Notes for Chris Welty on Ontologies Cost-Benefits of Ontology. ? for inclusion to a journal submission on ontologies. WP:b/99/seke/abouton(August 7, 1999).

Cost-Benefits of Ontologies

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

What are the benefits and what are the costs of using ontologies? Using ontologies can have several benefits:

Interoperability: When interfacing two components, access the ontology of each component to design a mapping between different concepts in different components.

Browsing/searching: The meta-knowledge within an ontology can assist an intelligent search engine with processing your query. For example, if a query returns no results, then the ontology could be used to automatically generalize the query to find nearest partial matches.

Reuse: Why waste time and money rebuilding component X when X already exists in someone else's library?

Structuring: It may be faster to build new systems via "ontological bootstrapping"; i.e. use the conceptualizations in ontologies to assist you with structuring the knowledge in a new domain.

Certain results from the software engineering and knowledge engineering literature suggest that the benefits from reuse and structuring may come at a considerable cost. These results are presented below. To the best of our knowledge, there are no counter results doubting the utility of ontologies for interoperability and searching/browsing.

1 Reusing Ontologies

Given the state-of-the-art in ontological engineering, we still lack costing models for software projects that adopt reusable ontological libraries. However, we can find such costing models in the standard software engineering literature. The COCOMO-II software cost estimation model offers an estimate of the cost of adapting reusable sub-routines for a new project [Abts, Clark, Devnani-Chulani, Horowitz, Madachy, Reifer, Selby & Steece 1998, 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 1.

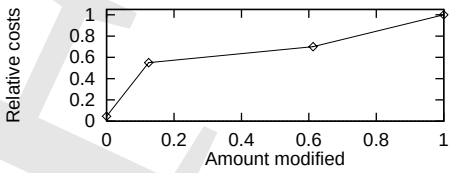

Figure 1. COCOMO-II, the cost of reuse with X% changes. The Y axis shows the costs of reuse divided by the cost of rebuilding from scratch.

The COCOMO-II results relate to the adaption cost of procedural systems. Hence, they may not apply to declarative descriptions of system terminology (i.e. an ontology). However, at the very least, these results caution us that just because we can access an ontology, this does not necessarily mean that we can use it as a tool to improve our productivity. Ontologies must be learnt prior to use and this learning time may have a non-trivial impact on the overall cost.

2 Ontology-Based Domain Structuring

2.1 Benefits of Structuring

Chris: this is a stand-alone section which you may wish to move out of this section into the general introduction of the article. However, these words will be required as a introduction to §2.2.

Before raising issues with the costs of ontological structuring, we pause to describe some of its benefits. Structuring via ontologies can be manual or automatic. As an example of manual structuring, suppose a requirements engineer sees "sale" as part of the requirements document. If she was an experienced analyst in this domain, she might then apply her background knowledge of this domain to check the current specification. For example, a general ontology for financial transaction (Figure 2) includes a "subsequent transaction" term. This alerts our requirements engineer to

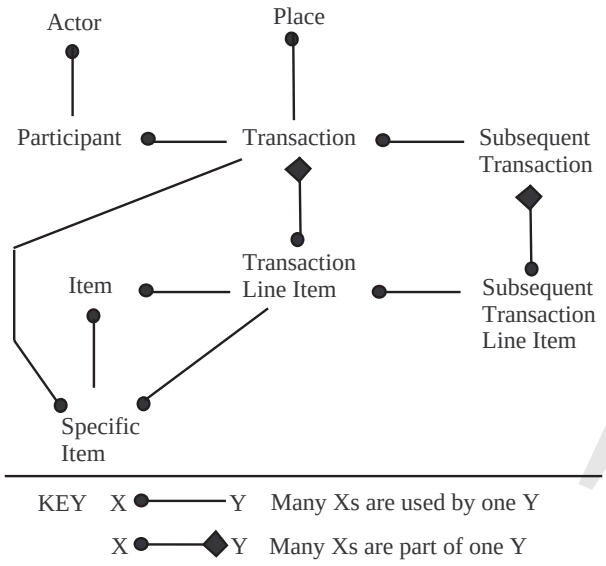

Figure 2. Part of Coad et al.'s [1997] definition of a financial transaction.

ask questions such as "are the sold items ever returned to the store?". That is, manual ontological structuring can assist in auditing the current version of a system description.

Three examples of automatic ontological structuring are:

RIME: RIME was an intelligent editor for simplifying the maintenance of DEC's XCON automatic computer configuration system. Bachant & McDermott [1984] found that if a rule editor was given meta-knowledge of the ontology of the problem solving methods within XCON, then very large rules could be quickly built from very small specifications. Similar results were seen in the SALT and SBF systems.

SALT: Marcus & McDermott's [1989] SALT system restricted its knowledge editors to only those terms relevant for the propose-and-revise problem solving used in the VT elevator configuration system [Marcus, Stout & McDermott 1987]. $^{2130}/_{3062} \approx 70\%$ of VT's rules could be auto-generated by SALT.

SBF: In Marques, Dallemagne, Kliner, McDermott & Tung's [1992] SBF study, nine applications of intelligent computer hardware configuration were built with and without the SBF toolkit. SBF auto-configured an expert system case tool via an automatic exploration of a library of problem solving methods. The ontological commitments of each method were then translated into data collection screens. Development times dropped from 63 to 250 days (without SBF) to 1 to 17 days (with SBF).

The general conclusion from all this work is that, given knowledge of the ontology of the intended inferencing tasks, a KB editor can be productively constrained to only collect knowledge required for those tasks. This is a clear benefit of ontologies.

2.2 Costs of Structuring

The benefits of ontology-based structuring come at a cost. Recall that systems like RIME, SBF, and SALT achieved productivity benefits via editors specialized for the ontology. This process of specialization can be expensive. While SALT's was successful in assisting users acquiring their knowledge, the ontological commitments of that environment were very hard to change. Hence, that editor could not be easily adapted to other expert systems. Some general principles exist for such customization (e.g. [Gil & Tallis 1997]) but is an open issue if the customization costs are outweighed by the productivity benefits. Swartout & Gill [1996] offer a general discussion of this area and a succinct comparison of different research directions. Note that ontology-based structuring tools must be customized in two ways. Firstly, as discussed above, they must be customized for any new ontologies. Secondly, they must be re-customized if the ontology changes. If ontologies are unstable (prone to large change over time), then we would doubt the overall benefits of automatic structuring tools based on ontologies.

Such instability would mean that along with the cost of constructing an ontology, there are two other costs:

- Ontology maintenance
- Maintenance of the software systems built using the (now changed) ontology.

How common is ontological instability? We don't know since we have very little experience with the long-term use of large ontologies. However, many researchers argue that in many cases, experts disagree about even well-established features of their domain; e.g. [Finkelstein, Gabbay, Hunter, Kramer & Nuseibeh 1994, Menzies, Easterbrook, Nuseibeh & Waugh 1999, Gaines & Shaw 1989]. If this is a widely applicable result, then we should routinely expect ontological instability. For example, Shaw [1988] used a terminology checking tool called repertory grids to compare the meaning of terms used by three geology experts on a common problem [Shaw 1988] (a sample repertory grid is shown in Figure 3). 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,

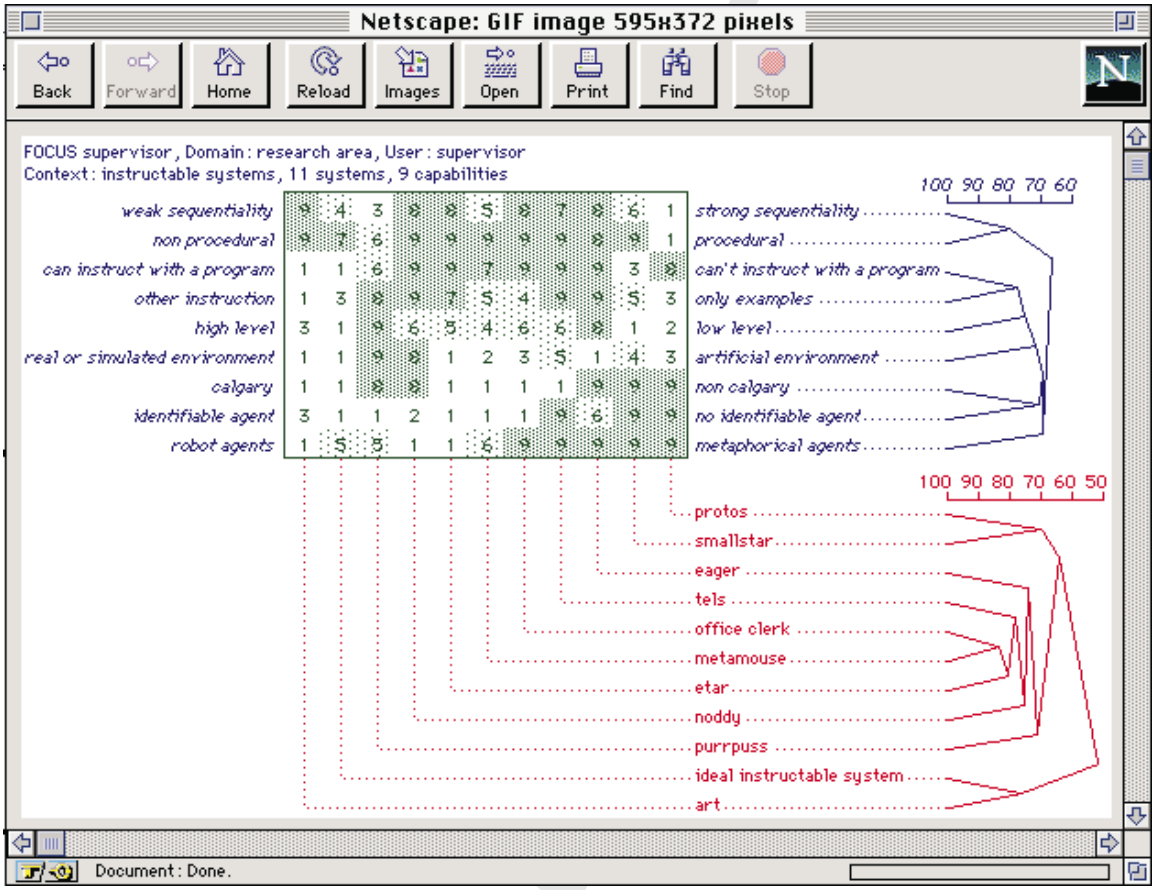

Figure 3. A web-based repertory grid tool. Dimensions are the rows in the table. Examples are shown bottom right. Example rankings are shown inside the table. these rankings are used to generate distances between dimensions and examples. For example, the dimensions "strong sequential to weak sequential" is very close to the dimension "procedural to non-procedural".

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

inter-expert agreement was analyzed (see Figure 5). Note (e.g.) E_1,E_3 results (line 4): expert E_1's terminology was compatible with the terminology of expert E_3 8.3% of the time (!!). These figures are much lower than in the calibration experiment suggesting that these experts held very different views about a supposedly standard terminology in their field.

2.2.1 Other Examples of Knowledge Instability

Chris: This is another stand-alone section could be culled without damage to my overall argument. However, if it interests you, I can easily double the size of this section.

We should expect ontological instability in domains where experts do not agree on widely used terms from that domain, or in domains where terms are evolving. There is

domain	Change in tree size	Change in 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 6. Impact of more examples.

some evidence to suggest that we should routinely expect such evolution. Agnew, Ford & Hayes [Agnew, Ford & Hayes 1993] comment that "expert-knowledge is comprised of context-dependent, personally constructed, highly functional but fallible abstractions". That is, we may always be patching up our expert knowledge. Maintaining an expert system can be a perpetual process [Compton, Horn, Quinlan & Lazarus 1989, Soloway, Bachant & Jensen 1987]. Knowledge bases may evolve for two reasons:

- To remove bugs. Systematic biases in expert preferences may result in incorrect/incomplete knowledge bases [Silverman 1992]. Working systems often contain multiple undetected errors [Preece & Shinghal 1992].
- To add enhancements. Experiments from machine learning suggest that there may be no end to the useful enhancements that can be made to a knowledge base. Catlett [1991] used a machine learner [Quinlan 1986] to automatically build 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 6. In all but 1 case (demon, first row), more experience meant significantly less errors, but larger theories.

Catlett's machine learning experiments are consistent with the *situated cognition* phenomena which, crudely speaking, can be summarized as follows: using knowledge changes knowledge [Menzies & Clancey 1998]. For example, Shalin, Geddes, Bertram, Szczepkowski & Dubois [1997] tried to find "accepted practice"; i.e. reused knowledge within expert communities (pilots, doctors, and army personnel performing troop maneuvers). 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.

In summary, experts often want to change knowledge after they have created some explicit record of it. Hence, as argued above, an on-going cost with ontologies maybe ontological maintenance and the revision of devices created using that ontology.

3 Discussion

The science of ontologies is too young to include detailed cost-benefit analyzes. In theory, using ontologies can have many benefits. However, certain software engineering and knowledge engineering results suggest that reusing old ideas like ontologies incurs several extra costs. Evaluation programs for ontologies should include checks that these costs do not out-weight the benefits of using ontologies.

Acknowledgements

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

References

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

Agnew, N., Ford, K. & Hayes, P. [1993], 'Expertise in context: Personally constructed, socially elected, and reality-relevant?', *International Journal of Expert Systems* **7**.

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

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

Coad, P., North, D. & Mayfield, M. [1997], *Object Models: Strategies, Patterns, and Applications*, Prentice Hall.

Compton, P., Horn, K., Quinlan, J. & Lazarus, L. [1989], Maintaining an expert system, *in* J. Quinlan, ed., 'Applications of Expert Systems', Addison Wesley, pp. 366–385.

Finkelstein, A., Gabbay, D., Hunter, A., Kramer, J. & Nuseibeh, B. [1994], 'Inconsistency handling in multi-perspective specification', *IEEE Transactions on Software Engineering* **20**(8), 569–578.

Gaines, B. & Shaw, M. [1989], Comparing the conceptual systems of experts, *in* 'IJCAI '89', pp. 633–638.

Gil, Y. & Tallis, M. [1997], A script-based approach to modifying knowledge bases, *in* 'Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI-97)'.

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

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

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

Menzies, T. & Clancey, B. [1998], Editorial, special issue on situated cognition, international journal of human-computer studies, Vol. 49.

Menzies, T., Easterbrook, S., Nuseibeh, B. & Waugh, S. [1999], An empirical investigation of multiple viewpoint reasoning in requirements engineering, *in* 'RE '99'. Available from `http://research.ivv.nasa.gov/docs/techreports/1999/NASA-IVV-99-009.pdf`.

Preece, A. & Shinghal, R. [1992], Verifying knowledge bases by anomaly detection: An experience report, *in* 'ECAI '92'.

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

Shalin, V., Geddes, N., Bertram, D., Szczepkowski, M. & Dubois, D. [1997], Expertise in dynamic, physical task domains, *in* P. Feltovich, K. Ford & R. Hoffman, eds, 'Expertise in Context', MIT PRess, chapter 9, pp. 195–217.

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.

Silverman, B. [1992], 'Survey of expert critiquing systems: Practical and theoretical frontiers', *Communications of the ACM* **35**, 106–127.

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', pp. 824–829.

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'.