Evaluation Issues for Problem Solving Methods

Tim Menzies
Artificial Intelligence Department
School of Computer Science and Engineering
The University of NSW
timm@cse.unsw.edu.au
http://www.cse.unsw.edu.au/~timm

October 29, 1997

Abstract

Research into problems solving methods (PSMs) has identified numerous possible benefits for this approach. However, given the current state-of-the-art in PSM evaluation, these benefits cannot yet be demonstrated. This paper critically evaluates the published PSM research to argue that further evaluations are now required. In particular, PSMs should be comparatively evaluated in order to test if PSMs offer a comparatively better approach to expert systems development.

1 Introduction

> *There are two kinds of truth, small truth and great truth. You can recognize a small truth because its opposite is a falsehood. The opposite of a great truth is another truth.* Neils Bohr

Problem solving methods (PSMs) are one of the main topics in modern knowledge acquisition. This paper argues the PSM literature is mostly *exploratory research*; i.e. for the most part it has yet to *answer* the numerous *questions* it has found concerning knowledge engineering.

Cohen [Cohen, 1995] distinguishes between exploratory research and *evaluation research*: exploratory research enters a new area and finds questions while evaluation research tries to answer those questions. PSM research results (e.g. [Linster, 1992, Schreiber & Birmingham, 1996]), are rarely presented as comparative empirical results; i.e. some experiment is run multiple times with some variation between each trial (exceptions: [Corbridge *et al.*, 1995, Gil & Tallis, 1997a, Motta & Zdrahal, 1996, Runkel, 1995, Zdrahal & Motta, 1996]). That is, PSMs have found questions, but not answers.

This is not because such empirical evaluations are impossible to explicate or collect. Numerous examples exist in the none-PSM knowledge engineering literature where a thorough empirical analysis has been performed (e.g. [Gordon & Shortliffe, 1985, Hayes, 1997, Lee & O'Keefe, 1996, Menzies *et al.*, 1992, Preston *et al.*, 1993, Reich, 1995, Vicente *et al.*, 1995, Waugh *et al.*, 1997, Weiss *et al.*, 1978, Yost, 1992, Yu *et al.*, 1979]).

This lack of empirical evaluation does not mean that the problem solving method research is somehow fatally flawed. The real strength of the PSM programme, in

my view, is that it lets us define questions about knowledge engineering which can be evaluated. That is, the PSM literature should be viewed as defining a evaluation programme that should keep us quite busy for many years to come. In this respect, PSMs are a tremendous success (in terms of the above Bohr quote, PSMs are a *great truth*).

Nor does this lack of empirical evaluations mean that PSM researchers are guilty of poor science. While this paper will be critical of many of the experiments in the PSM literature, it must be stressed that it takes time to design a good experiment. Better experimental designs are created via recognising the flaws in older designs. We cannot get to the better designs without the earlier versions. Hence, the current generation of PSM experiments are essential initial steps towards a viable empirical evaluation programme.

This paper does not offer experimental designs for empirical evaluations (my own view of such designs are recorded elsewhere; see [Menzies, 1995, Waugh *et al.*, 1997, Menzies *et al.*, 1997, Menzies & Cohen, 1997, Menzies, 1996b]). Before we can declare some evaluation design to be *good*, we need an initial discussion on what constitutes a good evaluation. Further, we need strong motivation to pursue evaluations since, generally speaking, evaluations involve a lot of work. The aim of this paper is hence to review evaluation principles and motivate further work in this area.

I will proceed as follows. Firstly, some briefing notes are presented on empirical methods which set the stage for a review of the list-of-claims made in the PSM literature (e.g. [Angele *et al.*, 1996, Benjamins, 1995, Breuker & de Velde (eds), 1994, Chandrasekaran *et al.*, 1992, Chandrasekaran *et al.*, 1992, Clancey, 1992, Eriksson *et al.*, 1995, Marques *et al.*, 1992, Motta & Zdrahal, 1996, Schreiber *et al.*, 1994, Shadbolt & O'Hara, 1997, Steels, 1990, Swartout & Gill, 1996, Tansley & Hayball, 1993, Wielinga *et al.*, 1992, Wielinga *et al.*, 1997, Gil & Melz, 1996]). Secondly, contradictory evidence will be presented to this list-of-claims. Those claims are that PSMs support the following:

- Claim 1: The construction of adequate KBS systems:
  - Claim 1.1: Which are computationally tractable
  - Claim 1.2: Which are successful in achieving some task
- Claim 2: Simplification of KBS construction, including:
  - Claim 2.1: Initial acquisition
  - Claim 2.2: Design and implementation
  - Claim 2.3: Better testing
  - Claim 2.4: Better maintenance
- Claim 3: Reusing knowledge
- Claim 4: Explanations

Another tacit assumption in the PSM literature will also be analysed:

- Claim 5: PSMs achieves 1..4 in a manner superior to other approaches.

This article assumes that the reader is familiar with PSMs (for an short introduction to PSM-based approaches, see [Linster & Musen, 1992]).

2 Empirical Methods

One way to assess a system is to carefully measure its behaviour in some experiment. Standards for careful observations of software experiments have been extensively researched. This section summarises the portions of that research which will be used in our subsequent discussion. For a full discussion, see [Fenton, 1991, Cohen, 1995]. For introductory remarks to experimental methods, software measurement, and the evaluation of expert systems, see [Reich, 1995, Fenton *et al.*, 1994, Gaschnig *et al.*, 1983]. For examples of good empirical evaluations, see [Yu *et al.*, 1979, Corbridge *et al.*, 1995, Menzies, 1996b, Vicente *et al.*, 1995, Sanderson *et al.*, 1989]. For examples of very good empirical evaluations, see [Hayes, 1997, Yost, 1992].

2.1 How to Evaluate

Certain basic principles for measurement should always be observed.

Basili [Basili, 1992], characterises 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. Fenton [Fenton, 1991] offers a theoretical and pragmatic analyses what makes for a good measurement. Good measurement programs must measure how a product was generated (process measures); what was generated (product measures); and what resources (e.g. time, skill level of developers) were used in the production.

Fenton's comments on resource measurements caution us that a piece of software should be tested on a different population to those used in developing it. In machine learning terms, this means training on one set of examples, then testing on an unseen set of examples [Quinlan, 1986]. In human-in-the-loop knowledge acquisition systems, this means that the developers of a system should try the system out on other people. Otherwise, we could encounter the *resource conflation problem*: i.e. the results could confuse the skill of the developer with the intrinsic value of the tool.

Measurements should be specified enough such that another researcher can reproduce them. Also, it is useful to have an active refutable hypothesis. A good experimenter defines a observation which, if seen would refute some active hypothesis.

Another useful principle is the *straw man*: i.e. some variant of the studied technique that appears to be obviously inferior. Straw men allow us to be surprised since, sometimes, straw men do not burn; i.e. the apparently stupider approach may sometimes perform as well as an apparently more sophisticated approach (e.g. see the Corbridge study below).

Experimental instrumentation must be *calibrated* using *baseline values* and *gold standards*. It is useful to compare measurements with some baseline value. For example, if we measure (e.g.) 43 in an experiment, that tells us less than if we can measure 43, then contrast that with the known baseline value of (e.g.) 21. Also, beware of *ceiling* and *floor* effects; i.e. experiments in which all the measurements are stuck around some highest or lowest figure. Ceiling and floor effects do not allow us to distinguish variables in an experiment. For example, suppose all students score 100 percent in a university entrance exam. That test would be a waste of time since it cannot rank candidates. Straw men are good for identifying floor effects: if all the measures are clustered around the measures for the straw man, then the measure should be changed. Lastly, as well as a baseline, it is useful to be able to compare the results to some have some objective gold standard.

Sample sizes (N) should be carefully controlled. Small sample sizes are hard to analyse. However, as random sample size get larger, they approach a bell shape (the normal distribution) which is a well understood distribution. In practice, N greater than 20 is acceptable and N greater than 30 is encouraged. On the other hand, there may be no benefit with making N very large (Cohen argues that sample sizes of N greater than 50 can be pointless [Cohen, 1995], p116). Spurious correlations can occur in large sample sizes which may require further experimentation or statistical analysis to confirm or deny [Courtney & Gustafson, 1983]. In general, when dealing with large sample sizes, it is best to restrict the conclusions to an analysis of the active hypothesis that prompted the experiment.

The above basic requirements do not tell us what numbers to collect. This issue has been addressed in the literature. [Buchanan & Shortliffe, 1984a, Gaschnig *et al.*, 1983] note that the success criteria should reflect end-user concerns and not internal criteria. For example, in the PIGE farm-management expert systems, the evaluation was not (e.g.) number of productions fired per second. Rather, it reflected the concerns of the population of farmers who might wish to buy the package. We used the following evaluation criteria: increased profitability per square meter per day [Menzies *et al.*, 1992].

Cohen remarks that:

> *Programs are not experiments, but rather the laboratory in which experiments are conducted* [Cohen, 1995], p xiii.

We should not ask experts to evaluate a program merely by watching it run. One side-effect of evaluations studies is the observation that, often, experts disagree ([Gaines & Shaw, 1989, Shaw, 1988, Gaschnig *et al.*, 1983, Yu *et al.*, 1979]). 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.

In the KBS literature, there are two prominent examples of sol-called *evaluations* that are better characterised as *program watching*. The PSM community have the Sisyphus studies [Linster, 1992, Schreiber & Birmingham, 1996] just as the older expert systems community had the *Oak Ridge spill* study [Barstow *et al.*, 1983, Johnson & Jordan, 1983]. In both studies, a group of international researchers agreed to develop systems for some common problem. The Sisyphus studies had a better criteria for success than Oak Ridge (though in the case of Sisyphus-II, it is not clear if all the participants meet that criteria: see below). With few exceptions (e.g. [Zdrahal & Motta, 1996], see below), reports by Sisyphus or Oak Ridge participants comprise reports of a single run of the system. Experimentation requires more than one run. Data is collected for each run and something is slightly different for each trial. Both these studies were useful in unifying and focusing the work of a large number of researchers. As such, they were a tremendous success (and the Sisyphus experiments are continuing). However, as we shall see below, these studies do not qualify as comparative empirical evaluation studies.

The opposite of the halo effect is when the recommendations of the expert system are rejected merely because some judge knows that the recommendations come from a computer program. [Gaschnig *et al.*, 1983] hence recommends *blinding* studies. In

such blinding studies, the evaluating agent is not told recommendations come from the expert systems and which come from other sources.

2.1.1 After the Evaluation

After performing one evaluation, your work does not stop there. A good experimenter critically reviews their results (if they don't, someone else will) and look for ways to improve them. For example, the MYCIN evaluation study [Yu *et al.*, 1979] took five years and two earlier versions to define adequately. Faults with the prior versions were used to design the next version [Buchanan & Shortliffe, 1984a].

This self-critique stage is very important. There are numerous examples in the literature where evaluation stopped too early. For example, the first trials with the XCON system (October to December 1979) comprised a panel of twelve experts. Ten orders for computers were configured by XCON. These configurations were assessed and the system deemed proficient. Note that this evaluation process had no well-define success criteria. In terms of the above discussion, it suffered from program watching and (possibly) the halo effect. Nevertheless, despite the results of this evaluation study, one year later, XCON was a poor configuration tool. The earlier evaluation was incomplete since it only studied a tiny fraction of the set of possible orders [Gaschnig *et al.*, 1983] (p270–271).

We should routinely expect to have to perform multiple evaluations since not everything can be measured in one experiment. An ideal experiment simultaneously scores highly on three dimensions: *specificity*: tight experiment controls; *face validity*: a correspondence of the experimental situation to the real situation; and *meaningfulness*: theoretical depth; i.e. generality and rigor. Sanderson [Sanderson *et al.*, 1989] comments that the experience of cognitive engineering and ergonomics is that:

> *It seems to be impossible to do all three at the same time. By maximizing any two, there always seems to be a compromise in the third... Individual studies might maximise different pairs of dimensions, but taken together the studies would offer converging evidence on an issue.*

Elsewhere, I have discussed the design of languages that support evaluation as an on-going process through the life cycle of an expert systems [Menzies, 1995, Menzies & Compton, 1997].

2.1.2 An Example

We illustrate some of the above points with an example taken from the PSM literature. Corbridge et.al. studied the efficacy of problem solving methods on KA [Corbridge *et al.*, 1995]. In this study, subjects had a fixed two-hour time period to extract a list of disorders and knowledge fragments from a transcript of a doctor talking to a patient. These lists were compared to a *gold standard*: a list generated using unlimited time by Corbridge et.al. The results are shown in Table 1.

In a statistical analysis of this study, there was no detected statistical difference between the "A" groups. That is, models of problem solving methods invented the night before assist KA just as well as models developed over more than a decade. Further, the "B" group were statistically better than the "A" groups. That is, using no PSM was better than using a PSM at all!!

The Corbridge et.al. study is an example of a good experiment. The experiment contains a straw man, a gold standard, and performed multiple trials with a controlled

Model	% disorders identied	% knowledge fragments identified
A.1: A quickly developed model which was to have played the role of the "straw man".	50	28
A.2: The KADS diagnosis PSM [Wielinga *et al.*, 1992]. If modern KA theory is correct, this model should have been the best KA assistant.	55	34
B.1: No model, which should have been the worst way to do KA.	75	41

Table 1: Analysis via different models

variation between each trial. Further, all the materials associated with the experiment are offered in the appendix to that paper; i.e. it is reproducible. One complaint with the study is that the time period involved (2 hours) may not reflect how industrial practioners really use tools like PSMs. In terms of comments of Sanderson, the Corbridge study was not optimised for face validity. However, since a single experiment cannot optimise for face validity *and* specificity *and* meaningfulness, this is not a fatal flaw with the Corbridge study. The best we can usually do is design multiple experiments that optimise different pairs of the above three goals.

3 Reviewing the Claims of PSMs

This section reviews the list-of-claims made by the PSM literature.

3.1 Claim 1: PSMs Enable the Construction of *Adequate* KBS

3.1.1 Claim 2.1: *Adequate* = Computationally Tractable

This section argues that the *knowledge-level*view taken by PSM research precludes arguments of computational tractability.

Expressibility vs tractability trade offs are discussed extensively in the knowledge representation (KR) literature (e.g. [Levesque & Brachman, 1985]). Solutions to a class of problems (the NP-hard problems) are known to have an exponential upper-bound on their runtimes; i.e. may be intractable. Much of the KR literature is concerned with finding restrictive cases in which a representation can be shown to tractable; i.e. worst-case runtimes are polynomial. This style of analysis requires a detailed knowledge of the syntactic structure of a knowledge base (e.g. [Tambe *et al.*, 1990, Tambe & Rosenbloom, 1994, Brachman & Levesque, 1984]). In the usual case, PSM research separates itself from symbol-level implementation detail (exceptions: [Benjamins, 1993, Fensel, 1995]). This has the advantage that many implementations could handle this functionality such as rules, frames, a statistical package, or even a human operator to operationalise the functionality However, without a commitment to symbol-level detail, a KR-style analysis of the computational tractability of a KB is impossible.

3.1.2 *Adequate* = Successfully Achieving Some Task

My reading of the current PSM literature is that experimental evaluations of PSMs are rare. Hence, it is hard to make a definitive evaluation of the success of PSM-based

Capacity	$200\frac{ft}{min}$	$250\frac{ft}{min}$	$300\frac{ft}{min}$	$350\frac{ft}{min}$	$400\frac{ft}{min}$
2000 lbs	success	success	fail	success	success
2500 lbs	fail	fail	success	success	sucess
3000 lbs	fail	success	success	success	success
3500 lbs	success	fail	fail	fail	fail
4000 lbs	fail	fail	fail	fail	fail

Table 2: The P&R local greedy search method for configuring elevators fails in 13 out of 25 legal ranges of speed and capacity. From [Zdrahal & Motta, 1996].

approaches. For example, three impressive PSM-based developments are SHELLEY: a PSM-based knowledge engineering workbench [Wielinga *et al.*, 1992]; VT: an elevator configuration system [Marcus *et al.*, 1987,Marcus & McDermott, 1989]; and Sisyphus-II: attempts to emulate VT [Schreiber & Birmingham, 1996]. These are discussed below.

An example of using the SHELLEY workbench is given in [Wielinga *et al.*, 1992]. In that example a knowledge engineer is shown mapping a transcript of an expert interview into a library of PSMs. If a user of SHELLEY decides that they are using some PSM, then the knowledge acquisition process can be directed towards collecting lists of the terminology required for that PSM. This report of SHELLEY includes no measurements which explore the efficacy of SHELLEY.

VT was built using a knowledge acquisition tool called SALT [Marcus *et al.*, 1987, Marcus & McDermott, 1989]. SALT was based on a *propose-and-revise* PSM. SALT's interface restricted itself to only collecting information relevant to that PSM. SALT automatically generated the majority of the VT rules (2130/3062=70 percent). The VT results are an impressive demonstration that, in one application, the SALT implementation technology can scale up to very large systems. However, the VT report is an *experience report* rather than an *evaluation report* of the adequacy of the system (such an evaluation report would describe data collected from numerous runs with slight variations between each trial).

While PSMs are extensively studied, their utility in working systems is rarely experimentally evaluated in the literature. For example, only one Sisyphus-II offering reports multiple runs with their implementation [Zdrahal & Motta, 1996]. Five variants on elevator speed and elevator capacity were explored. In 13 of the reported 25 runs, their system could not configure the elevator Their results are shown in Table 2.

Zdrahal and Motta argue that the failures of their elevator configuration system were fundamental to the propose and revise PSM in the Sisyphus-II specification [Zdrahal & Motta, 1996]. Standard propose and revise is a local greedy search; i.e. constraint violations are fixed as they occur. Such a hill-climbing algorithm may ignore solutions which are initially unpromising, but lead later on to better solutions. That is, the above errors where not a function of the Zdrahal and Motta implementation. Rather, they should also have been seen in the other Sisyphus-II offerings if they had followed the problem specification and if they had run their program over the range of legal inputs. The fact these errors were not reported elsewhere strongly suggests that the other Sisyphus-II offerings were not extensively tested. That is, it cannot be argued that the Sisyphus-II successfully achieve the task of elevator configuration.

In summary, PSM research generally lacks experimental evaluations (exceptions: the Corbridge study and the work of Motta and Zdrahal). Hence, the claim that PSMs successfully achieve some task is still an exploratory claim, not an evaluation claim.

3.2 Claim 2: SIMPLIFICATION OF THE KBS CONSTRUCTION

3.2.1 Claim 2.1: Initial Acquisition

The Corbridge study (mentioned above) suggests that it is not necessarily so that PSMs simplify initial acquisition.

3.2.2 Claim 2.2: Design and Implementation

Do PSMs clarify or confuse design and implementation issues? Evaluations in this regard are rare so this is an open issue. Two studies of the effects of PSMs on implementation are given below. Note that they offer contradictory results: more empirical evaluation is required.

Clancey reports that after a PSM analysis of 176 MYCIN rules, he could generate a new knowledge base where 80 percent of the rules had only a single condition. Further, the problem solving strategies removed all uncontrolled backtracking [Clancey, 1992]. However, this result has not been reported elsewhere and it can hardly be called an evaluation study (no repeated runs with some slight change between each run).

Motta and Zdrahal discuss the various Sisyphus-II implementations using their special knowledge of constraint satisfaction algorithms [Zdrahal & Motta, 1994]. I find their analysis more insightful into the construction process than the less-detailed, high-level PSM approach. This low-level view of a problem can find errors that experienced PSM practioners cannot. For example, Motta and Zdrahal argue that one declarative translation of the procedures in the Sisyphus-II specification blurred the distinction between hard constraints (which must not be violated) and soft constraints (which can be optionally violated) [Motta & Zdrahal, 1995].

Elsewhere [Menzies, 1996a], I have argued that PSMs can obscure important similarities. Significant similarities exist between seemingly different PSMs. If we actively explore those similarities, one basic abductive inference procedure becomes apparent: the extraction of a consistent subset of a theory that is relevant to some task. Abduction over and-or graphs can be shown to implement many of the PSMs; i.e. prediction, classification, explanation, tutoring, qualitative reasoning, planning, monitoring, set-covering diagnosis, consistency-based diagnosis, validation, and verification [Menzies, 1996a].

3.2.3 Claim 2.3: Better Testing

This section argues that the literature has yet to evaluate the testability of PSMs. While claims that PSMs improve KBS testing are common, e.g [Shadbolt & O'Hara, 1997, van Harmelen & ten Teije, 1997, van Harmelen & Aben, 1996, Fensel & Schoenegge, 1997]. However, none of this work has yet to devise an experimental evaluation of their claims that *PSMs = better testing.*

A well-developed area of testing research is the knowledge base verification and validation (V and V) community. Nearly all V and V work focuses on an analysis of the dependency networks between literals in a rule-base; e.g. [Preece, 1992] (exceptions: [Fensel *et al.*, 1996, van Harmelen & ten Teije, 1997, van Harmelen & Aben, 1996, Menzies & Compton, 1997, Menzies, 1996b, Waugh *et al.*, 1997, Zlatereva & Preece, 1994]). As such, it is usually a *symbol-level* analysis. One of the premises of knowledge level modeling is that intelligence can be analysed using knowledge *content* rather than knowledge *form*; i.e. it is irrelevant if it is expressed in rules or frames or *C* code or

whatever. Hence, for the most part, PSMs cannot utilise the symbol-level V and V techniques. Note that this rejection of symbol-level analysis was seen above when PSM tractability was discussed.

In my own research into KBS testing I encountered numerous issues which are just not addressed in the PSM literature. Firstly, in order to properly validate a theory, some outside source of knowledge must be mentioned. Generally, this *quality assessment knowledge* or *social context knowledge* must reflect how the knowledge base will be perceived when it is deployed. Secondly, in my view [Menzies & Compton, 1997], a real world test engine for a knowledge base has to handle inconsistent theories being developed in poorly measured domains. Routinely, such a test engine has to make assumptions and mutually exclusive assumptions must be kept in separate worlds. Once they worlds are generated, a preference criteria must be applied to heuristically select the *best* world. Using the criteria of *return the world(s) which contain the most known outputs*, we can fault published theories of neuroendocrinology using the data published to support them (a result first obtained by Feldman and Compton [Feldman *et al.*, 1989], then repeated and generalised by myself and Compton [Menzies & Compton, 1997]). This test engine has been experimentally evaluated in numerous studies in which various aspects of the theory being analysed were varied (e.g. theory size and connectivity [Menzies, 1996a], effects of conjunctions in the theory [Menzies *et al.*, 1997] and different interpretations of time [Waugh *et al.*, 1997]).

3.2.4 Claim 2.4: Better Maintenance

Maintenance is different to validation and verification. A good maintenance strategy permits knowledge modification but blocks changes which make changes harder in the future. That is, changes to section *I* of KB do not also break sections *J*, *K*, *L*,... Three significant explorations of the maintenance problem are ripple down rules [Compton & Jansen, 1990, Preston *et al.*, 1993]; the RIME editor [Bachant & McDermott, 1984, de Brug *et al.*, 1986, Soloway *et al.*, 1987]; and KA scripts [Gil & Tallis, 1997a]. Of these, only KA scripts comes from the PSM community. Ripple down rules (discussed below) abandons the idea of knowledge level altogether. RIME was not based on PSM-principles:

- RIME had a single hard-wired *PSM*: control of operator selection in problems space traversal.
- Problem-solving methods are explicit in standard PSM but are implicit in problem-space traversal systems. The observation that a problem-space traversal system is performing (e.g.) classification is a user-interpretation of a lower-level inference [Yost & Newell, 1989].

This section argues that the KA scripts results represent an interesting exploration of PSMs, but not an evaluation of the maintainability of PSMs. A comparative analysis of *do PSMs provide better maintenance than alternative approaches?* is done below (see claim 5).

In the case where numerous changes have to be made to a PSM, if the user does not complete all those changes, then the PSM may be broken. Gil and Tallis [Gil & Tallis, 1997a] use a scripting language to control the modification of a PSM to prevent broken knowledge. These *KA scripts* are controlled by the EXPECT TRANSACTION MANAGER (ETM) which is triggered when EXPECT's partial evaluation strategy detects a fault (errors are detected if a method cannot fire because the types of the

	Simple task #1				Harder task #2			
	no ETM		with ETM		no ETM		with ETM	
	S4	S1	S2	S3	S2	S 3	S1	S2
Total time (min)	25	22	19	15	74	53	40	41
Time completing transactions	16	11	9	9	53	32	17	20
Total changes	3	3	3	3	7	8	10	9
Changes made automatically	n/a	n/a	2	2	n/a	n/a	7	8

Table 3: Change times for ETM with four subjects: S1...S4. From [Gil & Tallis, 1997b]

input parameters to the methods are not available). In one study of maintenance times by four subjects (S1..S4) and two change tasks for EXPECT KBS, maintenance was easier with ETM. For example, ETM performed some changes automatically. The Gill & Tallis results are shown in Table 3.

These results cannot be read as an evaluation results demonstrating that PSMs simplify maintenance. While some attempt was made to perform repeated trials whilst varying some factor, the results do not satisfy several Fenton's software measurement criteria:

- Product measures: The two tasks undertaken are not described. We therefore cannot get a sense of the complexity of the tasks involved. Also, there is no success criteria offered for the tasks; i.e. it is not clear if the subjects *successfully* made changes to the system. Further, since the tasks were not defined, we cannot attempt to reproduce this experiment (In fairness to the authors of [Gil & Tallis, 1997a], it should be noted that this report was made at a conference with strict and short page limits).

- Resource measures: It is possible that some of the subjects in the [Gil & Tallis, 1997a] experiment were the authors of the ETM tool. If this were true, then we have the resource conflation problem.

Also, the results conflate three effects:

- It is possible that the ability to browse around the search space which the proof procedure could or has traversed is the major contribution of this work.

- It is also possible that transaction control over changes to procedural knowledge is a technique that could benefit the editors of any procedural representation, not just PSMs

- It is possible that PSMs enable the above two benefits. But this is not clear from these results.

3.3 Claim 3: Reusing Knowledge

Reuse is the holy grail of software and knowledge engineering. This section argues that the reusability of PSMs has not been adequately evaluated.

My reading of the PSM literature is that PSMs change more than they are reused. Between the various camps of PSM researchers, there is little agreement on the details of the PSMs. The list of primitives within the PSMs (e.g. select, classify, etc) from KADS [Wielinga *et al.*, 1992] and the SPARK/ BURN/ FIREFIGHTER project [Marques

et al., 1992] are significantly different. Also, the number and nature of the problem solving methods is not fixed. Often when a domain is analysed using PSM, a new method is induced [Linster & Musen, 1992].

When we look at published problem solving methods, we see many differences. For example, [Menzies, 1998] describes eight different supposedly reusable models of diagnosis (four from the PSM community, four from elsewhere). While some of the these views on diagnosis share some common features, they reflect fundamentally divergent different views on how to perform diagnosis. I therefore believe that, at least in the case of diagnosis, a consensus view on diagnosis has not stabilised with time and that such a view may not do so in the foreseeable future. More generally, since PSMs have not stabilised over time, their extensive reuse is hence unlikely. My reading of the current literature is that my no-reuse argument cannot be faulted due to drawbacks with the PSM reuse evaluations studies.

Two major studies in PSM-based reuse are the SPARK/ BURN/ FIREFIGHTER (hereafter, SBF) experiment [Marques *et al.*, 1992] and the MeKA study [Runkel, 1995]. In the SBF toolkit, SPARK builds a domain-specific KA tool that is tailored to the business information supplied by the user. BURN conducts a structured interview with the expert. This interview maps the business information offered by the user into a library of inference sub-routines (called mechanisms). The mapping process is guided by PSM meta-knowledge. At choice points in the mapping, SBF can ask the user questions questions which select different PSMs. Once this mapping has been made, a rule base can be generate which solves the business problem. This is given to the FIREFIGHTER environment which assists the user in executing and debugging the operationalised program. Marques et.al. report significantly reduced development times for expert systems using the 13 mechanisms in the SBF toolkit. In the nine applications studied by Marques et.al., development times changed from one to 17 days (using SBF) to 63 to 250 days (without using SBF).

To the best of my knowledge, this study represents the high-water mark in reported productivity increases in software or knowledge engineering. Nevertheless, the study has its drawbacks:

- Poor controls on product measures: There is no success criteria offered for each application. Hence it is possible that the SBF-based application are somehow inferior to the none-SBF applications.

- Poor controls on resource measures: Who were the personnel who worked on the SBF development? If they were the SBF developers, then like the KA scripts study, these we may have the resource conflation problem.

- The SBF experiment is virtually unrepeatable. Only a few organisations like DEC can spare the personnel to work for nearly a year on throw-away prototypes.

In the MeKA study, Runkel describes eight applications using *mechanisms for knowledge acquisition*, or MeKA. Each MeKA divided a PSM into data structure knowledge and control knowledge. MeKAs contain four modules: (i) an acquire/ module which gathers information such as a formula; (ii) a verify module that checks it; (iii) a generalise module which tries to apply the new knowledge to more general expressions, e.g. is the formula applicable to other parameters?; (iv) a dialogue module which handles the screen design for the other modules. All the MeKAs had to be built for the first application (0 percent reuse), but MeKA reuse in subsequent applications rose as high as 88 percent. The Runkel results are shown in Table 4.

Development order	Application name	$\frac{Reused\ MeKAs}{Total\ MeKAs}$
1	Room assignment	$\frac{0}{7} = 0\%$
2	Elevator configuration	$\frac{5}{8} = 63\%$
3	Elevator design validation	$\frac{7}{8} = 88\%$
4	Configuration validation	$\frac{6}{7} = 86\%$
5	Truck design #1	$\frac{5}{9} = 56\%$
6	Truck pricing	$\frac{10}{12} = 83\%$
7	Truck design #2	$\frac{8}{16} = 50\%$
8	Truck manufacturing	$\frac{15}{17} = 88\%$

Table 4: Reuse in the MeKA system. From [Runkel, 1995].

The MeKA study is a better experiment than SBF in that the MeKA work has more chance of being reproducible. However, it terms of evaluating the reusability of PSMs, the MeKA study has some drawbacks. Runkel does not comment on who built the applications. If was himself, then once again we may have the resource conflation problem. Also, unlike the SBF study, Runkel does not record the time taken to build each application. That is, if Runkel's goal related to productivity improvements, his measurements could not address that question.

3.4 Claim 4. Better Explanations

Wielinga et.al. argue that one of the advantages of KADS (a technique which uses PSMs) is that its superior ability to explain the inner workings of an expert system [Wielinga *et al.*, 1992]. Clancey's *Heuristic Classification* paper [Clancey, 1985] is an impressive PSM-based reverse engineering of numerous expert systems in terms of his heuristic classification technique. The reader is left with a strong impression that heuristic classification explains the inner-workings of the surveyed expert systems. This section repeats the arguments of [Menzies & Compton, 1994] to argue that this strong impression may not be true.

From an empirical evaluation perspective, it is still an open question if PSM-based approaches produce better explanations. Current thinking in the explanation field (e.g. [Wick & Thompson, 1992, Leake, 1991, Leake, 1993, Paris, 1989]) is that good explanations cannot be generated merely via an abstract trace of the system's traversal over a task description (e.g. a PSM) or the *print the rules that fired* approach used in early expert systems such as MYCIN [Buchanan & Shortliffe, 1984b]). In the current view, explanation is a problem solving task in its own right. Explanations are user-specific:

> *The audience of an explanation can significantly affect the purpose and therefore the content of an explanation* [Wick & Thompson, 1992]

Explanation is an inference procedure that determines what is to be presented to the user. Leake [Leake, 1991] and Paris [Paris, 1989] discuss explanation algorithms where explanation presentation is constrained to those explanations which contain certain significant structures. Paris's significant structures are determined at design time while Leake assigns significance at runtime. For example, when the goal of the explanation is to minimise undesirable effects, the runtime significant structures are any pre-conditions to anomalous situations. From a range of possible inferences, some subset is selected

that meets some understandability criteria for different users and different goals. That is, a model that is good for explanatory purposes contains some degree of indeterminacy (can generate more than one behaviour). Also, Leake argues convincingly that a cache of prior explanations and an active user model are essential components of a good explanation module [Leake, 1993].

User-profiles, indeterminate models, and case libraries are not issues addressed in current PSMs approaches. Therefore, in their current form, PSMs may not be a good generalised explanation tool.

3.5 Claim 5. PSMs Achieve Claims 1..4 in a Manner Superior to Other Approaches

Consider the statement *software technology X lets me do task Y*. This is hardly an evaluation statement since there may be many software technologies that allow us to implement Y. A better statement is comparative: *software technology X lets me do task Y better than software technology Z*. In order to compare an approach, we need to identify an alternative approach. This section records certain alternatives to PSMs which have demonstrated competency: problem-space traversal systems, standard software engineering, ripple down rules, and human-computer interaction. At the very least, these systems are *straw men*. PSMs should at least be able to out-perform the following, apparently less sophisticated, techniques.

3.5.1 Standard Software Engineering

Significant levels of reuse are already reported in the standard software engineering literature. Stark reports code reuse levels of 70-80 percent using FORTRAN and some object-oriented design principles [Stark, 1993]. Frakes and Fox found maximum median values for reuse in requirements, design, and code reuse at 15, 70, and 40 percent respectively [Frakes & Fox, 1995]. Frakes and Fox found no significant correlation between reuse and technology options such as the use of CASE tools; the presence of code repositories; or *language level* (assembler has a lower language level than object-oriented languages such as Smalltalk); The factors that were positively correlated to reuse were all organisational factors such as practioner education in reused; unified software process; or industry type (telecommunications always was one of the highest reusers, possibly due to the standard hardware configurations in that field).

These reports of Stark, Frakes and Fox suffer from inexact measures of reuse. For example, the Frakes and Fox study never looked at source code: it's data was based on a questionnaire sent and returned by post. Nevertheless, this work does show that some levels of significant reuse may be achievable without technology options such as PSMs. The empirical evaluation goal of PSM researchers should be some test that PSMs produce higher levels of reuse that (e.g.) standard software engineering.

3.5.2 Problem-Space Traversal Systems

One of the largest studies in knowledge maintenance is the RIME's KB editor [de Brug *et al.*, 1986, Soloway *et al.*, 1987]. RIME acquired parts of the meta-knowledge for the XCON computer configuration system [Bachant & McDermott, 1984]. RIME is a problem-space traversal system. It assumes that the KB comprised operator selection knowledge which controlled the exploration of a set of problem spaces. After asking a few questions, RIME could auto-generate complex executable rules. RIME has not

been evaluated in an empirical experiment (but see [Soloway *et al.*, 1987]). Nevertheless, RIME is a landmark system. Other PSM-based maintenance research has yet to make an empirical case that their techniques are superior to RIME.

Another problems-space traversal system was Yost's Sisyphus-II contribution [Yost, 1994]. The rest of the Sisyphus-II contribution were all PSM approaches. I can see no evidence of productivity benefits of PSM over Yost's system in the Sisyphus-II results. Indeed, Yost's "old-fashioned" problem-space traversal system was developed in times comparable to SALT and faster than many of the other PSM approaches.

These comments on comparative evaluations and Sisyphus-II could be criticised as follows. The sample size was too small and too inaccurately measured to yield meaningful results (most Sisyphus-II groups were less-than-rigorous in documenting their development times except for Runkel et.al. [Runkel & Birmingham, 1994] and Yost [Yost, 1994]). I return below to the issue of comparative analysis of maintenance techniques (see ripple down rules). For the moment, all we need to say is that the comparative advantage of PSMs for maintenance over none-PSM approaches is an open issue in the literature.

3.5.3 Ripple Down Rules

Compton takes a weak situated cognition [Menzies, 1997] line, which he calls *justification in context*, His argument is that *patching in the context of error* is a more realistic KA approach than assuming that a human analyst will behave in a perfectly rational way to create some initial correct design [Compton & Jansen, 1990].

RDR implements this patching process. The RDR representation is optimised for fault localisation in KBS without PSMs [Compton & Jansen, 1990, Compton *et al.*, 1993]. RDR knowledge is organised into a *patch tree*. If a rule is found to be faulty, some patch logic is added on a *unless* link beneath the rule. The patch is itself a rule and so may be patched recursively. Whenever a new patch (rule) is added to an RDR system, the case which prompted the patch is included in the rule. These *cornerstone cases* are used below when fixing an RDR system. At runtime, the final conclusion is the conclusion of the last satisfied rule. If that conclusion is faulty, then the fault is localised to the last satisfied rule. Once a fault is localised, an expert can then ask the system for a list of possible patches. The system replies with a *difference list* which is calculated as follows. As the *current case* navigates down the RDR tree, if it finds a some satisfied rule, it then checks their *unless* patches. The different between the current case and the cornerstone case of the last satisfied rule is the difference list. In terms of empirical evaluation, RDR is exceptional in that RDR builds evaluation into the life cycle of the whole system. At anytime, the current RDR tree can correctly classify 100 percent of the cases seen to date.

RDR trees are a very low-level representation. RDR rules cannot assert facts that other RDR rules can use. In no way can a RDR tree be called a model in a PSM sense. Further, the RDR formalism makes no commitment to tree structures that are optimal. An RDR tree can contain repeated tests, redundant knowledge, and its sub-trees can overlap each other semantically. Despite these apparent drawbacks, RDR has produced large working expert systems in routine daily use. In practice the RDR trees are only twice as big as the optimum tree [Gaines & Compton, 1992] and runtimes have never been an issue. It may be somewhat misguided to attempt to optimise an RDR tree to (e.g.) remove the redundancies or separate the overlaps. The important feature of an RDR tree is that is it optimised for maintenance. Alternative representations may run faster, but incurs the penalty of more complicated maintenance.

In practice, RDR appears to work very well. For example, the PIERS system at St. Vincent's Hospital, Sydney, modeled 20 percent of human biochemistry sufficiently well to make diagnoses that are 95 percent accurate [Preston *et al.*, 1993]. RDR has succeeded in domains where previous attempts, based on much higher-level constructs, never made it out of the prototype stage [Patil *et al.*, 1981]. Further, while large expert systems are notoriously hard to maintain [de Brug *et al.*, 1986], the no-model approach of RDR has never encountered maintenance problems. System development blends seamlessly with system maintenance since the only activity that the RDR interface permits is patching faulty rules in the context of the last error. For a 2000-rule RDR system, maintenance was very simple (a total of a few minutes each day).

3.5.4 Human-Computer Interaction

The core of the competency of problem-space traversal systems and ripple down rules is some knowledge base. However, it may be possible to build a system that supports (e.g.) diagnosis without such a knowledge base. For example, [Vicente *et al.*, 1995] discusses diagnosis, and fault detection using 'ecological interface design' (EID). An EID contains visual representations at five levels of an abstraction hierarchy:

- The *functional purpose*: visualisations of the purpose for which the interface was designed;
- The *abstract function*: visualisations of the intended causal structure of the process in terms of mass, energy, information or value flows;
- The *generalised function*: visualisations of the basic functions that the device is designed to achieve;
- The *physical function*: visualisations of the characteristics of the components and the connections between them;
- The *physical form*: visualisations of the appearance and spatial location of the components.

Interestingly, none of this abstraction hierarchy may exist in a knowledge base of an EID system. Rather, the above five principles are used by an interface designer when re-arranging and augmenting the display on the screen. No inference engine accesses this hierarchy at runtime in the conventional knowledge representation sense.

[Vicente *et al.*, 1995] suggest that EID-based interfaces lead to better performance on diagnosis tasks when subjects are simply asked to monitor the physical While it was not the intention of the authors of [Vicente *et al.*, 1995] to do so, this work offers a challenge to the standard KBS view of model-based diagnosis and repair. Rather than build sophisticated KBS systems, perhaps we should be looking at better interface design? At the very least, it seems to me that the KBS community should attempt a comparative evaluation of KBS-style diagnosis and fault detection vs EID-based diagnosis and fault detection.

4 Conclusion

> *Give me a fruitful error any time, full of seeds, bursting with its own corrections. You can keep your sterile truths for yourself.* Vilfredo Pareto

PSM research is still exploratory; i.e. it lets us define questions for knowledge engineering (the list-of-claims). However, PSM research is not evaluatory; i.e. the questions it asks have yet to be answered using empirical evaluation techniques. Evaluation is very important. Before we can sell PSMs to the wider knowledge engineering community, we should be able to demonstrate that PSMs are valuable. Such demonstrations are lacking in the current literature. We should work towards performing such evaluations in the future.

This article has analysed the drawbacks with current PSM evaluation studies. This analysis can be used to avoid certain traps in future PSM evaluations such as:

- *Lack of empirical experiments*: With few exceptions, PSM evaluations are not conducted via some experiment conducted multiple times with some variation between each trial.
- *Poor calibration*: When empirical evaluations are presented, their measurements lack baselines or comparisons to alternative approaches. Such comparisons are possible, given the numerous *straw men* documented above such as ripple down rules, EID, and problem-space traversal systems.
- *Lack of experimental critique*: Using standard metrics theory, we can find flaws in the experimental designs of the few known PSM evaluations. Such flaws should have been commented on previously in the literature. An analysis of these flaws will show what kind of experimental evaluations we should conduct in the future.

This critique has only focused on PSMs. A similar critique (too much exploration, not enough evaluation) cannot yet be applied to the ontologies research (e.g. [Gruber, 1993]). PSMs research is at least 14 years old [Chandrasekaran, 1983] while ontologies are a much newer concept which are being explored. However, I would hope that within the next five years, ontological researchers move away from exploratory research towards evaluation research.

References

[Angele *et al.*, 1996] Angele, J., Fensel, D., & Studer, R. (1996). Domain and Task Modelling in MIKE. In et.al., A. S., (Ed.), *Domain Knowledge for Interactive System Design*. Chapman & Hall.

[Bachant & McDermott, 1984] Bachant, J. & McDermott, J. (1984). R1 Revisited: Four Years in the Trenches. *AI Magazine*, pages 21–32.

[Barstow *et al.*, 1983] Barstow, D., Aiello, N., Duda, R., Erman, L., Forgy, C., Gorlin, D., Greiner, R., Lenat, D., London, P., McDermott, J., Nii, H. P., Politakis, P., Reboh, R., Rosenchein, S., Scott, A., van Melle, W., & Weiss, S. (1983). Languages and Tools for Knowledge Engineering. In Hayes-Roth, F., Waterman, D., & Lenat, D., (Eds.), *Building Expert Systems*, chapter 9, pages 283–345. Addison-Wesley.

[Basili, 1992] Basili, V. R. (1992). The Experimental Paradigm in Software Engineering. In Rombach, H. D., Basili, V. R., & Selby, R. W., (Eds.), *Experimental Software Engineering Issues: Critical Assessment and Future Directions, International Workshop, Germany*, pages 3–12.

[Benjamins, 1993] Benjamins, R. (1993). *Problem Solving Methods for Diagnosis*. PhD thesis, University of Amsterdam.

[Benjamins, 1995] Benjamins, R. (1995). Problem-Solving Methods for Diagnosis and their Role in Knowledge Acquisition. *International Journal of Expert Systems: Research & Applications*, 8(2):93–120.

[Brachman & Levesque, 1984] Brachman, R. & Levesque, H. (1984). The Tractability of Subsumption in Frame-Based Description Languages. In *AAAI '84*, pages 34–37.

[Breuker & de Velde (eds), 1994] Breuker, J. & de Velde (eds), W. V. (1994). *The CommonKADS Library for Expertise Modelling*. IOS Press, Netherlands.

[Buchanan & Shortliffe, 1984a] Buchanan, B. & Shortliffe, E. (1984a). *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project*, chapter 10. Uncertainty and Evidential Support, pages 209–232. Addison Wesley.

[Buchanan & Shortliffe, 1984b] Buchanan, B. & Shortliffe, E. (1984b). *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project*. Addison-Wesley.

[Chandrasekaran, 1983] Chandrasekaran, B. (1983). Towards a Taxonomy of Problem Solving Types. *AI Magazine*, pages 9–17.

[Chandrasekaran *et al.*, 1992] Chandrasekaran, B., Johnson, T., & Smith, J. W. (1992). Task Structure Analysis for Knowledge Modeling. *Communications of the ACM*, 35(9):124–137.

[Clancey, 1985] Clancey, W. (1985). Heuristic Classification. *Artificial Intelligence*, 27:289–350.

[Clancey, 1992] Clancey, W. (1992). Model Construction Operators. *Artificial Intelligence*, 53:1–115.

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

[Compton & Jansen, 1990] Compton, P. & Jansen, R. (1990). A Philosophical Basis for Knowledge Acquisition. *Knowledge Acquisition*, 2:241–257.

[Compton *et al.*, 1993] Compton, P., Kang, B., Preston, P., & Mulholland, M. (1993). Knowledge Acquisition Without Analysis. In *European Knowledge Acquisition Workshop*.

[Corbridge *et al.*, 1995] 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*.

[Courtney & Gustafson, 1983] Courtney, R. & Gustafson, D. (1983). Shotgun Correlations in Software Measures. *Software Engineering Journal*, pages 5–11.

[de Brug *et al.*, 1986] de Brug, A. V., Bachant, J., & McDermott, J. (1986). The Taming of R1. *IEEE Expert*, pages 33–39.

[Eriksson *et al.*, 1995] Eriksson, H., Shahar, Y., Tu, S. W., Puerta, A. R., & Musen, M. A. (1995). Task Modeling with Reusable Problem-Solving Methods. *Artificial Intelligence*, 79(2):293–326.

[Feldman *et al.*, 1989] 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*.

[Fensel, 1995] Fensel, D. (1995). *The Knowledge Acquisition And Representation Language KARL*. Kluwer Academic Publisher.

[Fensel & Schoenegge, 1997] Fensel, D. & Schoenegge, A. (1997). Hunting for Assumptions as Developing Method for Problem-Solving Methods. In *Workshop on Problem-Solving Methods for Knowledge-based Systems, IJCAI '97, August 23*.

[Fensel *et al.*, 1996] Fensel, D., Schonegge, A., Groenboom, R., & Wielinga, B. (1996). Specification and Verification of Knowledge-Based Systems. Available from `http://ksi.cpsc.ucalgary.ca/KAW/KAW96/fensel/svkbs.html`.

[Fenton *et al.*, 1994] Fenton, N., Pfleeger, S., & Glass, R. (1994). Science and Substance: A Challenge to Software Engineers. *IEEE Software*, pages 86–95.

[Fenton, 1991] Fenton, N. E. (1991). *Software Metrics*. Chapman and Hall, London.

[Frakes & Fox, 1995] Frakes, W. & Fox, C. (1995). Sixteen Questions About Software Reuse. *Communications of the ACM*, 38(6):75–87.

[Gaines & Compton, 1992] Gaines, B. & Compton, P. (1992). Induction of Ripple Down Rules. In *Proceedings, Australian AI '92*, pages 349–354. World Scientific.

[Gaines & Shaw, 1989] Gaines, B. & Shaw, M. (1989). Comparing the Conceptual Systems of Experts. In *IJCAI '89*, pages 633–638.

[Gaschnig *et al.*, 1983] Gaschnig, J., Klahr, P., Pople, H., Shortliffe, E., & Terry, A. (1983). Evaluation of Expert Systems: Issues and Case Studies. In Hayes-Roth, F., Waterman, D., & Lenat, D., (Eds.), *Building Expert Systems*, chapter 8, pages 241–280. Addison-Wesley.

[Gil & Melz, 1996] Gil, Y. & Melz, E. (1996). Explicit Representations of Problem-Soving Strategies to Support Knowledge Acquisition. In *Proceedings AAAI' 96*.

[Gil & Tallis, 1997a] Gil, Y. & Tallis, M. (1997a). A Script-Based Approach to Modifying Knowledge Bases. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI-97)*.

[Gil & Tallis, 1997b] Gil, Y. & Tallis, M. (1997b). A Script-Based Approach to Modifying Knowledge Bases. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI-97)*.

[Gordon & Shortliffe, 1985] Gordon, J. & Shortliffe, E. H. (1985). A Method for Managing Evidential Reasoning in a Hierarchical Hypothesis Space. *Artificial Intelligence*, 26(3):323–357.

[Gruber, 1993] Gruber, T. (1993). A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisition*, 5(2):199–220.

[Hayes, 1997] Hayes, C. (1997). A Study in Solution Quality Human Expert and Knolwedge-Based System Reasoning. In Feltovich, P., Ford, K., & Hoffman, R., (Eds.), *Expertise in Context*, chapter 14, pages 339–362. MIT PRess.

[Johnson & Jordan, 1983] Johnson, C. & Jordan, S. (1983). Emergency Management of Inland Oil and Hazardous Chemical Spills: A Case Study In Knowledge Engineering. In Hayes-Roth, F., Waterman, D., & Lenat, D., (Eds.), *Building Expert Systems*, chapter 10, pages 349–397. Addison-Wesley.

[Leake, 1991] Leake, D. (1991). Goal-Based Explanation Evaluation. *Cognitive Science*, 15:509–545.

[Leake, 1993] Leake, D. (1993). Focusing Construction and Selection of Abductive Hypotheses. In *IJCAI '93*, pages 24–29.

[Lee & O'Keefe, 1996] Lee, S. & O'Keefe, R. (1996). The Effect of Knowledge Representation Schemes on Maintainability of Knowledge-Based Systems. *IEEE Transactions on Knowledge and Data Engineering*, 8(1):173–178.

[Levesque & Brachman, 1985] Levesque, H. & Brachman, R. (1985). A Fundamental Tradeoff in Knowledge Representation and Reasoning (Revised Version). In Brachmann, R. & Levesque, H., (Eds.), *Readings in Knowledge Representation*, pages 41–70. Palo Alto, Morgan Kaufmann.

[Linster, 1992] Linster, M. (1992). A review of Sisyphus 91 and 92: Models of Problem-Solving Knowledge. In Aussenac, N., Boy, G., Gaines, B., Linser, M., Ganascia, J.-G., & Kordratoff, Y., (Eds.), *Knowledge Acquisition for Knowledge-Based Systems*, pages 159–182. Springer-Verlag.

[Linster & Musen, 1992] Linster, M. & Musen, M. (1992). Use of KADS to Create a Conceptual Model of the ONCOCIN task. *Knowledge Acquisition*, 4:55–88.

[Marcus & McDermott, 1989] Marcus, S. & McDermott, J. (1989). SALT: A Knowledge Acquisition Language for Propose-and-Revise Systems. *Artificial Intelligence*, 39:1–37.

[Marcus *et al.*, 1987] Marcus, S., Stout, J., & McDermott, J. (1987). VT: An Expert Elevator Designer That Uses Knowledge-Based Backtracking. *AI Magazine*, pages 41–58.

[Marques *et al.*, 1992] Marques, D., Dallemagne, G., Kliner, G., McDermott, J., & Tung, D. (1992). Easy Programming: Empowering People to Build Their Own Applications. *IEEE Expert*, pages 16–29.

[Menzies, 1995] Menzies, T. (1995). *Principles for Generalised Testing of Knowledge Bases*. PhD thesis, University of New South Wales.

[Menzies, 1996b] Menzies, T. (1996b). On the Practicality of Abductive Validation. In *ECAI '96*.

[Menzies, 1997] Menzies, T. (1997). Is Knowledge Maintenance an Adequate Response to the Challenge of Situated Cognition for Symbolic Knowledge Based Systems? Special issue of the International Journal of Human Computer Studies: "The Challenge of Situated Cognition for Symbolic Knowledge Based Systems". In press. Available from `http://www.cse.unsw.edu.au/~timm/pub/docs`.

[Menzies, 1998] Menzies, T. (1998). OO Patterns: Lessons from Expert Systems. *Softare Practice & Experience*. In press.

[Menzies, 1996a] Menzies, T. (September, 1996a). Applications of Abduction: Knowledge Level Modeling. *International Journal of Human Computer Studies*, 45:305–355.

[Menzies *et al.*, 1992] Menzies, T., Black, J., Fleming, J., & Dean, M. (1992). An Expert System for Raising Pigs. In *The first Conference on Practical Applications of Prolog*.

[Menzies & Cohen, 1997] 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*.

[Menzies *et al.*, 1997] Menzies, T., Cohen, R., Waugh, S., & Goss, S. (1997). Evaluating Conceptual Qualitative Modeling Languages. In *Submitted to the Banff KAW '98 workshop*. Available from `http://www.cse.unsw.EDU.AU/~timm/pub/aka97/papers`.

[Menzies & Compton, 1994] Menzies, T. & Compton, P. (1994). Knowledge Acquisition for Performance Systems; or: When can "tests" replace "tasks"? In *Proceedings of the 8th AAAI-Sponsored Banff Knowledge Acquisition for Knowledge-Based Systems Workshop, Banff, Canada.*

[Menzies & Compton, 1997] Menzies, T. & Compton, P. (1997). Applications of Abduction: Hypothesis Testing of Neuroendocrinological Qualitative Compartmental Models. *Artificial Intelligence in Medicine*, 10:145–175.

[Motta & Zdrahal, 1995] Motta, E. & Zdrahal, Z. (1995). The Trouble with What: Issues in Method-Independent Task Specifications. In *Proceedings of the 9th AAAI-Sponsored Banff Knowledge Acquisition for Knowledge-Based Systems Workshop Banff, Canada.*

[Motta & Zdrahal, 1996] Motta, E. & Zdrahal, Z. (1996). Parametric Design Problem Solving. In *Proceedings of the 10th Banff Knowledge Acquisition for Knowledge-Based System Workshop.*

[Paris, 1989] Paris, C. (1989). The Use of Explicit User Models in a Generation System for Tailoring Answers to the User's Level of Expertise. In Kobsa, A. & Wahlster, W., (Eds.), *User Models in Dialog Systems*, pages 200–232. Springer-Verlag.

[Patil *et al.*, 1981] Patil, R., Szolovitis, P., & Schwartz, W. (1981). Causal Understanding of Patient Illness in Medical Diagnosis. In *IJCAI '81*, pages 893–899.

[Preece, 1992] Preece, A. (1992). Principles and Practice in Verifying Rule-based Systems. *The Knowledge Engineering Review*, 7:115–141.

[Preston *et al.*, 1993] Preston, P., Edwards, G., & Compton, P. (1993). A 1600 Rule Expert System Without Knowledge Engineers. In Leibowitz, J., (Ed.), *Second World Congress on Expert Systems.*

[Quinlan, 1986] Quinlan, J. (1986). Induction of Decision Trees. *Machine Learning*, 1:81–106.

[Reich, 1995] Reich, Y. (1995). Measuring the Value of Knowledge. *International Journal of Human-Computer Studies*, 42(1):3–30.

[Runkel, 1995] 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.*

[Runkel & Birmingham, 1994] Runkel, J. & Birmingham, W. (1994). Solving VT by Reuse. In Gaines, B. & Musen, M., (Eds.), *Proceedings of the 8th AAAI-Sponsored Banff Knowledge Acquisition for Knowledge-Based Systems Workshop*, pages 42.1–42.28.

[Sanderson *et al.*, 1989] Sanderson, P., Verhapge, A., & Fuld, R. (1989). State-space and Verbal Protocol Methods for Studying the Human Operator in Process Control. *Ergonomics*, 32(11):1343–1372.

[Schreiber & Birmingham, 1996] Schreiber, A. T. & Birmingham, W. P. (1996). The Sisyphus-VT initiative. *International Journal of Human-Computer Studies*, 44(3/4).

[Schreiber *et al.*, 1994] Schreiber, A. T., Wielinga, B., Akkermans, J. M., Velde, W. V. D., & de Hoog, R. (1994). CommonKADS. A Comprehensive Methodology for KBS Development. *IEEE Expert*, 9(6):28–37.

[Shadbolt & O'Hara, 1997] Shadbolt, N. & O'Hara, K. (1997). Model-based Expert Systems and the Explanations of Expertise. In Feltovich, P., Ford, K., & Hoffman, R., (Eds.), *Expertise in Context*, chapter 13, pages 315–337. MIT PRess.

[Shaw, 1988] Shaw, M. (1988). Validation in a Knowledge Acquisition System with Multiple Experts. In *Proceedings of the International Conference on Fifth Generation Computer Systems*, pages 1259–1266.

[Soloway *et al.*, 1987] Soloway, E., Bachant, J., & Jensen, K. (1987). Assessing the Maintainability of XCON-in-RIME: Coping with the Problems of a VERY Large Rule-Base. In *AAAI '87*, pages 824–829.

[Stark, 1993] Stark, M. (1993). Impacts of Object-Oriented Technologies: Seven Years of Software Engineering. *J. Systems Software*, 23:163–169.

[Steels, 1990] Steels, L. (1990). Components of Expertise. *AI Magazine*, 11:29–49.

[Swartout & Gill, 1996] Swartout, B. & Gill, Y. (1996). Flexible Knowledge Acquisition Through Explicit Representation of Knowledge Roles. In *1996 AAAI Spring Symposium on Acquisition, Learning, and Demonstration: Automating Tasks for Users.*

[Tambe *et al.*, 1990] Tambe, M., Newell, A., & Rosenbloom, P. (1990). The Problem of Expensive Chunks and its Solution by Restricting Expressiveness. *Machine Learning*, 5(3):299–348.

[Tambe & Rosenbloom, 1994] Tambe, M. & Rosenbloom, P. (1994). Investigating Production System Representations for Non-combinatorial Match. *Artificial Intelligence*, 68(1).

[Tansley & Hayball, 1993] Tansley, D. & Hayball, C. (1993). *Knowledge-Based Systems Analysis and Design*. Prentice-Hall.

[van Harmelen & Aben, 1996] van Harmelen, F. & Aben, M. (1996). Structure-Preserving Specification Languages for Knowledge-Based Systems. *International Journal of Human-Computer Studies*, 44:187–212.

[van Harmelen & ten Teije, 1997] van Harmelen, F. & ten Teije, A. (1997). Validation and Verification of Conceptual Models of Diagnosis. In *European Symposium on the Validation and Verification of Knowledge Based Systems, Leuven, Belgium*.

[Vicente *et al.*, 1995] Vicente, K., Christoffersen, K., & Pereklita, A. (1995). Supporting Operator Problem Solving Through Ecological Interface Design. *IEEE Transactions of Systems, Man, and Cybernetics*, 25(4529-545).

[Waugh *et al.*, 1997] Waugh, S., Menzies, T., & Goss, S. (1997). Evaluating a Qualitative Reasoner. In *Australian AI '97. Available from* `http://www.cse.unsw.edu.au/~timm/pub/docs`.

[Weiss *et al.*, 1978] Weiss, S., Kulikowski, C., & Amarel, S. (1978). A Model-Based Method for Computer-Aided Medical Decision-Making. *Artificial Intelligence*, 11.

[Wick & Thompson, 1992] Wick, M. & Thompson, W. (1992). Reconstructive Expert System Explanation. *Artificial Intelligence*, 54:33–70.

[Wielinga *et al.*, 1997] Wielinga, B., Akkermans, J., & Schreiber, A. (1997). A Comptence Theory Approach to Problem Solving Method Construction.

[Wielinga *et al.*, 1992] Wielinga, B., Schreiber, A., & Breuker, J. (1992). KADS: a Modeling Approach to Knowledge Engineering. *Knowledge Acquisition*, 4:1–162.

[Yost, 1992] Yost, G. (1992). *TAQL: A Problem Space Tool for Expert System Development*. PhD thesis, Computer Science, Carnegie Mellon.

[Yost, 1994] Yost, G. (1994). Implementing the Sisyphus-93 Task Using SOAR/TAQL. In Gaines, B. & Musen, M., (Eds.), *Proceedings of the 8th AAAI-Sponsored Banff Knowledge Acquisition for Knowledge-Based Systems Workshop*, pages 46.1–46.22.

[Yost & Newell, 1989] Yost, G. & Newell, A. (1989). A Problem Space Approach to Expert System Specification. In *IJCAI '89*, pages 621–627.

[Yu *et al.*, 1979] 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.

[Zdrahal & Motta, 1994] Zdrahal, Z. & Motta, E. (1994). An In-Depth Analysis of Propose & Revise Problem Solving Methods. In Mizoguchi, R., Motoda, H., Boose, J., Gaines, B., & Compton, P., (Eds.), *Proceedings of the Third Japanese Knowledge Acquisition for Knowledge-Based Systems Workshop: JKAW '94*.

[Zdrahal & Motta, 1996] Zdrahal, Z. & Motta, E. (1996). Improving Conpetence by Intergrating Case-Based Reasoning and Heuristic Search. In *10th Banff Knowledge Acquisition for Knowledge-Based Systems Workshop, November 9-14, 1996, Banff, Canada*.

[Zlatereva & Preece, 1994] Zlatereva, N. & Preece, A. (1994). State of the Art in Automated Validation of Knowledge-Based Systems. *Expert Systems with Applications*, 7:151–167.

Some of the Menzies papers can be found at `http://www.cse.unsw.edu.au/~timm/pub/docs`