Partial Credit." *Psychological Bulletin* 70, No. 4 (1968): 213–220.

Davis, R. "Reasoning from First Principles in Electronic Troubleshooting." *International Journal of Man-Machine Studies* 24 (1984): 347–410.

Duchessi, P., H. Shawky, and J. P. Seagle. "A Knowledge-Engineered System for Commercial Loan Decisions." *Financial Management* (Autumn 1988): 57–65.

Duda, R., J. Gaschnig, and P. Hart. "Model Design in the Prospector Consultant System for Mineral Exploration." In D. Michie, ed., *Expert Systems in the Microelectronic Age*. Edinburgh University Press, 1979, pp. 153–167.

Eglese, R. W. "Heuristics in Operational Research." In V. Belton and R. M. O'Keefe, eds., *Recent Developments in Operational Research.* Oxford, UK: Pergamon Press, pp. 49–68.

Ernst, C. J., ed. *Management Expert Systems*. Reading, MA: Addison-Wesley, 1988.

Fleiss, J. L. *Statistical Methods for Rates and Proportions.* New York: John Wiley, 1981.

Fox, M. S. and S. F. Smith. "ISIS: A Knowledge-based System for Factory Scheduling." *Expert Systems* 1, No. 1 (1984): 25–49.

Gaschnig, J., P. Klahr, H. Pople, E. Shortliffe, and A. Terry. "Evaluation of Expert Systems: Issues and Case Studies." In F. Hayes-Roth, D. A. Waterman, and D. B. Lenat, eds., *Building Expert Systems.* Reading, MA: Addison-Wesley, 1983.

Gruhl, J. "Model Credibility and Independent Evaluation: Three Case Studies." *Omega* 10, No. 5 (1982): 525–537.

Hamilton, S. and N. L. Chervany. "Evaluating Information System Effectiveness—Part I: Comparing Evaluation Approaches." *MIS Quarterly* 5, No. 3 (1981): 55–69.

Hansen, J. and W. Messier. "A Preliminary Investigation of EDP-XPERT." *Auditing: A Journal of Theory and Practice* 6, No. 1 (1986): 109–123.

Hickam, D. H., E. H. Shortliffe, M. B. Bischoff, A. C. Scott, and C. D. Jacobs. "The Treatment Advice of a Computer-based Cancer Chemotherapy Protocol Advisor." *Annals of Internal Medicine* 103 (1985): 929–936.

Jackson, P. *Introduction to Expert Systems*. Reading, MA: Addison-Wesley, 1986.

Keen, P. W. "Value Analysis: Justifying Decision Support Systems." *MIS Quarterly* 5, No. 1 (1981): 1–15.

Kerlinger, F. *Foundations of Behavioral Research.* New York: Holt, Reinhart & Winston, 1973.

Kulikowski, C. A. and S. H. Weiss. "Representation of Expert Knowledge for Consultation: The Casnet and Expert Projects." In P. Szolovitz, ed., *Artificial Intelligence in Medicine.* Boulder, CO: Westview Press, 1982, 21–56.

Landry, M., Malouin, J.-L., and M. Oral. "Model Validation in Operations Research." *European Journal of Operational Research* 14 (1983): 207–220.

Langlotz, C. P. and E. H. Shortliffe. "Adapting a Consultation System to Critique User Plans." *International Journal of Man-Machine Studies* 19 (1983): 479–496.

Langlotz, C. P., E. H. Shortliffe, and L. M. Fagan. "Using Decision Theory to Justify Heuristics." In *Proceedings of AAAI 86.* Menlo Park, CA: AAAI, (1986), pp. 215–219.

Lethan, H. and H. Jacobsen. "ESKORT—An Expert System for Auditing VAT Accounts." *Proceedings of Expert Systems and Their Applications.* Avignon, France, 1987.

Liebowitz, J. "Useful Approach for Evaluating Expert System." *Expert Systems* 2, No. 3 (1986): 86–96.

Meservy, R., A. Bailey, and P. Johnson. "Internal Control Evaluation: A Computational Model of the Review Process." *Auditing: A Journal of Theory and Practice* 6, No. 1 (1986): 44–74.

Messier, W. F. and J. V. Hansen. "A Case Study and Evaluation of EDP-XPERT." *International Journal of Intelligent Systems in Accounting, Finance and Management* 1, No. 3 (1992): 173–186.

Moninger, W. R., T. R. Stewart, and P. Mcintosh. "Validation of Knowledge-Based Systems for Probabilistic Reasoning." In *Proceedings of the 1988 AAAI Workshop on Validation and Testing of Knowledge-Based Systems.* Menlo Park, CA: AAAI.

Mosteller, F. and R. E. K. Rourke. *Sturdy Statistics.* Reading, MA: Addison-Wesley, 1973.

O'Keefe, R. M. "The Evaluation of Decision-Aiding Systems: Guidelines and Methods." *Information & Management* 17 (1989): 217–226.

O'Keefe, R., O. Balci, and E. Smith. "Validating Expert System Performance." *IEEE Expert* 2, No. 4 (1987): 81–89.

O'Leary, D. "Validation of Expert Systems." *Decision Sciences* 18, No. 3 (1987): 468–486.

———. "Methods of Validating Expert Systems." *Interfaces* 18, No. 6 (1988a): 72–79.

———. "On the Representation and the Impact of Reliability on Expert System Weights." *International Journal of Man-Machine Studies* 29, No. 6 (1988b): 637–646.

———. "Expert System Prototyping as a Research Tool." In E. Turbin and P. Watkins, ed., *Applied Expert Systems.* Amsterdam: North-Holland, 1988c, pp. 17–32.

O'Leary, D. and N. Kandelin. "Validating the Weights in Rule-based Expert Systems." *International Journal of Expert Systems* 1, No. 3 (1988): 253–279.

O'Leary, D. and P. Watkins. "Expert Systems in Internal Auditing." Unpublished paper, School of Accounting, University of Southern California, 1989.

Quinlan, J. R. "Discovering Rules by Induction from Large Collections of Samples." In D. Michi, ed., *Expert Systems in the Microelectronic Age.* Edinburgh University Press, UK, 1979, pp. 168–201.

Rushby, J. *Quality Measures and Assurance for AI Software.* NASA Contract Report 4187, Washington DC, 1988.

Shatz, H., R. Strahs, and L. Campbell. "ExperTAX: The Issue of Long-Term Maintenance." *Proceedings of the 3rd International Conference on Expert Systems* (1987): 291–300.

Shaw, M. and J. Woodward. "Validation in a Knowledge Support System: Construing and Consistency with Multiple Experts." *International Journal of Man-Machine Studies* 29, No. 3 (1988): 329–350.

Shpilberg, D. and L. E. Graham. "Developing ExperTAX: An Expert System for Corporate Tax Accrual and Planning." In M. A. Vasarheyli, ed., *Artificial Intelligence in Accounting and Auditing.* New York: Markus Weiner, 1989, pp. 343–372.

Waterman, D. A. *A Guide to Expert Systems.* Reading, MA: Addison-Wesley, 1986.

Weiss, S. M. and C. A. Kulikowski. *A Practical Guide to Designing Expert Systems.* New Jersey: Rowman and Allanheld, 1984.

Williams, G. "Comparing the Joint Agreement of Several Raters with Another Rater." *Biometrics* 32, No. 2 (1976): 619–627.

Yu, V., B. Buchanan, E. Shortliffe, S. Wraith, R. Davis, A. Scott, and S. Cohen. "Evaluating the Performance of a Computer-based Consultant." *Computer Programs in Biomedicine* 9, No. 1 (1979): 95–102.

Yu, V., L. Fagan, S. Wraith, W. Clancy, A. Scott, J. Hanigan, R. Blum, B. Buchanan, and S. Cohen. "Antimicrobial Selection by Computer." *Journal of the American Medical Association* 242, No. 12 (1979b): 1279–1282. (See also Buchanan and Shortliffe [1985, chap. 31].)