Knowledge Maintenance Heresies: Meta-Knowledge Complicates KM

Tim Menzies
NASA/WVU Software Research Lab, 100 University Drive, Fairmont WV 26554
<tim@menzies.com>

Figure 1. Researchers discussing knowledge maintenance with the author.

The role of a discussion paper is to provoke discussion. Hence, my comments will be heretical rather than theoretical. For the theory, see [19].

Today I want to say two things. Firstly, the knowledge maintenance (KM) problem is our next big challenge. A common method for tackling the KM problem is to use some type of meta-knowledge to represent expected/unexpected or good/bad knowledge. This meta-knowledge takes many forms including architectures [26], patterns [12], problem solving methods (PSMs) [29, 31], or ontologies [13]. Elsewhere [20], I have argued that patterns of design and architecture are very similar to PSMs and ontologies:

- Both represent abstract descriptions of supposedly common parts of many designs.
- OO patterns are typically structural; exceptions: some of the patterns in [10]
- PSMs patterns are typically functional and represent cliches in inferencing.
- Ontologies are very compatible with the patterns literature.

The second thing I want to say is that, contrary to a widespread belief in the field, I believe that meta-knowledge complicates, not clarifies, KM.

Before discussing any supposed failings with knowledge maintenance, I should balance my remarks with some success stories. My own interest with knowledge engineering stems from a strong belief in the practicality of the technology. Many commercially useful systems have been developed including XCON [18], VT [16], SBF [17], and PIERS [22]. Successful systems must be maintained. XCON grew very large and turned from a knowledge engineering success story into a knowledge maintenance nightmare. However [2] found that if a rule editor was given meta-knowledge of inference cliches within XCON, then very large rules could be quickly built from very small specifications. More generally, given knowledge of the intended inferencing tasks, a KB editor can be productively constrained to only collect knowledge required for those tasks. This approach proved fruitful in many other applications (e.g. SBF,VT).

Nevertheless, the knowledge maintenance problem is far from solved. The XCON, SBF, and VT knowledge editors need meta-knowledge of inference cliches. Such meta-knowledge can be problematic (see below).

I begin my critique with a statement of something I no longer believe is true: “the explicit and high-level expression of knowledge in a KBS makes them easy to maintain”. There is just too much experiential evidence suggesting that even with high-level expressions, maintaining knowledge is hard. For example, half of XCON’s thousands of rules are changed every year [28]. To some extent, this might be due to its changing operational requirements (XCON configured

A. Originally

```
 RULE(22310.01) IF  (bhthy or
     utsh_bhft4 or
     vhthy) and not on_t4
           and not surgery
           and (antithyroid or
                hyperthyroid)
THEN DIAGNOSIS("...thyrotoxicosis")
```

B. Same rule, 3 years later

```
 RULE(22310.01) IF  ((((T3 is missing)
       or (T3 is low and
           T3_BORD is low))
       and TSH is missing
       and vhthy
       and not (query_t4 or on_t4 or
                 or surgery or tumour
                 or antithyroid
                 or hypothyroid
                 or hyperthyroid))
      or ((((utsh_bhft4 or
             (Hythe and T4 is missing
              and TSH is missing))
             and (antithyroid or
                 hyperthyroid))
           or  utsh_bhft4
           or  ((Hythe or borthy)
                and T3 is missing
                and (TSH is undetect
                     or TSH is low)))
           and not on_t4 and not
               (tumour or surgery)))
     and (TT4 isnt low or T4U isnt low)
THEN DIAGNOSIS("...thyrotoxicosis")
```

Figure 2. A rule maintained for 3 years.

computers for DEC and DEC keeps releasing new computers). However, even in supposedly stable domains, knowledge keeps being patched. Garvin ES-1 [7] offered interpretations of biochemical results. Over its lifetime, the biochemical assay hardware remained constant and, presumably, humans did not evolve significantly. Yet KB maintenance was on-going. The kind of changes seen within that KB are shown in Figure 2.

The change in KB size of Gavrin ES-1 is shown in Figure 3. Note that the rate of change within this system was linear; i.e. even in a stable domain, knowledge kept changing[1]. What could cause such a constant change? Well, expert systems store the views of experts and, experimentally, we know that experts disagree- even with themselves. Shaw used a terminology checking tool called repertory grids to compare the meaning of terms used by three geology experts on a common problem [27]. Two experiments were performed. In the calibrating experiment, experts reviewed their own knowledge, 12 weeks after they created it. This first experiment gives baseline expected agreement figures for a repertory grid analysis (see Figure 4). In the second experiment, inter-expert agreement was analyzed (see Figure 5). Note (e.g.) E_1,E_3: the results were much lower than in the calibration experiment suggesting that these experts held very different views about a supposedly standard problem in their field.

Worse than experts disagreeing is when experts agree, and they are wrong. The widespread reuse of incorrect knowledge adds significantly to the maintenance problem. We already have one example from the KE literature in which many people working over many months built models that contained bugs. In the Sisyphus-II experiments, various research groups re-implemented part of the VT elevator configuration system [15]. All the groups implemented a PSM with the same error [32]. The Sisyphus-II propose-and-revise PSM was a local greedy search. Local hill-climbing may ignore solutions which are initially unpromising, but lead later on to better solutions. In one experiment, this local greedy search algorithm failed to configure 13/25 elevator configurations: see Figure 6. What is disturbing about this result was that only one of the Sisyphus-II groups reported this error. Apparently, the rest trusted their reusable PSM so much, that they did not perform detailed validation studies.

[1]Technically, the Garvin ES-1 size changes are also consistent with a logarithmic curve; i.e. maintenance effort should decrease to zero over time. However, a visual inspection of the plot strongly suggests a linear fit.

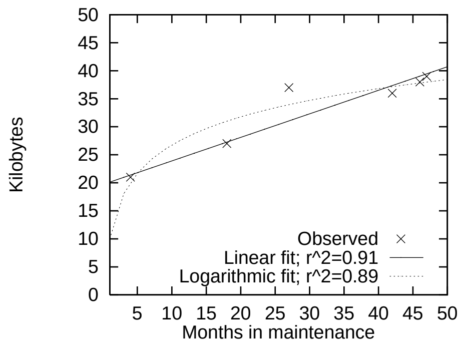

Figure 3. KB changes in Garvin ES-1.

Expert	%understands	%agrees
E_1	62.5	81.2
E_2	77.8	94.4
E_3	85.7	78.6

Figure 4. Self-agreement, 12 weeks later

Expertpairs	%understands	%agrees
E_1, E_2	62.5	33.3
E_2,E_1	61.1	26.7
E_1,E_3	31.2	8.3
E_3,E_1	42.9	33.3
E_2,E_3	44.4	20.0
E_3,E_2	71.4	33.3

Figure 5. Inter-expert agreement

KM is here to stay, I argue, since human knowledge rarely stabilizes. New experience always gives new insights which significantly change old knowledge. As evidence for this, consider the following case study from the machine learning community. Catlett [3] used C4.5 [23] to learn decision trees for 20 problems using either all the N=3000..5000 training cases or half the cases (randomly selected). The change in tree size and error rates are shown in Figure 7. In all but 1 case (demon, first row), more experience meant significantly less errors, but larger theories. This result is consistent with the *situated cognition* phenomena which, crudely speaking, can be summarized as

	ft/min				
Capacity (lbs)	200	250	300	350	400
2000	√	√	×	√	√
2500	×	×	√	√	√
3000	×	√	√	√	√
3500	√	×	×	×	×
4000	×	×	×	×	×

Figure 6. PSMs succeeding (√) or failing (×) to configure an elevator.

domain	Δ tree size	Δ error
demon	0.97	0.51
wave	1.91	0.95
diff	1.46	0.69
othello	1.68	0.8
heart	1.61	0.65
sleep	1.73	0.91
hyper	1.74	0.83
hypo	1.45	0.85
binding	1.51	0.82
replace	1.38	0.8
euthy	1.33	0.61
mean	1.52	0.77

Figure 7. Impact of more examples.

follows: using knowledge changes knowledge [21]. For example, Shalin et.al. [25] tried to find “accepted practice”; i.e. reused knowledge within expert communities. They found that experts do modify their behaviour according to community standards of “accepted practice”. However, it is only novices who slavishly re-apply that accepted practice. Experts adapt accepted practice when they apply it. That is, experts:

- Partially match current problem to libraries of accepted practice.
- Implement an acceptance test for their adaptation.
- Modify the accepted practice library if acceptance failure.

This lack of stability in human knowledge is not good news for fans of supposedly reusable meta-knowledge. Three studies suggest that it is at least an open question if adapting abstracted forms of previous knowledge (e.g. architectures, patterns, PSMs, ontologies) are a productivity tool in new situations. In the first study, international KA experts used some background knowledge to guide their analysis of a transcript of a patient talking to a doctor [9]. One group used an abstract model of diagnosis matured over many years; another used an abstract model invented very quickly (the “straw man”); and the rest used no model at all. The results are shown in Figure 8. The “mature model” group performed as well as the “straw man” group.

Reuse Model	% disorders identified	% knowledge fragments identified
Straw man: invented very quickly	50	28
Mature model: decades of work	55	34
No model	75	41

Figure 8. Productivity using different models.

Further, the “no model” group out-performed the groups using the models!

The second study [20] describes eight different supposedly reusable models of diagnosis (four from the PSM community, four from elsewhere). While some of these views on diagnosis share some common features, they reflect fundamentally divergent views on how to perform diagnosis. I therefore believe that, at least in the case of diagnosis, the consensus view has yet to stabilized and may not do so in the near future. More generally, I’m not sure that a consensus view on any of the PSMs has been reached, despite decades of research. There are significant differences between the list of PSM primitives offered by Clancey [4], KADS [30], and SBF. Also, the number and nature of the inference knowledge is not fixed. Often when a domain is analysed, a new PSM is induced [14].

The third study I want to mention documents the extent to which an ontology *supported* application development within DARPA’s High Performance Knowledge Based Systems (HPKB) initiative [5]. *Support* was measured in terms of the words that appeared in some new application: if ⅔ of those words came from an ontology, then that ontology offered a 67% support for that application. Two teams were involved: one at SRI and one from Teknowledge who used the Cycorp knowledge base (hereafter CYC/Tek). The teams built applications using an upper ontology (UO) released by Cycorp. Along with the UO, CYC/Tek and SRI made their own local extensions. Both teams built and debugged their ontology using a set of sample questions (SQ) issued by the HPKB evaluation team. At a pre-announced date, 110 test questions (TQA) were issued and the applications were scored. After a brief respite, a scope change was announced, followed (several days later) by test questions for the new scope (TQC). The SRI system analyzed by Cohen *et.al* could only handle 40 of the 110 questions so the CYC/Tek results are divided into CYC/Tek(110) and CYC/Tek(40) where the latter is the subset of the CYC/Tek system relevant to the questions that SRI could handle. The results are shown in Figure 9. The main results are: the local ontological extensions supported new applications 3-4 times more than the UO terms; as the scope change (TQA-TQC) the UO offered less and less support; and CYC/Tek’s reuse of the UO was greater than that of SRI. These results

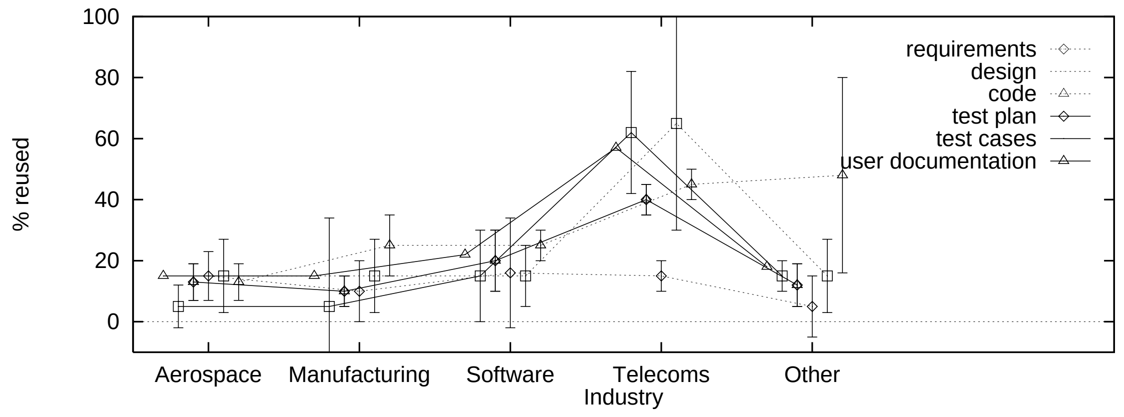

Figure 10. 95% confidence intervals for reuse levels in different industries.

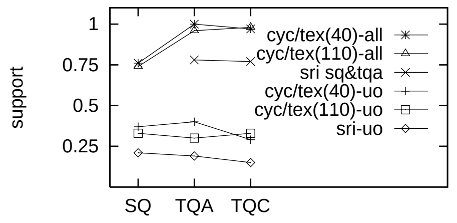

Figure 9. Resue of ontologies

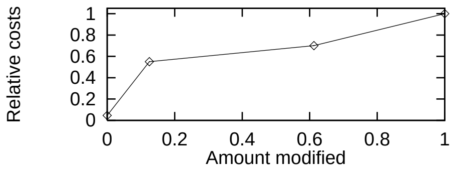

Figure 11. COCOMO-II, the cost of reuse with X% changes

suggested that the recent words you added yourself to an ontology offer more support than words added previously by other authors. This reflects somewhat badly on the ontology construction exercise since the CYC ontologies represent the planet's most serious effort to produce truly reusable ontologies.

There are at least two counter arguments to the above pessimism about reuse. Firstly, my arguments must be flawed since reuse already is widespread in the software engineering field. I would deny the premise of that argument: reuse levels in software engineering are low and seem to be uncorrelated to technologies such as reusable components. Frakes & Fox surveyed 100s of European and North American IT professionals to conclude that reuse levels were low (20% or less). Further, it was not correlated to technology (e.g. use of COBOL, C++, case tools, reuse libraries...) [11]. Reuse seemed to be correlated to non-software technology issues; e.g. hardware standards enabled high levels of reuse in the telecommunications industry (see Figure 10).

A second counter argument to the above reuse-pessimism is as follows. While we may use little of an ontology or a PSM, it may still be useful as a "pointer tool". That is, the ontology/PSM could be used as a structuring tool for exploring a new domain. Roughly speaking, reusing abstracted forms of old knowledge is pointing the way saying "these kinds of things are important, even if these particular things are not". In this approach, developers kick-start the development with an ontology/PSM. However, developers may extensively modify (or even discard) the ontology as their understanding of a domain increases. Further experimentation is required to demonstrate the value of this method. In the meantime, we can offer one caution. The COCOMO-II software cost estimation model offers an estimate of the cost of adapting reusable sub-routines for a new project [1, p21]. A learning curve must be traversed before a module can be adapted. By the time you know enough to change a little of that module, you may as well have re-written 60% of it from scratch; see Figure 11. That is, if you plan to use meta-knowledge (e.g. inference patterns or ontologies) as a "pointer tool", then if the re-structuring exceeds some small amount, you have lost much of the economic benefit of that meta-knowledge.

Having rejected meta-knowledge as the basis for successful KM, I should now describe alternative KM strategies. Space restrictions prevent a detailed account. Instead, I direct the reader to the ripple-down-rules work of Compton, Richards, Preston, et.al. [6, 8, 22, 24] where patch histories to the knowledge base are maintained. Candidates for new patches are inferred by an analysis of the paths taken through the KB and the patches found on those paths. Patch histories are low-level syntactic knowledge yet capture the context of change of an expert system. Very large expert systems have been built and maintained in this manner, without needing knowledge engineers.

Acknowledgements

This work was partially supported by NASA through cooperative agreement #NCC 2-979.

References

[1] C. Abts, B. Clark, S. Devnani-Chulani, E. Horowitz, R. Madachy, D. Reifer, R. Selby, and B. Steece. Cocomo ii model definition manual. Technical report, Center for Software Engineering, USC,, 1998. `http://sunset.usc.edu/COCOMOII/cocomox.html#downloads`.

[2] J. Bachant and J. McDermott. R1 Revisited: Four Years in the Trenches. *AI Magazine*, pages 21–32, Fall 1984.

[3] J. Catlett. Inductive learning from subsets or disposal of excess training data considered harmful. In *Australian Workshop on Knowledge Acqusition for Knowledge-Based Systems, Pokolbin*, pages 53–67, 1991.

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

[5] P. Cohen, V. Chaudhri, A. Pease, and R. Schrag. Does prior knowledge facilitate the development of knowledge-based systems? In *AAAI'99*, 1999.

[6] P. Compton, G. Edwards, A. Srinivasan, P. Malor, P. Preston, B. Kang, and L. Lazarus. Ripple-down-rules: Turning knowledge acquisition into knowledge maintenance. *Artificial Intelligence in Medicine*, 4:47–59, 1992.

[7] P. Compton, K. Horn, J.R. Quinlan, and L. Lazarus. Maintaining an expert system. In J.R. Quinlan, editor, *Applications of Expert Systems*, pages 366–385. Addison Wesley, 1989.

[8] P.J. Compton and R. Jansen. A Philosophical Basis for Knowledge Acquisition. *Knowledge Acquisition*, 2:241–257, 1990.

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

[10] M. Fowler. *Analysis Patterns: Reusable Object Models*. Addison Wesley, 1997.

[11] W.B. Frakes and C.J. Fox. Sixteen questions about software reuse. *Communications of the ACM*, 38(6):75–87, June 1995.

[12] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1995.

[13] T.R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220, 1993.

[14] M. Linster and M. Musen. Use of KADS to Create a Conceptual Model of the ONCOCIN task. *Knowledge Acquisition*, 4:55–88, 1 1992.

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

[16] S. Marcus, J. Stout, and J. McDermott. VT: An Expert Elevator Designer That Uses Knowledge-Based Backtracking. *AI Magazine*, pages 41–58, Winter 1987.

[17] D. Marques, G. Dallemagne, G. Kliner, J. McDermott, and D. Tung. Easy Programming: Empowering People to Build Their own Applications. *IEEE Expert*, pages 16–29, June 1992.

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

[19] T. Menzies. Knowledge maintenance: The state of the art. In *The Knowledge Engineering Review*, number 1, pages 1–46, 1999. Available from `http://www.cse.unsw.EDU.AU/~timm/pub/docs/97kmall.ps.gz`.

[20] T.J. Menzies. OO patterns: Lessons from expert systems. *Software Practice & Experience*, 1998. In press. Available from `http://www.cse.unsw.edu.au/~timm/pub/docs/97probspatt.ps.gz`.

[21] T.J. Menzies and B. Clancey. Editorial, special issue on situated cognition, international journal of human-computer studies, 1998.

[22] P. Preston, G. Edwards, and P. Compton. A 1600 Rule Expert System Without Knowledge Engineers. In J. Leibowitz, editor, *Second World Congress on Expert Systems*, 1993.

[23] J.R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[24] D. Richards and P. Compton. Combining formal concept analysis and ripple down rules to support the reuse of knowledge. In *SEKE '97: Proceedings of 1997 Conf. on Software Eng. & Knowledge Eng, Madrid*, 1997.

[25] V.L. Shalin, N.D. Geddes, D. Bertram, M.A. Szczepkowski, and D. Dubois. Expertise in dynamic, physical task domains. In P.J. Feltovich, K.M. Ford, and R.R. Hoffman, editors, *Expertise in Context*, chapter 9, pages 195–217. MIT PRess, 1997.

[26] M. Shaw and D. Garlan. *Software Architecture: Perspectives on an Emerging Discipline*. Prentice Hall, 1996.

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

[28] E. Soloway, J. Bachant, and K. Jensen. Assessing the maintainability of xcon-in-rime: Coping with the problems of a very large rule-base. In *AAAI '87*, pages 824–829, 1987.

[29] B.J. Wielinga, A.T. Schreiber, and J.A. Breuker. KADS: a Modeling Approach to Knowledge Engineering. *Knowledge Acquisition*, 4:1–162, 1 1992.

[30] B.J. Wielinga, A.T. Schreiber, and J.A. Breuker. KADS: a Modeling Approach to Knowledge Engineering. *Knowledge Acquisition*, 4:1–162, 1 1992.

[31] G.R. Yost. Acquiring knowledge in soar. *IEEE Expert*, pages 26–34, June 1993.

[32] Z. Zdrahal and E. Motta. 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*, 1996.