To appear: International Journal of Human Computer Studies, Special Issue on Evaluation of Knowledge Engineering Techniques, 2000, WP:b/97/quench/words/words(July 3, 1999).

Critical Success Metrics: Evaluation at the Business-Level

Tim Menzies[‡]**,**
[‡]NASA/WVU IV&V Facility, 100 University Drive, Fairmont WV 26554 <tim@menzies.com>

Abstract

If we lack an objective human expert oracle which can assess a system, and if we lack a library of known or desired behaviour, how can we assess an expert system? One method for doing so is a *critical success metric* (CSM). A CSM is an assessment of a running program which reflects the business concerns that prompted the creation of that program. Given *pre-disaster* knowledge, a CSM can be used while the expert system is in routine use, without compromising the operation of the system. A general CSM experiment is defined using pre-disaster points which can compare (e.g.) human to expert system performance. Examples of using CSMs are given from the domains of farm management and process control.

Introduction

Experts can often disagree about what constitutes a competent system ([Shaw 1988, Gaschnig, Klahr, Pople, Shortliffe & Terry 1983]). Even if these authors are expert in their fields, they may still be unable to perform objective expert evaluations. The *halo effect* prevents a developer for looking at a program and assessing its value. Cohen likens the halo effect to a parent gushing over the achievements of their children and comments that...

> *What we need is not opinions or impressions, but relatively objective measures of performance.* [Cohen 1995], p74.

The method for objective assessment explored in this article is *critical success metrics* (CSMs); i.e. some number inferred from the system which, if it passes some value, demonstrates conclusively that the system is a success. If such a critical measurement is observed, then the system will be deemed to be a success, regardless of other less critical measures (e.g. slow runtimes). While this paper will focus on CSMs for knowledge engineering, there is nothing stopping software engineers from using the same principles in their work (e.g. [Reel 1999]).

For example, consider the PIGE farm management expert system [Menzies, Black, Fleming & Dean 1992]. PIGE advised on diets and genotypes for pigs growing in a piggery. Given a particular configuration of the livestock, an optimization model could infer the annual profit of the farm. Alternate configurations could be explored using a simulation model. A user can choose some settings, then run the simulation model to see if the system's performance improved. The CSM for PIGE was *can the system improve farm profitability as well as a pig nutrition expert?* If this could be demonstrated, then the tool could be sold as a kind of automatic pig growth specialist. To collect this CSM, at the end of a three month prototyping stage, we compared the performance of the pig nutritionist who wrote the PIGE rules against PIGE. We observed that, measured in purely economic terms, this expert system out-performed its human author (!!). The CSM study results for PIGE are shown in Figure 1.

This single CSM study changed the direction of the project. The graph of the CSM study became a succinct argument for collecting further funding. It was also very useful in sales work. PIGE became Australia's first exported expert system and was used on a routine daily basis in America, Holland, Belgium, France, Spain and Australia. In part, the success of the system was due to its ability to demonstrate its utility via a CSM.

Nevertheless, the CSM study of PIGE is a poor evaluation study. A good experiment is run multiple times with some variation between each trial [Cohen 1995]. CSMs should be viewed as the inner measurement process within a well-defined experiment. A general class of such experiments are described below, along with an example in a process control domain. This example will use a technique called a *pre-disaster point* (defined below). Our example will be preceded by general notes on CSMs and their advantages.

About CSMs

This section offers some basic notes on CSMs: their definition, some pragmatic issues, their advantages and disadvantages.

Definition

CSMs are a reflection of:

- the contribution of the behaviour of the software
- in a particular business context.

Hence:

- They are very domain-specific. However this does not mean they are can't be analyzed. This article concerns itself with the general themes of CSMs.

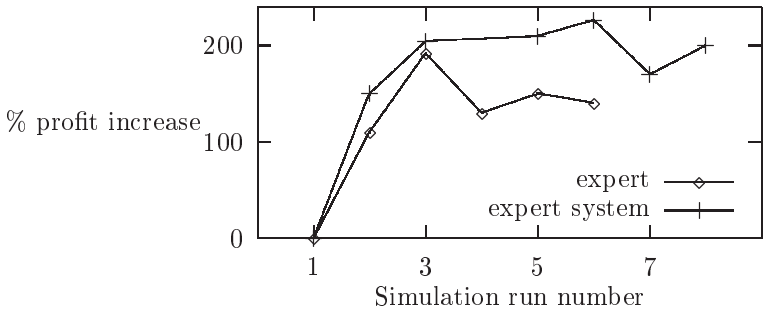

Figure 1: Critical success metrics for PIGE. From [Menzies et al. 1992].

- They typically do not refer to internal properties of a program. In the language of software metrics, a CSM is some survey on *external attributes* reflecting the context of a system [Fenton & Pfleeger 1997, p78]. In this regard, CSMs are very different to the syntactic anomaly detection systems (*internal attributes*) of the KBS verification community [Preece 1992].
- They cannot be developed by programmers without extensive input from business users. Programmers developing CSMs without business user involvement typically focus on internal attributes; e.g. lines of code per function, bugs fixed per day, etc. Such internal attributes may not connect to the business case which motivated the program's development.
- They can only be collected once the program is running in its target context.

Pragmatic Issues

Even if can't collect CSMs until an expert system is deployed, we should still define them at a very early stage. Evaluation should be considered as early as possible when building a system [Gaschnig et al. 1983]. The incremental application of a pre-defined success criteria can be a powerful tool for managing evolving systems [Booch 1996]. Often, the evaluation criteria imposes extra requirements on the implementation. We may need to build a very simple initial system that collects baseline measurements which reflect current practice. For example, once I identified *increases sales per day* as the CSM for a dealing room expert system. However, this number was not currently being collected in the current software. Sales per day could be estimated from the quarterly statements, but no finer grain data collection was performed at that site. Hence, prior to building the expert system, a database system had to be built to collect the baseline data.

While CSMs are obvious in retrospect, they can take weeks of analysis to uncover. For example:

- It took two weeks full time analysis on the domain before the above dealing room CSM was uncovered.
- In the process control system discussed below, the CSMs were only isolated once a prototype expert system system was developed.
- In the PIGE system, nutrition experts argued for weeks about the merits of different protein utilization models. Then the marketing people commented that such considerations were irrelevant if it could not be demonstrated that the systems recommendations improved the overall profitability of a farm. Hence, the evaluation focus moved from the protein utilization models to issues of modeling the farm economics. The results, shown above, were an impressive demonstration of the marketability of the system.

The observation that CSMs can take some time to isolate would not surprise software engineering metrics researchers. Basili [1992], characterizes software evaluation as a *goal-question-metric* triad. Beginners to experimentation report whatever numbers they can collect without considering the goal of the research project, what questions relate to that goal, and what measurements could be made to address those questions. Before goal-question-metric there must be an analysis involving the stake holders of the project to establish the appropriate goals. Offen & Jeffery [1997] offer the appropriate caution that this important task can take a non-trivial amount of time.

Advantages

CSMs have business-level advantages as well as technical advantages as an assessment tool. For many business situations, CSMs are useful:

- A sales situation where it must be demonstrated to a client that some software system is better than (e.g.) manual methods; e.g. the PIGE example given above.
- Any audit situation where is must be demonstrated that the cost of a new application was worthwhile; e.g. the dealing room system described above.
- Engineering applications where controlling software must be evaluated; e.g. the process control example given below.
- A negotiation situation where a cautious user group must be convinced of the merits of some software system
- Any software situation where software must be continually monitored; e.g. air-traffic control; legal systems; evidence-based medical applications.

As an evaluation tool, CSMs have advantages over other evaluation tools for expert systems. Expert systems are usually evaluated via panels of experts or some database of known or desired behaviour. Such evaluations can report the accuracy of those system to an enviable degree of accuracy. For example:

- Hayes [1997] can demonstrate that her expert system developed in two years performs as well as someone with five years experience in that field.
- Preston, Edwards & Compton [1993] report a biochemical interpretation system that is 95 percent accurate on the cases it analyzes.
- Yu, Fagan, Wraith, Clancey, Scott, Hanigan, Blum, Buchanan & Cohen [1979] reports that MYCIN, an expert system for prescribing antibiotics, clearly out-performs senior medical personnel.
- Menzies & Compton [1997] have defined a precise validation system which can show exactly what percentage of known behaviour cannot be explained by some model.

Using CSMs, we are placing an external business-level success criteria on a running system. Hence, we can evaluate a system even when:

- No objective source of expertise is available; i.e. expert panel members are unavailable.
- There exists no representative library of the known/desired behaviour of the system; i.e. we have yet to have enough experience with the domain to record all the possible things which can happen.

Also, the evaluation will be a *business-level* evaluation. Business users may demand objective evidence as to the business value of some program before allowing it to control some critical business process. This evaluation may not comprise developer-level concerns such as runtimes or (in the case of PIGE) current fashions in theories of protein utilization. In the PIGE and dealing room examples, the CSMs had to reflect the fundamental business case which motivated the project: increased profitability.

Further, given a *pre-disaster point*, we can do this while the system is in routine operation. A pre-disaster point refers to a state of the system that is less-than-optimum, but not yet critically under-performing. As we shall see below, CSMs plus pre-disaster knowledge allows us to assess a running system without compromising its operation.

Disadvantages

CSMs are business-level tools for assessing a system and can provide a yes-no assessment of a system. This is their main strength, and their failing. Evaluation can have many aims, not all of which are served by CSMs. For example, as argued below, CSMs are not a tool for *model-based evaluation* or for making *externally valid conclusions*.

Model-Based Evaluations Model-based evaluation was described by Newell [1972] in his famous argument "Why you can't play 20 questions with Nature, and win". Reacting to an excess of experimental zeal in cognitive psychology, Newell argued forcibly against conducting experimental programmes that offered single answers to yes-no questions. Rather, said Newell, we should:

- Perform multiple studies of complex systems.
- Collect large amounts of data.
- Unify those results into some rich theoretical structure describing some process of interest.

Newell argued that large simulations to incremental explore parts of some rich theoretical structure are far more insightful than (e.g.) twenty yes-no questions.

CSMs encourage yes-no answers and so run foul of Newell's objection. To see this, suppose the CSM for PIGE had yielded a negative result. In that circumstance, note that no information is available from the CSM to help us repair PIGE. CSMs are a behaviour summary tool: all manner of complex internal interactions are condensed down to a single number.

Now contrast this CSM-evaluation with a data-rich Newell-style evaluation. Once a problem has been detected, then an experimenter has access to a large data base of information. This database can be used to find and fix the error.

The data-poor nature of CSM-evaluations is not a fatal flaw with the CSM paradigm. Clearly, if the aim of the evaluation is not "assess this system" but "assess and repair", then other data must be collected to support that repair. However, within that larger scaler data collection, we would strongly urge that CSMs be included. To see this, consider the developer who has graphed (e.g.) lines of code per method for the 4100 methods in her system. What does such data collection mean if not correlated against some business concern (e.g. time to implement the next business change request)?

Also, I pursued the idea of CSMs since, in my experience, precise data collection was always expensive. In controlled laboratory conditions (Newell's home ground), it is very easy to collect highly reliable numbers. However, out in the field, the situation is very different. Even when rich models are available to describe some phenomena, the data is not available to populate that model (for theoretical techniques for performing model-based evaluations in data-poor domains, see [Menzies & Compton 1997]). In many domains, it is difficult and expensive to find or build sets of observations. For example:

- The (in)famous *Limits to Growth* study attempted to predict the international effects of continued economic growth [Meadows, Meadows, Randers & Behrens 1972]. Less than 0.1 percent of the data required for the theories was available [Coles 1974].
- Data collections in neuroendocrinology (the study of nerves and glands) can be just as sparse since data collection in that domain is very expensive. In one extreme example, 300,000 sheeps brains had to be filtered to extract 1.0 milligrams of purified thyroptin-releasing hormone [Krieger 1980].

External Validity Another issue is the *external validity* of a CSM experiment. External validity is concerned with what

can be generalized from one experiment to other situations. Current research in knowledge engineering tries to evaluate classes of systems in order to make predictions about best techniques for future problems. CSMs are tightly focused on a single system. Hence, it would be hard to use them for some externally valid conclusion. While CSMs can give yes-no answers about one system, they cannot offer details conclusions about classes of systems. Suppose a range of systems were being assessed, and the systems differed on some internal feature of the tools used: (e.g.) multiple vs single parent inheritance. The summation within the CSM measure may blur or ignore this tool distinction.

On the other hand, CSMs are not the only evaluation tool with external validity issues. The Sisyphus project is an international consortium of knowledge engineering researchers who build different systems from the same specification using different tools [Linster 1992, Schreiber & Birmingham 1996, Shadbolt, O'Hara & Crow 2000]. Sisyphus was intended to define an evaluation study for certain techniques in knowledge engineering. While the Sisyphus project has unified a diverse range of researchers, the results to date are inconclusive. Shadbolt et al. [2000] lists all the problems associated with the labors of Sisyphus and concludes:

> ... none of the Sisyphus experiments have yielded much evaluation information (though at the time of writing Sisyphus-III) is not yet complete).

Nevertheless, the Sisyphus researchers remain optimistic and the project continues.

In summary, practioners should collect all the data they can about the behavior of their systems. Further, they should always consider what general statement can be made from their observations. However, they should also acknowledge that data collection and generalization is a resource bound activity. In resource-bound environments, the very least that should be attempted are CSM data collections and an assessment of the system at hand.

CSM Evaluation Using Pre-Disaster Points

This section offers a general design for an evaluation experiment using CSMs and a *pre-disaster* point. The aim of this evaluation is to check if the program is dumber than than some human, with respect to some chosen CSMs. In the experiment, the human or expert system is trying to control some aspects of the environment; e.g. make a diagnosis, prescribe medicines which reduce fever, improve profitability, etc.

Trials would alternate between the human and computer experts. A trial would begin when the system is in some steady state; i.e. there appears to be no currently active problems. During the course of each trial, the expert under trial would have sole authority to order adjustments to the environment. The trial would terminate whenever the pre-disaster point was reached. Authority to adjust the environment would then pass to the human experts. At the conclusion of each trial, a CSM is applied to assess the environment during the trial period.

At the end of a statistically significant number of trials (say, 20 for each population of experts), the mean performance of the two populations of experts would be compared using a t-test as follows. Let m and n be the number of trials of expert system and the human experts respectively. Each trial generates a performance score: $X_1 \ldots X_m$ with mean μ_x for the humans; and performance scores $Y_1 \ldots Y_n$ with mean μ_y for the expert system. We need to find a Z value as follows:

$$S_x^2 = \frac{\Sigma(x_i - \mu_x)^2}{m-1}$$

$$S_y^2 = \frac{\Sigma(y_i - \mu_y)^2}{n-1}$$

$$Z = \frac{\mu_x - \mu_y}{\sqrt{\frac{S_x^2}{m} + \frac{S_y^2}{n}}}$$

Let a be the degrees of freedom. If $n = m = 20$, the $a = n + m - 2 = 38$. We reject the hypothesis that expert system is worse than the human (i.e. $\mu_x < \mu_y$) with 95% confidence if Z is less than $(-t_{38,0.95} = -1.645)$.

Note that this human/expert system comparison could also be used to assess different expert systems.

An Example: Process Control CSMs

This section offers a detailed example of the above experiment. In the summer of 1986/87, I implemented QUENCH, an expert system computer program for the control of the quench oil tower at ICI Australia's Olefines petrochemical plant in Sydney [Menzies & Markey 1987]. Once the system was built, I offered to management the experimental design discussed below. The evaluation experiment was approved but, due to a change in management, never performed. Nevertheless, the experiment is relevant here since it illustrates many of the practical issues associated with CSM evaluations. For example:

- A simple rule-based system written in two weeks would take nearly a year to evaluate.
- The evaluation criteria chosen made no reference to the internal structure of the rule base.
- The CSMs described below were slow to develop. In fact, only once a working knowledge base was developed could we reverse engineer a success criteria for the system.
- CSMs cannot be generated via mere *program watching*. A detailed analysis is offered below of the drawbacks of letting experts subjectively evaluate an expert system (see below, the discussion on instrumentation and testing in *Assessing the Obvious Method*).
- The CSM collection implies only small changes to the running of the system. Further, using a pre-disaster point, the evaluation can occur without losing profit from the system. That is, evaluation may be practical even for systems as complicated as a large petrochemical plant.

Background to the QUENCH System

The Olefines petrochemical plant produces 240,000 tonnes of ethylene per year. It is a highly complex plant consisting in part of some 125 km of piping connecting numerous chemical processes. A unit of this plant is the quench oil tower. Inside the tower, hot cracked gases are cooled from around 400C to around 100C by mixing with oil. Certain gases are extracted at the top of the tower and the used quench oil, containing variable amounts of dissolved gases, is removed from the bottom. These dissolved gases effect the density of the removed oil. If the quench oil density moves outside of a certain narrow range, it can not be sold. In this case, ICI loses the revenue that would have come from its sale. Further, it must pay for the reprocessing or the disposal of the bad oil.

In order to keep the density on specification, the temperature at the bottom and the top of the tower must be maintained within one half of a degree of a target temperature. This is accomplished by altering the flow rates though the piping that surrounds the tower and/or by adjusting the heat exchange units attached to this piping. In practice, this is a non-trivial task. There have been cases when the operators of the tower have spent days attempting to return the density to an acceptable value. This process is directed by the the supervising engineers who communicate their instructions to the operators using heuristics similar to production rules. For example, to correct a very high quench oil density, an engineer could say to an operator:

```
if   the target(temps)=correct and
     the bottom(tower(temps))=high
then bring the bottom of the tower
     back on target by increasing
     the quench oil recycle flow
     rate by 20 tonnes per hour.
```

QUENCH contained 104 such rules.

Features of Large Petrochemical Plants

This section describes the features of large petrochemical plants that complicate the process of evaluation. Safety is a paramount consideration. Unsafe operating conditions could cost the lives of the workers in these plants.

Large petrochemical plants produce hundreds of millions of dollars worth of chemicals each year. The loss of a single day's revenue can cost a company hundreds of thousands of dollars. These economic imperatives are so pressing that the prolonged operation of these plants at less-than-optimum performance can not be tolerated.

There are major difficulties associated with deriving precise formalizations of these complex systems. For example, a mathematical model of the quench oil system would require the solutions of hundreds of simultaneous equations. Certain parameters required in these equations require uncertain physical properties data; i.e. these parameters are not known. Consequently it is possible that after months of development work, a mathematical model of the quench oil system may be grossly inaccurate. Without precise formalizations, the only way to accurately predict the effects of certain changes to the plant is to make those changes and observe the effects.

The design of these large plants is typically customized to meet local requirements. Hence, the experience gained in (e.g.) controlling quench oil towers in other plants may not be relevant to this quench oil tower. In fact, the two supervising engineers who helped write QUENCH's rules are the only authorities on the control of the Olefines' quench oil tower. In the jargon of the psychologist or the statistician, there is no control group available for experiments on the tower. Further, there is no objective expertise that can be called upon to accurately assess the suggestions made by quench oil tower experts (be they computers or human beings).

The Obvious Evaluation Method

One method for assessing the expertise of the program is to run it in parallel with the existing system. The supervising engineers could compare QUENCH's suggestions with their own advice for problem situations. This method will be referred to as the *obvious method* and (the pre-disaster CSM evaluation described below will be called *the preferred method*). The obvious method has several advantages:

- It does not upset the normal operations of the plant.
- The plant remains under the control of the experts with the most experience on controlling the plant; i.e. the supervising engineers.
- It requires no control group.
- The computer and the human experts are being tested under identical plant conditions.

Experimental Design Theory

Regrettably, there are glaring design faults in the obvious method. These faults are described below after an introduction to experimental design theory.

Campbell and Stanley [Campbell & Stanley 1970] assess experimental designs in terms of their *internal* and *external validity*.

> *Internal validity* is the basic minimum without which any experiment is uninterpretable: Did in fact the experimental methods make a difference in this specific experimental instance? *External validity* asks the question of *generalizability*: To what populations, settings, treatment variables, and measurement variables can this effect be generalized? [Campbell & Stanley 1970] (p4).

As to external validity, the claim of this paper is that the preferred method can be generalized to other expert system evaluations. For more comments on external validity, see the discussion above in *Disadvantages*.

As to internal validity, if we can not interpret the results of our experiment, then the experiment would have been pointless. Campbell and Stanley list several factors that could jeopardize internal validity. These factors have one feature in common: they could result in the effect of an experimental variable under study being confused with other factors. Each represents the effects of:

- *Selection:* i.e. if a test group was selected based upon their score on a certain measure, then this score could bias the behaviour of the test group in a certain way. Selection problems have been observed in the knowledge engineering literature. A bayesian system for medical diagnosis in a Leeds hospital apparently out-performed senior clinicians. However, a subsequent evaluation by another team in Copenhagen identified that the first study artificially restricted the number of possible diagnosis. When this restriction was removed, the performance of the bayesian system fell to 65 percent of that of human doctors [Gaschnig et al. 1983] (p250–251). Also, Gashing et.al. report that a preliminary evaluation of the XCON system [McDermott 1993] failed to detect flaws in XCON since it only studied a tiny fraction of the set of possible XCON inputs [Gaschnig et al. 1983] (p270–271).
- *Maturation*: i.e. changes over time in the test subjects. For example, the subject in an experiment may grow bored, tired, hungry, etc. and their reactions to various experimental variables may alter for reasons that are not under study. Maturation is a common problem with knowledge engineering research. Researchers often report improvement in some process when they use their own tools for a period of time (e.g [Runkel 1995]). Such results hence conflate the effect of the tool with the effect of the developer learning how to best apply their own tool.
- *Instrumentation*: i.e. the calibration of the testing device changes. The springs of a weight scale may wear out, observers may change, or the reports of the same observer may alter as they gain experience with the experiment. Instrumentation is a common problem in many knowledge engineering studies. Knowledge engineering researchers rarely calibrate their measurements against some gold standard or *straw man*- an obviously inferior method [Menzies 1998*a*] (but some exceptions exist as noted below in the related work section). Without such a calibration, most knowledge engineering researchers can only say *software technology X lets me do task Y*. This is a less convincing statement than *software technology X lets me do task Y better than software technology Z*.
- *Statistical Regression*: i.e. the items/ people/ events in a test group are selected according to some extreme characteristic possessed by those items/ people/ events. Campbell and Stanley note that *the more deviant the score, the larger the error of measurement it probably contains* [Campbell & Stanley 1970], (p11). They observe that test results from such extreme test groups tend to revert to the mean behaviour. For example, in an experiment testing some skill, observations could show that the dull could become brighter and the bright duller.
- *History*: i.e. events occurring between observations other those under study. The variable of history is relevant to the feature of *experimental isolation*. If the effect under study can not be isolated from other effects, then it is hard to distinguish the results of known influences from unknown influences.
- *Testing*: i.e. the act of making an initial observation may somehow alter subsequent observations.
- *Mortality*: i.e. mortality refers to the changes to groups under comparison resulting from drop outs from the groups.

Assessing the Obvious Method

On several of the above points, the obvious method ranks quite well.

- *Maturation*: Maturation is not a problem since the test is not lengthy. The obvious method is an evaluation of expertise at a particular point in time.
- *Selection, and statistical regression*: The expert system is tested against whatever changes occur to the plant. Since there is no choice involved in selecting these test cases, these factors are not issues for this experimental design.
- *Mortality*: Mortality is only a issue to be considered for tests that take an appreciable period of time. Hence, it is not an issue with the obvious method.

However, the effects of history, instrumentation and testing are majors flaw in the obvious method.

- *History*: The suggestions of the supervising engineers are not always followed faithfully by the plant room operators. It is not uncommon for evening shift and night shift operators to ignore expert advice and apply their own control protocols. It is possible that these operators would tend to ignore a computer's advice even more than those of a human being. Having documented this problem, it will not be discussed further. The resolution of this problem is an administrative problem that will be crucial to the process of evaluation. For example, some monitor must be added to the experimental design such that we can check that if QUENCH recommends *X*, then the operators performed *X*. However, the nature of that monitor is a workplace relations issue that is beyond the scope of this researcher.
- *Instrumentation and Testing*: The program's expertise will be assessed by the supervising engineers. It is possible that their own perceptions of the program could alter with time. These engineers have been intimately connected with the program for several months. Human factors such as the halo effect (discussed above), egotistical considerations, disappointment or elation at their perceptions of the program's performance, etc., may distort their evaluation.

Hence, we reject the obvious method and move to the preferred method.

Defining CSMs for QUENCH

The preferred method requires CSMs and a pre-disaster point. This section offers CSMs. The next section offers a pre-disaster point.

There are three possibles CSMs for QUENCH:

1. A poll of all the electronic surveillance equipment that monitors the plant. This possibility is really a whole host of possibilities. There are many ways that the plant's

surveillance equipment could be summed together into a single performance figure. Such a summation would be a whole research topic in itself. Fortunately, there are easier methods.

2. The time to failure. (a method proposed by Kehoe, personal communication). The time between the starting the trial and reaching the pre-disaster point could be the performance figure. The longer this time, the better the performance.

3. Revenue from quench oil (a method proposed by Dr. Michael Brisk, ICI, personal communication). The sum of revenues gained from processing the quench oil could be the performance measure. If the density goes off specification, and money must be spent to reprocess or dispose of the bad oil, then this amount should be deducted from the sum. Like the time to failure, the greater this figure, the better the performance.

Methods two and three are not exclusive. The system could be studied using both criteria.

Defining the Pre-Disaster Point for QUENCH

We define the QUENCH pre-disaster point as follows: the point at which the supervising engineers realize that, despite their best efforts, the plant is defying their control strategies. If the plant reaches this pre-disaster point, then the control of the plant should be transferred to the best possible control system. In the case of testing QUENCH, the best possible control system is the supervising engineers. In the other case, when it is the engineers controlling the plant, the engineers would retain their authority to order alterations to the plant. They would then continue in their attempts to regain control over the plant processes.

Pre-disaster for QUENCH could be defined as a bad quench oil density that was not improving, for (say) two days in succession. The time delay of two days allows for the expert time to recognize a problem, give advice for that problem, and for the tower to react to the expert's advice. If the at end of this time the density was still bad and not improving, then the expert would be deemed to have lost control of the tower.

The terms *bad* and *not improving* could be defined using the ranges developed during the implementation of QUENCH. The expert system has the ability to assigns *symbolic tags* to numeric ranges. The ranges for the quench oil density (expressed in kilograms per cubic meter) are shown in Figure 2. The time rate of change in the density (expressed in change in density per 24 hours) has the symbolic tags shown in Figure 3.

Using these tags, we can define the pre-disaster point as a quench oil density that is either:

- moderately high or very high density and not falling quickly or falling slowly OR
- moderately low or very low density and not rising quickly or rising slowly.

Tag	Range
very high	> 1070
moderately high	> 1060
ok	> 1050
moderately low	> 1040
very low	$<= 1040$

Figure 2: Assessing quench oil density in QUENCH. From [Menzies & Markey 1987].

Tag	Range
rising quickly	> 7
rising slowly	> 2
steady	> -2
falling slowly	> -7
falling quickly	$<= -7$

Figure 3: Defining *changes* in QUENCH. From [Menzies & Markey 1987].

Maturation and the Preferred Method

While the preferred method addresses the problem of objectivity seen with the obvious method, it will be effected by maturation. Consider the following:

- During stable operating periods, an evaluation of QUENCH's expertise in bringing the quench oil density back on specification is meaningless. Any test of this expertise must wait for periods of operational instability.
- The response time of the quench oil tower to changes in low rates and heat-exchangers can be as much as several days. Hence, once unstable conditions are encountered, an evaluation of the effectiveness of QUENCH's suggestions may have to wait for as much as a week.
- If we assume twenty trials for each population, and that each trial takes at least a week, then the total experiment time will be at least 40 weeks.
- The current version of the QUENCH rule set was developed in two weeks. As a result of assessing the current version of the program, the system developers would gain months of experience with the system. This experience could be used to modify and improve the program. Therefore...
- The evaluation process could result in substantial modifications to QUENCH's rule set.

Another way of expressing the above could be to say that the experiment is testing the expertise of a system that is learning. The evaluation experiment is to be attempted for an expert system who is in the shallow end of a learning curve. As a result of the experience gained during the evaluation process, the rule set would be improved and the expert system will move rapidly up the learning curve. The problem is that this improvement would occur concurrently with the experiment.

Kehoe (personal communication) offers an interesting resolution to the maturation problem. He argues that another CSM could be added to the system. Let *F* be the number of

times the system is executed divided by the number of times the knowledge base is edited:

- If F tends to zero, the system is not being used.
- If F is less than one, then each run of the program is prompting a revision; i.e. there is something seriously wrong with QUENCH.
- If F is much greater than one, then the system is being run much more than it is being changed. Such an observation would suggest that some community finds using QUENCH to be of value.

Another response to this maturation problem would be to to forbid the modification of the rules during the evaluation period; i.e. stop the system moving along the learning curve. This is an undesirable solution. It is highly probable that the existing rule set could be vastly improved. It was developed in a fortnight and this is a surprisingly short time for an expert system. Human cognitive processes are notoriously hard to formalize. The experience of expert systems developers is that any current specification of an expert solution to a problem is incomplete [Menzies 1998*b*]. As experience with an expert system accumulates, inadequacies in the system's reasoning will always be detected. To correct these inadequacies, the system's knowledge based (e.g. QUENCH's rule set) must be modified. This cycle of flaw detection followed by knowledge base modification can continue indefinitely but concludes when the user is satisfied that the system can provide adequate performance in an adequate number of cases. Depending on the expert system application, this refinement process can continue for many years. Compton reports one case where the modification process seemed linear; i.e. it may never stop [Compton 1994].

This is not to say that the existing rule set lacks any utility for controlling the tower. The problem of assessing QUENCH only arose since the supervising engineers reported that they are satisfied with the output of the program. The short development time might have resulted from the choice of problem. QUENCH was ICI Australia's first direct experience with expert systems. The quench oil tower problem was selected as a comparatively simple first test case for the expert system methodology. One of the factors that made the problem simple was the Olefines' supervising engineers. These people spend significant amounts of their time explaining the workings of the Olefines plant to the control room operators. Hence, they have had considerable experience in expressing their knowledge in a concise manner.

Nevertheless, it is the author's belief that the program's rule set would benefit from further modification. It would be foolish to believe that QUENCH had somehow avoided the need for the long term knowledge base refinement process found to be necessary in other expert system application. Further, ICI would prefer the best possible control system for their tower. They may be less than enthusiastic about an experiment that inhibits the development of an optimum rule set. Hence, except for the Kehoe extension, I offer no revision to the preferred method to handle maturation.

Related Work

At the time of creating the QUENCH system, there was nothing in the petrochemical literature about empirical evaluation of expert systems. For example, in [Morari & McAvoy 1986] and [Ctc 1986] we can read hundreds of pages on American and Japanese expert systems and never read anything about evaluation. Perhaps the reason for this curious omission is the difficulties inherent in the task. As seen above, a whole host of factors threaten the internal validity of evaluating experiments in such plants.

More generally, business-level empirical KBS evaluation is rarely performed in the knowledge engineering field (but some exceptions were noted in the introduction). By business-level, I mean measures of a running expert system which relate to the business case which motivated the development of that expert system. A CSM is a business-level evaluation measure. Elsewhere, I have criticized this lack of evaluations in the knowledge engineering field [Menzies 1998*b*, Menzies 1998*a*]. This critique was motivated by the work of Feldman, Compton & Smythe [1989], followed by myself and Compton [Menzies & Compton 1997]. Feldman, Compton, and myself devised, refined, and optimized a general graph-based abductive framework for assessing a KBS. That framework used a library of known or desired behaviour [Menzies 1995]. An example of using this framework is given in [Menzies & Compton 1997]. One advantage of that framework over standard verification and validation is that the computational limits of the technique can be studied via *mutators* which auto-generate variants of known graphs [Menzies 1996, Waugh, Menzies & Goss 1997, Menzies, Cohen & Waugh 1998].

General principles for evaluation of knowledge engineering methods are discussed in [Cohen 1995, Shadbolt et al. 2000, Hori 2000]. General principles for comparative empirical evaluation of knowledge engineering methods are discussed in [Menzies 1998*a*]. Such comparative evaluations can take the form of:

- Analyzing program vs expert performance; e.g. [Hayes 1997, Menzies et al. 1992, Yu et al. 1979]. In general, only these program vs expert performance evaluations yield results relevant to the business case that motivated the construction of the expert system.
- Analyzing expert vs expert performance using different tools (e.g. [Corbridge, Major & Shadbolt 1995]) or records of their knowledge (e.g. [Shaw 1988]);
- Analyzing the performance of variants within some program either via an empirical average case analysis (e.g. [Waugh et al. 1997, Menzies et al. 1998]) or a theoretical analysis such as graph theory (e.g. [Menzies & Cohen 1997]) or a worst-case time complexity analysis (e.g. [Tambe & Rosenbloom 1994, Levesque & Brachman 1985]).

The verification and validation community offer test procedures for KBS:

- The verification community typically focuses on syntactic anomalies within a KBS (e.g. circularities, tautologies) [Preece 1992].

- The validation community focuses on the connection of the program to its environment. However, a typical validation paper focuses on (e.g.) automatic test case generation from an analysis of the dependency network within a program (e.g. [Ginsberg 1990, Zlatareva 1993]). The advantage of this technique is that it can be guaranteed that test cases can exercise all branches of a knowledge base. The disadvantage of this technique is that, for each proposed new input, an expert must still decide what constitutes a valid output. This decision requires knowledge external to the model, least we introduce a circularity in the test procedure (i.e. we test the a KBS using test cases derived from the structure of that KBS). Further, auto-test-generation focuses on incorrect features in the current model. I prefer to use some criteria from a totally external source since such external test cases can highlight
- Usually, publications from verification or validation community do not discuss how to assess a KBS with respect to the business case.

After evaluation, comes maintenance. KBS maintenance is a large field, discussed elsewhere [Menzies 1999].

Conclusion

CSMs let us evaluate a system without requiring a panel of experts of a database of known or desired behaviour. A behavioral success criteria is derived from the business case that motivated the construction of the expert system. The system is then executed and measurements are made which inform the success criteria. Coupled with a *pre-disaster point*, CSMs let us statistically evaluate a system in operation, without compromising that operation.

The general themes of CSMs presented here are as follows. CSMs are usually very domain-specific since they reflect the contribution of the behaviour of the software in a particular business context. Hence, they typically do not refer to internal properties of a program and they cannot be developed by programmers without extensive input from business users. CSMs are usually obvious, but only in retrospect: a CSMs can take weeks of analysis to uncover. CSMs are useful for assessing a particular system, but are a poor method for making general conclusions about a class of tools. CSMs may only be collectable from the working system. However, CSMs should be explored very early in the life cycle of an expert system since CSM collection may imply the extension of the system's design to collect the required data.

Acknowledgements

Discussions with Dr. Michael Brisk (ICI, Australia) and Dr. Jim Kehoe (Psychology, Uni. NSW), and comments from the anonymous referees, helped refine this material. This work was partially supported by NASA through cooperative agreement #NCC 2-979.

References

Basili, V. R. [1992], The experimental paradigm in software engineering, *in* H. D. Rombach, V. R. Basili & R. W. Selby, eds, 'Experimental Software Engineering Issues: Critical Assessment and Future Directions, International Workshop, Germany', pp. 3–12.

Booch, G. [1996], *Object Solutions: Managing the Object-Oriented Project*, Addison-Wesley.

Campbell, D. & Stanley, J. [1970], *Experimental and Quasi-Experimental Designs for Research*, Rand McNally & Company.

Cohen, P. [1995], *Empirical Methods for Artificial Intelligence*, MIT Press.

Coles, H. S. [1974], *Thinking About the Future: A Critique of the Limits to Growth*, Sussex University Press.

Compton, P. [1994], 'Personal communication'. regarding the status of the PIERS system.

Corbridge, C., Major, N. & Shadbolt, N. [1995], Models Exposed: An Empirical Study, *in* 'Proceedings of the 9th AAAI-Sponsored Banff Knowledge Acquisition for Knowledge Based Systems'.

Ctc [1986], 'Special issue on expert systems', Control Theory and Advanced Technology. Vol. 2, No. 3.

Feldman, B., Compton, P. & Smythe, G. [1989], Hypothesis Testing: an Appropriate Task for Knowledge-Based Systems, *in* '4th AAAI-Sponsored Knowledge Acquisition for Knowledge-based Systems Workshop Banff, Canada'.

Fenton, N. E. & Pfleeger, S. [1997], *Software Metrics: A Rigorous & Practical Approach*, International Thompson Press.

Gaschnig, J., Klahr, P., Pople, H., Shortliffe, E. & Terry, A. [1983], Evaluation of expert systems: Issues and case studies, *in* F. Hayes-Roth, D. Waterman & D. Lenat, eds, 'Building Expert Systems', Addison-Wesley, chapter 8, pp. 241–280.

Ginsberg, A. [1990], Theory reduction, theory revision, and retranslation, *in* 'AAAI '90', pp. 777–782.

Hayes, C. [1997], A study in solution quality human expert and knolwedge-based system reasoning, *in* P. Feltovich, K. Ford & R. Hoffman, eds, 'Expertise in Context', MIT PRess, chapter 14, pp. 339–362.

Hori, M. [2000], 'Stability of a domain-oriented component library: An explanatory case study', *International Journal of Human Computer Studies* . (to appear).

Krieger, D. [1980], The hypothalmus and neuroendocrinology, *in* D. Krieger & J. Hughes, eds, 'Neuroendocrinology', Sinauer Associates, Inc., pp. 3–122.

Levesque, H. & Brachman, R. [1985], A fundamental tradeoff in knowledge representation and reasoning (revised version), *in* R. Brachmann & H. Levesque, eds, 'Readings in Knowledge Representation', Morgan Kaufmann, Palo Alto, pp. 41–70.

Linster, M. [1992], A review of sisyphus 91 and 92: Models of problem-solving knowledge, *in* N. Aussenac, G. Boy, B. Gaines, M. Linser, J.-G. Ganascia & Y. Kordratoff, eds, 'Knowledge Acquisition for Knowledge-Based Systems', Springer-Verlag, pp. 159–182.

McDermott, J. [1993], 'R1 ("xcon") at age 12: lessons from an elementary school achiever', *Artificial Intelligence* **59**, 241–247.

Meadows, D., Meadows, D., Randers, J. & Behrens, W. [1972], *The Limits to Growth*, Potomac Associates.

Menzies, T. [1995], Principles for Generalised Testing of Knowledge Bases, PhD thesis, University of New South Wales. Avaliable from `http://www.cse.unsw.edu.au/~timm/pub/docs/95thesis.ps.gz`.

Menzies, T. [1996], 'Applications of abduction: Knowledge level modeling', *International Journal of Human Computer Studies* **45**, 305–355. Available from `http://www.cse.unsw.edu.au/~timm/pub/docs/96abkl1.ps.gz`.

Menzies, T. [1998*a*], 'Evaluation issues for problem solv-

ing methods'. Banff KA workshop, 1998. Available from http://www.cse.unsw.edu.au/~timm/pub/docs/97eval.

Menzies, T. [1998*b*], 'Towards situated knowledge acquisition', *International Journal of Human-Computer Studies* **49**, 867–893. Available from http://www.cse.unsw.EDU.AU/~timm/pub/docs/98ijhcs.

Menzies, T. [1999], 'Knowledge maintenance: The state of the art', *The Knowledge Engineering Review* **14**(1), 1–46. Available from http://www.cse.unsw.EDU.AU/~timm/pub/docs/97kmall.ps.gz.

Menzies, T., Black, J., Fleming, J. & Dean, M. [1992], An expert system for raising pigs, *in* 'The first Conference on Practical Applications of Prolog'. Available from http://www.cse.unsw.EDU.AU/~timm/pub/docs/ukapril92.ps.gz.

Menzies, T. & Cohen, R. [1997], A graph-theoretic optimisation of temporal abductive validation, *in* 'European Symposium on the Validation and Verification of Knowledge Based Systems, Leuven, Belgium'. Available from http://www.cse.unsw.edu.au/~timm/pub/docs/97eurovav.ps.gz.

Menzies, T., Cohen, R. & Waugh, S. [1998], Evaluating conceptual qualitative modeling languages, *in* 'Banff KAW '98 workshop.'. Available from http://www.cse.unsw.EDU.AU/~timm/pub/docs/97evalcon.

Menzies, T. & Compton, P. [1997], 'Applications of abduction: Hypothesis testing of neuroendocrinological qualitative compartmental models', *Artificial Intelligence in Medicine* **10**, 145–175. Available from http://www.cse.unsw.edu.au/~timm/pub/docs/96aim.ps.gz.

Menzies, T. & Markey, B. [1987], A micro-computer, rule-based prolog expert-system for process control in a petrochemical plant, *in* 'Proceedings of the Third Australian Conference on Expert Systems, May 13-15'.

Morari, M. & McAvoy, T. [1986], *Chemical Process Control: CPC III*, A Cache Publication.

Newell, A. [1972], You can't play 20 questions with nature, and win, *in* W. Chase, ed., 'Visual Information Processing', New York: Academic Press, pp. 283–308.

Offen, R. & Jeffery, R. [1997], 'Establishing software measurement programs', *IEEE Software* pp. 45–53.

Preece, A. [1992], 'Principles and practice in verifying rule-based systems', *The Knowledge Engineering Review* **7**, 115–141.

Preston, P., Edwards, G. & Compton, P. [1993], A 1600 Rule Expert System Without Knowledge Engineers., *in* J. Leibowitz, ed., 'Second World Congress on Expert Systems'.

Reel, J. [1999], 'Critical success factors in software projects', *IEEE Computer* pp. 18–23.

Runkel, J. [1995], Analyzing tasks to build reusable model-based tools, *in* 'Proceedings of the 9th AAAI-Sponsored Banff Knowledge Acquisition for Knowledge-Based Systems Workshop Banff, Canada'.

Schreiber, A. T. & Birmingham, W. P. [1996], 'The sisyphus-vt initiative', *International Journal of Human-Computer Studies* **44**(3/4).

Shadbolt, N., O'Hara, K. & Crow, L. [2000], 'The experimental evaluation of knowledge acquisition techniques and methods: History, problems and new directions', *International Journal of Human-Computer Studies* . (to appear).

Shaw, M. [1988], Validation in a knowledge acquisition system with multiple experts, *in* 'Proceedings of the International Conference on Fifth Generation Computer Systems', pp. 1259–1266.

Tambe, M. & Rosenbloom, P. [1994], 'Investigating production system representations for non-combinatorial match', *Artificial Intelligence* **68**(1).

Waugh, S., Menzies, T. & Goss, S. [1997], Evaluating a qualitative reasoner, *in* A. Sattar, ed., 'Advanced Topics in Artificial Intelligence: 10th Australian Joint Conference on AI', Springer-Verlag.

Yu, V., Fagan, L., Wraith, S., Clancey, W., Scott, A., Hanigan, J., Blum, R., Buchanan, B. & Cohen, S. [1979], 'Antimicrobial Selection by a Computer: a Blinded Evaluation by Infectious Disease Experts', *Journal of American Medical Association* **242**, 1279–1282.

Zlatareva, N. [1993], Distributed verification and automated generation of test cases, *in* 'IJCAI '93 workshop on Validation, Verification and Test of KBs Chambery, France', pp. 67–77.