Knowledge Maintenance: The State of the Art

Tim Menzies

NASA/WVU Software Research Lab,
100 University Drive, Fairmont, USA
<tim@menzies.com>*

June 1, 1999

*WP:b/9x/kmall/tutorial.tex

What is KM?

Knowledge maintenance (KM) of knowledge-based systems =

- Insight + reflect + change + preserve

Insight = seeing bug or enhancements

Reflect = searching the dependencies between concepts

Change = the easiest thing to do badly, the hardest thing to do correctly.

Preserve = changes that address new insights do not introduce bugs into older concepts.

What is not KM?

An ability to rapidly build or change a system:

- The “rapid acquire assumption”.
- Insufficient unless “build” or “change” supports “preserve”.

An ability to watch knowledge execute:

- The “operationalisation KM assumption”.
- Insufficient without “behavioural knowledge” (see below)

An ability to check a KB using meta-knowledge:

- The “recursive maintenance problem”).
- Insufficient unless the meta-knowledge can be maintained as well.

Tutorial: summary #1

- Survey of KM techniques
- Challenge issues for KM research (see page 2).
- A call-to-arms: certain incorrect assumptions in modern KM research.

Tutorial: summary #2

Trite sound bites summarizing my views:

- Knowledge does not stabilise.
- Experts don't agree.
- Modeling is ok, but models aren't.
  - Useful: building models
  - Useless: carving them in stone
- Reuse:
  - Likely: reuse=productivity=myth.
  - Unlikely: me reuse your ideas.
- Maintain the process, not the product.

A Quick KM Primer

KM state-of-the-art [Menzies, 1999] (see this reference for more details on the material in this tutorial).

KM tools supporting preserve:

- Logic-based: [Debenham, 1998,Compton and Jansen, 1990]
- Network-based: [Menzies, 1996,Menzies and Compton, 1997]
- Procedures-based: [Gil and Tallis, 1997, Tallis, 1998]

Meta-knowledge to support KM: [Bachant and McDermott, 1984, Marcus et al., 1987]

KM without meta-knowledge: [Richards and Compton, 1997]

Who needs KM? Just rebuild! [Marques et al., 1992]

General texts:

- Empirical artificial intelligence [Cohen, 1995]
- Advice from SE [Fenton and Pfleeger, 1997]

Preliminaries

Tutorial: Objective

Many factors have combined to reduce our belief that we can know the "truth" (in some absolute sense) about our world. If we doubt that our initial specification of a system will be incomplete, and routinely expect those systems to change significantly over their life time, then we must move the focus of knowledge engineering and software engineering from acquisition and analysis to maintenance.

To facilitate this change, we offer here a review of the state-of-the-art in the emerging field of knowledge maintenance (KM). Techniques from many different communities (e.g. software engineering, requirements modeling, the verification & validation community, case-based reasoning, machine learning, object-oriented databases) will be shown to all contribute to addressing the KM problem.

Tutorial: Brief Description

In the software and knowledge engineering literature we can see maintenance strategies offered to maintain seven main types of knowledge and five main ways these are processed.

This tutorial will review dozen of software systems that contribute to these 7*5=35 types of knowledge maintenance to make the following conclusions:

- Open issues with the current maintenance research are identified. These include (a) areas that are not being addressed by any researcher; (b) the “recursive maintenance problem”; and (c) drawbacks with “rapid acquire systems” and the “operationalisation KM assumption”.
- A process is described for commissioning a new maintenance tool.
- A general common principle for maintenance (search-space reflection) is isolated.

Tutorial: Audience

Practicing software engineers interested in using or assessing leading edge technologies; software managers reviewing the state of the art in their field; software engineering and knowledge engineering researchers; graduate students preparing literature reviews of their field.

About the Author

Dr. Tim Menzies holds a Ph.D. in artificial intelligence from the University of New South Wales, Australia, and works at the NASA/WVU Software Independent Verification and Validation Facility, Fairmont West Virginia. In that position, he explores logical methods for improving our ability to reason about the products of software engineering (specs, code, design documents).

At the time of this writing, Dr. Menzies has 67 publications, dozens of which are in international refereed forums*. Much of this work is an analysis of where our current generation of software engineering and knowledge engineering techniques stop working.

He is also an active figure in the knowledge acquisition (KA) community and is one of the co-chairs of the evaluation of knowledge engineering methodologies track at the annual international KA workshop

*See `http://www.cse.unsw.edu.au/~timm/pub/docs`.

Expectation Management (the FAQ)

“You skipped lots of slides.” Can’t cover all this material in 3 hours.

“Some material was rushed” or “I wanted more details on XYZ” This is an overview only:

- Every page is someone’s research life- can’t show it all.
- If we pique(*) your interest in XYZ, and you know XYZ is connected to ABC, we have succeeded.
- For more details, see [Menzies, 1999].

(*)Pique \Pique\,v. t. To excite to action by causing resentment or jealousy; to stimulate; to prick; as, to pique ambition, or curiosity.

Biases of the Author

The majority Euro-view of KA/KM:

- Understand and adapt KBs via meta-knowledge of cliched terms and inference procedures (a.k.a. ontologies and problem solving methods).
- e.g. KADS [Wielinga et al., 1992a]

The minority Australian-view of KA/KM (includes author):

- Meta-knowledge only delays the KM problem (how do you maintain the meta-K?).
- Alternative: maintain via syntactic structures such as context (RDR) or dependency graphs (knowledge normalization).
- e.g. RDR, p60 [Preston et al., 1993,Richards and Compton, 1997, Menzies, 1998];
- e.g. knowledge normalization,p 51 [Debenham, 1998]:

Motivation

True or False?

The explicit and high-level expression of knowledge in a KBS makes them easy to maintain.

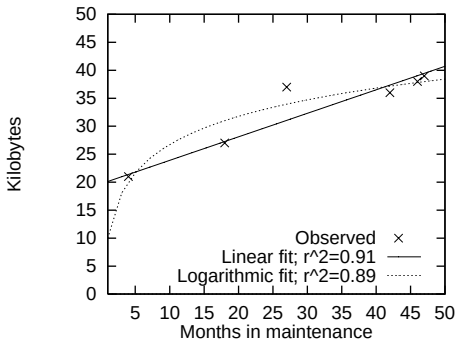
False!

XCON:

- Half of its thousands of rules changed every year [Soloway et al., 1987].
- ?? due to changing environment (XCON configured computers for DEC computers and DEC keeps releasing new computers).

Garvin ES-1:

- KBS change occurs in even static domains [Compton et al., 1989]



Looks linear; i.e. change will be forever; i.e. maintenance =forever.

A Garvin ES-1 Rule (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")
```


Three 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")
```

[Compton et al., 1989]

Experts Disagreeing

[Shaw, 1988]: used repertory grids to compare the meaning of terms used by different experts on a common geology problem.

A. The calibrating experiment: 12 weeks later, self-review. These figures given baseline expected agreement figures for this instrument.

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

B. The real experiment. Note E_1,E_3: very low levels of agreement.

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

Does Knowledge Stabilize?

[Catlett, 1991]: use C4.5 [Quinlan, 1986] to learn decision trees for 20 problems using either:

- all the N=3000..5000 training cases or
- half the cases (randomly selected)

In all but 1 case (demon, first row), more experience meant significantly less errors, but larger theories,

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

Does Knowledge Stabilize? (2)

Situated cognition:

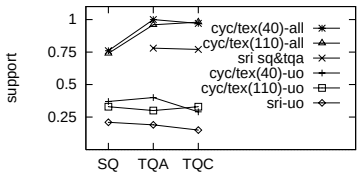
- Using knowledge changes knowledge [Menzies and Clancey, 1998].

For example, [Shalin et al., 1997]:

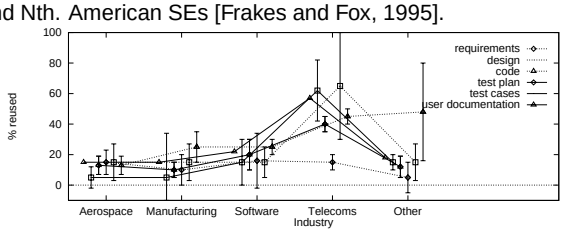
- Experts do modify their behaviour according to community standards of “accepted practice”.
- But its only novices who slavishly re-apply that accepted practice.
- Experts adapt accepted practice when they apply it:
  - Partially match current problem to libraries of accepted practice.
  - Implement an acceptance test for their adaptation.
  - Modify the accepted practice library if acceptance failure.

Reuse: The Reality

Use of “reusable” terms (the upper ontology or UO) stable or decreased as spec changed(SQ→TQA→TQC). Also, groups used their own local and recent extensions terms far more (the ALL terms). Data from 2 groups (SRI or Cyccorp/Tecknowledge) in the HPKB project [Cohen et al., 1999].



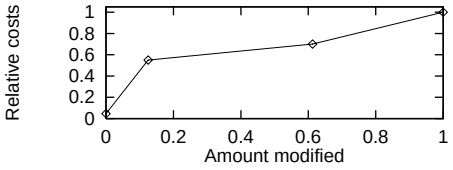
Reuse levels in SE constant and low (exception: telecommunications industry). Reuse not correlated to technology (COBOL to C++, case tools,...). Data from paper survey of 100s of European and Nth. American SEs [Frakes and Fox, 1995].



Reuse & Productivity

Q: Is old knowledge (that we can adapt) still a productivity tool for new apps?

A1: YES if adaptations small ($< 12.5\%$). Otherwise, overall cost benefits not impressive. Data from 2954 NASA SE modules [Abts et al., 1998, p21].



A2: Maybe not. Decades-old diagnosis models give same production benefits as models invented very quickly. And no model gives higher production benefits. Data from KA experts reading patient-doctor transcripts [Corbridge et al., 1995]

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

So, What Role for Models?

My argument is:

- Building new models is useful:
  - Unites a community.
  - Reflection over models let us plan for events we have not/cannot directly experience.
- Holding on to old models may not be useful:
  - While I may reuse some of my old ideas;
  - I doubt that I will reuse yours we belong to the same clan.

Summary

KM is here to stay.

Not because of:

- Sloppy analysis
- Lack of formal rigor (e.g. formal methods)
- Poor management practices
- The wrong tools
- etc etc

But because of:

- The fundamental nature of expert knowledge.
- Using “it” changes “it”.
- Hitting the nail changes your grip on the hammer.
- Exercising knowledge makes you refine that knowledge

Pressing and urgent issue: how to improve KM?

7 Kinds of Knowledge

The KM Space

In the literature, I can see 7 kinds of knowledge being maintained using 5 activities [Menzies, 1999].

Ktype: Knowledge type	Ptype:Knowledge Processing Activity				
	Acquire	Operation-alise	Fault	Fix	Preserve
WordK	all	most	few	few	few
SentenceK (ontologies)	most	most	many	many	few
BehaviouralK	some	few	few	few	none
PSMs	some	many	few	few	few
QualityK	few	few	none	none	none
FixK	few	many	none	none	none
SocialK	few	none	none	none	none

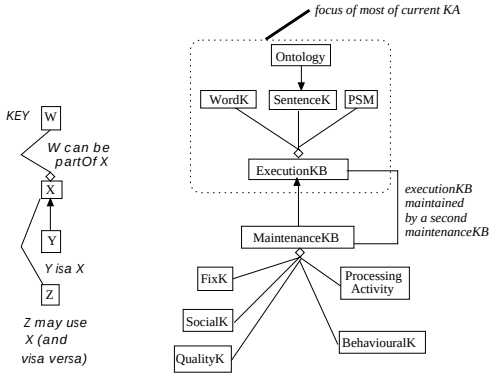
Comments:

- KM needs knowledge acquisition (KA). To maintain “it”, first you have to get “it”. Next you might have to update “it”.
- Not all parts of the 7*5 points in the KM space are covered.
- Large space to explore. High points only at this tutorial. See [Menzies, 1999] for more details.
- Must explain the Ktypes before the Ptypes.

7 Knowledge Types

Current focus: *execution KBs* which acquire and operationalise *wordK* (e.g. atomic terms) and *sentenceK* (e.g. rules and ontologies) using problem solving methods (PSMs)

Emerging focus: *maintenance KBs* which contain: *behaviouralK* storing the known or desired behaviour of the KB; *socialK* representing the social context; *fixK* representing KB repair strategies; *qualityK* representing how the quality of the KB will be assessed.



K-type: WordK

WordK: the things we can't divide any further. E.g.

- Logic programming: wordK=propositions
- OO high-level design wordK=method names
- Functional systems: word=simple functions (e.g. a comparison `age(patient)=old`).
- Rule-based systems: wordK=atoms; e.g.:

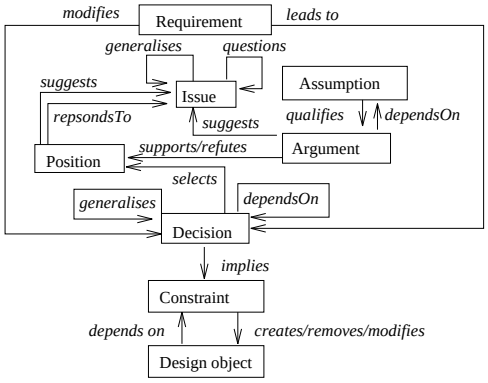
if infection is meningitis *and*
infection is bacterial *and*
patient has undergone surgery *and*
the patient has had neurosurgery *and*
neurosurgery-time was $<$ 2 months ago *and*
patient got a ventricular-urethral-shunt
then infection = eColi (.8) *or klebsiella (.75)*

[Clancey, 1992]

K-type: SentenceK

The rule on page 28 is sentenceK connecting that connect wordKs.

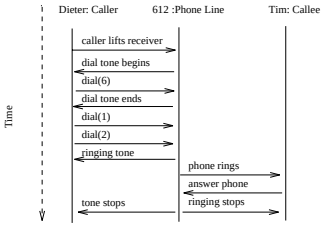
Meta-knowledge of legal sentence types=ontology:"an explicit specification of a conceptualization" [Gruber, 1993]. E.g. here is an ontology of design discussions [Ramesh and Dhar, 1992].



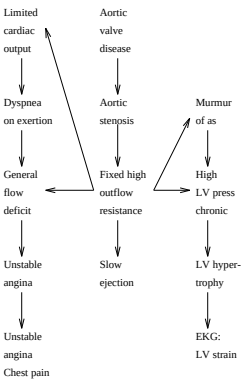
In an ontology, abstract terms often appear high in some *isa* hierarchy while specific domain terms appear lower down the hierarchy; e.g. recall the 7 types of knowledge on page 27.

K-type: BehaviouralK

Knowledge of known or desired or past behaviour.



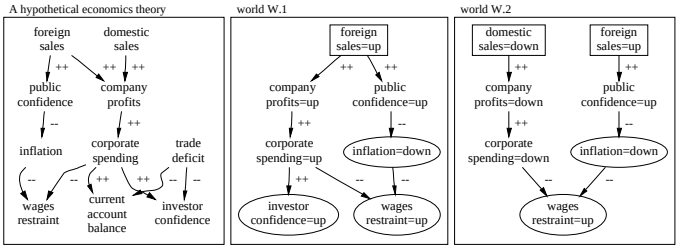
BehaviouralK in OO analysis [Booch et al., 1997]



BehaviouralK in case-based reasoning [Kolodner, 1991, Kolodner, 1993]: new situations are managed by retrieving and modifying previously see cases.

K-type: BehaviouralK & HT4 (2)

[Menzies and Compton, 1997]



Ins	foreignSales=up, domesticSales=down
goals G	investorConfidence=up, inflation=down, wageRestraint=up
P.1	foreignSales=up, ?companyProfits=up, ?corporateSpending=up, investorConfidence=up.
P.2	domesticSales=down, ?companyProfits=down, ?corporateSpending=down, wageRestraint=up.
⋮	⋮

?X= assumptions; by the way: NP-hard

Can find interesting errors in published theories of neuroendocrinology that were invisible to journal reviewers and paper authors and mathematical analysis.

K-type: BehaviouralK (3)

Coverage tools = SE behaviouralK

- All branches coverage
- All usages coverage
- All decision points coverage
- All ... coverage

Desired behaviour specified as global constraints:

- Succinct expression
- Can be checked directly using model-checkers [Clarke et al., 1986]
- Can be checked via theorem-provers: ground, then negate constraints. Error detected if the negated constraints are reachable,

Operationalisation KM Assumption

"If we can watch a program execute, then we can understand it."

Definitions:

- Test cases : $< input, output >$
- Anomaly: $expected(output) \neq actual(output)$
- Mere program running: $< input, expected(output) = \emptyset >$, i.e. no behaviouralK.

Output can be voluminous, too slow to process (HT4=NP hard): need heuristic agents to find significant outputs

- Compare with desired (HT4- page 31)
- Code coverage
- Model checkers

DON'T ask the developer to evaluate via program watching:

- "Halo effect"
- "What we need is not opinions or impressions, but relatively objective measures of performance." [Cohen, 1995, p74].

Ktype: PSMs: Problem Solving Methods

[Clancey, 1992]. Recall this rule:

if infection is meningitis *and*
infection is bacterial *and*
patient has undergone surgery *and*
the patient has had neurosurgery *and*
neurosurgery-time was $<$ 2 months ago *and*
patient got a ventricular-urethral-shunt
then infection = eColi (.8) *or klebsiella (.75)*

This rule blurs the true heuristic ...

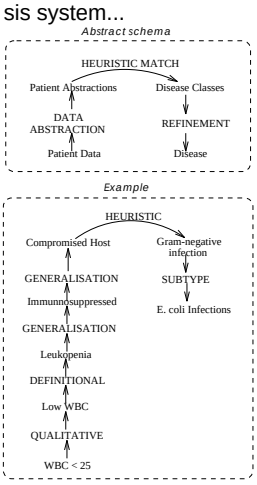
if patient got a ventricular-urethral-shunt
then infection = e.coli (.8) *or klebsiella (.75)*

... with more general problem solving method knowledge:

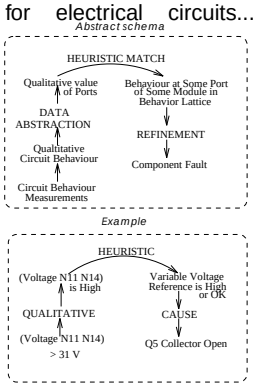
Strategy	*Description*
exploreAndRefine	Explore super-types before sub-types.
findOut	If an hypothesis is subsumed by other findings which are not present in this case then that hypothesis is wrong.
testHypothesis	Test causal connections before mere circumstantial evidence.

Ktype: PSMs (2)

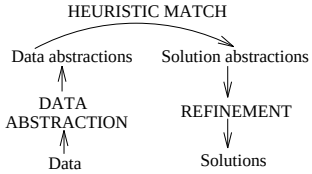
Inside a medical diagnosis system...



...and a fault system for electrical circuits...



... is the same inference cliche: heuristic classification [Clancey, 1985].



Ktype -PSMs (3)

Solo PSM:

- Controlled traversal through nested "operator spaces" (SOAR-PSCM [Newell, 1993, Yost and Newell, 1989, Yost, 1993]); customizable abductive choice operators [Menzies and Mahidadia, 1997].

Multiple PSMs:

- SPARK/BURN/FIREFIGHTER (SBF) [Marques et al., 1992]; generic tasks [Chandrasekaran et al., 1992]; configurable role-limiting methods [Swartout and Gill, 1996,Gil and Melz, 1996]; model construction operators [Clancey, 1992]; CommonKADS [Wielinga et al., 1992a, Schreiber et al., 1994]; the Method-To-Task approach [Eriksson et al., 1995]; components of expertise [Steels, 1990]; MIKE [Angele et al., 1996]); TINA [Benjamins, 1995]. Libraries of PSMs are described in [Benjamins, 1995, Breuker and de Velde (eds), 1994,Chandrasekaran et al., 1992,Motta and Zdrahal, 1996, Tansley and Hayball, 1993].

- PSM approaches contrasted in the *Related Work* section of [Wielinga et al., 1992a]

Ktype: PSMs (4): success stories

SALT KB editor for VT (elevator configuration):

- PSM meta-knowledge restricted the dialogues to only those terms relevant for the propose-and-revise PSM used in VT [Marcus and McDermott, 1989].

- $(^{2130}/_{3062} \approx 70\%)$ of VT's rules were auto-generated by SALT

SPARK/BURN/FIREFIGHTER [Marques et al., 1992]:

- 9 applications (hardware configuration).

- Intelligent editor to explore distinguishing features between PSM.

- Auto-configuration of executable from PSM library.

- Development times = one to 17 days (using SBF)

- Development times = to 63 to 250 days (without using SBF)

- High-water mark in reported productivity increases in software or knowledge engineering.

Digression: PSMs, Ontologies, and Patterns

[Menzies, 1997] argues that OO patterns of design [Gamma et al., 1995] and architecture [Buschmann et al., 1996, Shaw and Garlan, 1996] are analogous to PSMs and ontologies:

- Both recent abstract descriptions of supposedly common parts of many designs.

- OO patterns are typically structural; exceptions: some of the patterns in [Fowler, 1997]

- PSMs patterns are typically functional (recall page 35).

- Ontologies are very compatible with the patterns literature.

KA has building PSMs since at least 1983 [Chandrasekaran, 1983] and ontologies [Neches et al., 1991] since at least 1991:

- Current problems with PSMs/ontologies will be encountered in the future by patterns research

Problems with PSMs

(Caution: minority view, recall page 12.)

PSMs are not stable over time:

PSM primitives offered by Clancey [Clancey, 1992], KADS [Wielinga et al., 1992b], and SBF are different.

The number and nature of the PSMs not fixed. Often, new domain = new PSMs [Linster and Musen, 1992].

Diagnosis PSM, has not stabilized, may not do so in the near future:

- [Menzies, 1997] describes eight different models of diagnosis (four from the problem solving method community, four from elsewhere).

- Some common features,

- But, they reflect fundamentally divergent different views on how to perform diagnosis.

Ditto for ontologies (recall Cohen results on page 21- reuse results not impressive).

Ktype: QualityK

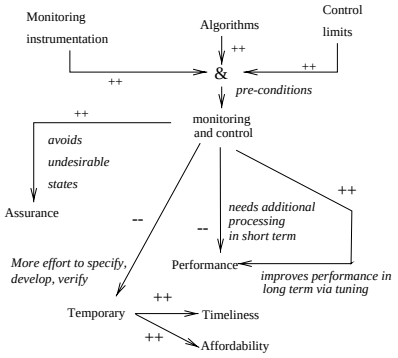
QualityK = some manner of generating an opinion about the value of the KB.

E.G. using behaviouralK:

- HT4 qualityK (page 31)- a good theory can generate worlds containing known or desired behaviour

E.G. using non-functional requirements:

- e.g. portability, evolvability, development affordability, security, privacy, or reusability:



[Boehm, 1996]

Ktype: QualityK (2) via Critical Success Metrics (CSMs)

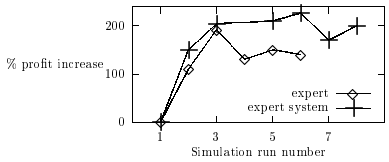
[Menzies, 2000]

Domain-specific; obvious in retrospect (can take weeks of analysis to uncover) may require extra data capture; reflect the contribution of the software to a particular business context, hence:

- Typically do not refer to internal properties of a program.
- Cannot be developed by programmers without from business users.
- Can only be collected once the program is running in its target context.

E.G. CSMs for a pig-farm management system [Menzies et al., 1992]:

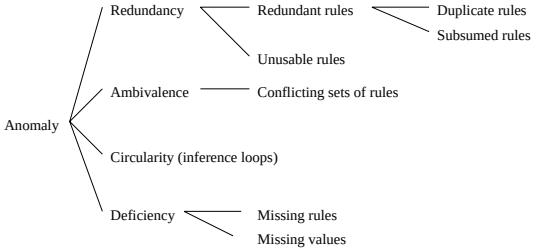
- NOT the European vs Australian protein unification model
- Rather, profit per M^2 per day



Ktype: QualityK (3) via Product-Oriented Assessment

House of quality [Shaw and Garlan, 1996].

Syntactic anomaly knowledge [Preece and Shinghal, 1992]:



And, by the way, fielded expert systems have these anomalies [Preece and Shinghal, 1992] ($^{Errors}/_{Anomalies}$):

	MMU	TAPES	NEURON	DISPLAN	DMS1
Size (literals)	105	150	190	350	540
Logical subsumption errors	0	$^5/_5$	0	$^4/_9$	5
Missing rule errors	0	$^{16}/_{16}$	0	$^{17}/_{59}$	0
Circularities in reasoning errors	0	0	0	$^{20}/_{24}$	0

ETM: a Product-Oriented Assessment Tool

ETM= the EXPECT transactions manager [Gil and Tallis, 1997].

- Detects errors in PSMs in a LOOM representation via a partial evaluation of methods.
  - This partial evaluation is driven by a particular example (a.k.a. an item in the behaviouralK).
- Errors are detected if a method cannot fire because the types of the input parameters to the methods are not available (formally, this is a variant on PSB “unusable rules” from page 42).
- The completeness of this error detector is a function of the completeness of the behaviouralK used to drive the partial evaluator.

Ktype: QualityK subtype: InconsistencyK

What is a conflict?

- General answer: `oops(X, not X).`

Other answers for different representations:

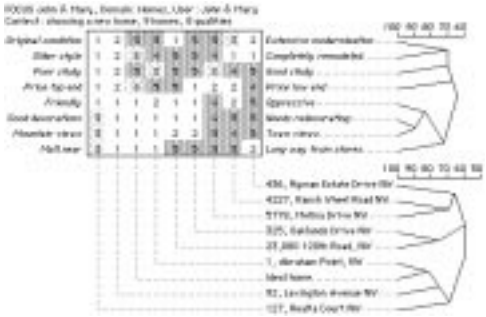
- HT4 (page 31) variables have one and one only state assignment:

```
oops(Var1/State1, Var1/State2):-
    not(State1 = State2).
```

- Class hierarchies: sub-class constraints should be violated by a super-class (the substitution principle).
- State-transistion diagrams from 2 authors. Conflicts if [Easterbrook and Nuseibeh, 1996]:
  - A transition exists between two states in one diagram;
  - Those two states appear in the other diagram;
  - The transition does not appear in the other diagram.

Ktype: QualityK subtype: InconsistencyK (2)

InconsistencyK for WordK using repertory grids [Gaines and Shaw, 1989]



- Consensus: same item, same categorisations;
- Correspondence: (a.k.a. synonyms) items with different names, but the same categorisation;
- True conflict: same items, different categorisations;
- Contrast: different items, different categorisations

Ktype: FixK

How to correct errors.

SEEK [Politakis, 1985]

```
seekDiagnosisRule
if   majorSymptoms and
     minorSymptoms and
     tests and
     not exclusions
then diagnosis is true
with confidence
     [definitely
     or probably
     or possibly]
```

SEEK2 [Ginsberg et al., 1988]

```
seekFixRule
if   the number of cases
     suggesting
     generalisation is
     greater than
     the number of cases
     suggesting
     specialisation
and  the most frequently
     missing component
     is the major symptoms
then delete some major
     symptoms
```


Ktype: SocialK

Knowledge cannot be understood with understanding its social context [Winograd and Flores, 1987, Clancey et al., 1996].

Paper approaches that informally describe the organizational context of a system:

- The organizational model of KADS [Wielinga et al., 1992b]
- The stakeholders of the Olle-126 [Olle et al., 1991].

Operationalized socialK:

- Clancey et.al.'s BRAHMS system [Clancey et al., 1996]: macro-workflow of an organization is an emergent process that is inferred from all the micro-behaviour of the agents in an organization.
- Design rationales are a record of why a community decided to change some aspect of a system [Moran and Carroll, 1996]. E.g. REMAP (page 29):
  - Logs design discussions.
  - Can track the impact of a change of mind to the constraints on the development.
  - Can replay previous discussions to generate historical understanding of how some decision was achieved.

The Recursive Maintenance Problem

Who shaves the barber? Who guards the guardians? How do we maintain the maintenance knowledge (QualityK, FixK, SocialK, Ontologies, PSMs)?

If we use KB2 to build and assess KB1, do we need KB3 to b&a KB2 and KB4 to b&a KB3 and ∴.

Theory: don't need to maintain KB2. So succinct, errors obvious. Wrong!

Results from the Sisyphus-II experiments (re-implement VT using your favorite KE approach). 13/25 elevator configurations failed due to basic flaw in all the Sisyphus-II implementations* [Zdrahal and Motta, 1996]. This problem was reported in only one of the other Sisyphus-II contributions:

	$\frac{ft}{min}$				
Capacity (lbs)	200	250	300	350	400
2000	√	√	×	√	√
2500	×	×	√	√	√
3000	×	√	√	√	√
3500	√	×	×	×	×
4000	×	×	×	×	×

*Sisyphus-II propose-and-revise is a local greedy search. Local hill-climbing may ignore solutions which are initially unpromising, but lead later on to better solutions.

5 Processing Activities

5 Processing Activities

Ktype: Knowledge type	Ptype:Knowledge Processing Activity				
	Acquire	Operation-alise	Fault	Fix	Preserve
WordK	all	most	few	few	few
SentenceK (ontologies)	most	most	many	many	few
BehaviouralK	some	few	few	few	none
PSMs	some	many	few	few	few
QualityK	few	few	none	none	none
FixK	few	many	none	none	none
SocialK	few	none	none	none	none

- Gathering "it": a.k.a. acquire
- Making "it" run: a.k.a. operationalise
- Finding bugs in "it": a.k.a. fault (or realizing an enhancement is required).
- Fixing "it's" errors (or implementing the enhancement)
- Ensuring that new fixes don't damage old fixes: a.k.a. preserve

Rapid Acquire Assumption

1997 survey of readers of the `comp.ai` news group.

Q: What is your maintenance technique?

A: Rapid acquire systems (RAS):

- high-level environment
- maybe point-and-click graphical editors
- maybe operationalisation support.

The RAS assumption:

- If knowledge is expressed at a sufficiently high-level...
- then its flaws are obvious and quick to change.
- Yeah, right...

RAS: Valid?

Merely browsing knowledge may not reveal how the inference engine will use it at runtime.

Merely watching a program execute may not reveal subtleties in its behaviour (the operationalisation KM assumption, page 33).

Very short descriptions of knowledge may contain invisible faults (examples below).

Small Models with Few Errors

Population growth:

- [Levins and Puccia, 1985]:

$$\frac{dN}{dT} = rN$$

  $T = time$, $N = population$, $r > 0, r < 0$ if environment benign, hostile respectively.

- Error #1: $^{dN}/_{dT} = 0$ when N at C, the maximum carrying capacity of the environment

- Population growth (again):

$$\frac{dN}{dT} = rN\left(1 - \frac{N}{C}\right)$$

- Error #2: Over-population ($N > C$), hostile environment $r < 0$, population increases (huh?)

Myers [Myers, 1977]: controlled experiments with 63 lines of PL/1

- 59 experienced data processing professionals
- Unlimited time
- 5 of the 15 errors found

Ptype: Acquire

Manual, unstructured, or semi-structured approaches.

Most SE/KE methodologies offer some of the following:

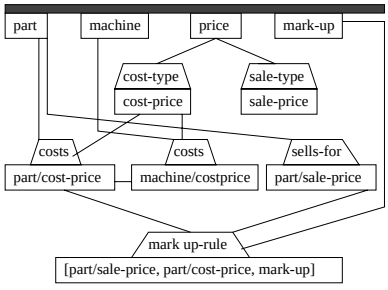
- A notation; e.g. ER [Date, 1995], Harel state charts [Harel, 1995], UML [Booch et al., 1997].
- Tool support drawing with (sometimes) operationalisation; e.g. RATIONAL-ROSE [Corporation, 1997], CAKE [Rich and Feldman, 1992];
- Representation of different stakeholders' opinions, goals (e.g. [Olle et al., 1991]) and combination rules (e.g. [Easterbrook and Nuseibeh, 1996]).
- Hints and tips:
  - On what data structures to collect; e.g. [Wielinga et al., 1992a,Olle et al., 1991,Gamma et al., 1995,Buschmann et al., 1996, Fowler, 1997, Coad et al., 1997]
  - On software process; e.g. [Booch, 1996].

Ptype: Acquire (2)

Manual or semi-automatic, structured techniques, e.g. database normalization [Date, 1995] or knowledge normalization [Debenham, 1998]. Conceptual model: items connected by objects to describe-

- Data are simple variables.
- Information are relations connecting variables.
- Knowledge that execute over the relations.

Decomposable item: may be constructed from other items or objects. Knowledge normalization = discard *decomposable* data, information, knowledge [Debenham, 1995].

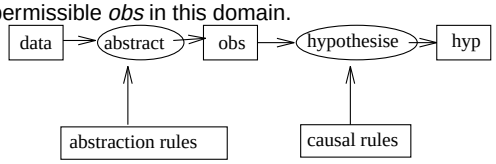


Acquire Using PSM Knowledge

Use formal methods to map from informal to formal knowledge [van Harmelen and Aben, 1996].

Restrict questioning to just those issues required to

- Select between PSMs
- Define PSM's ontological commitments (required data types)
- Refine selected PSM; e.g. if PSM=diagnosis, then ask about permissible *obs* in this domain.



E.G. SHELLEY [Wielinga et al., 1992a], MOLE [Kahn et al., 1985], SBF [Marques et al., 1992], SALT [Marcus and McDermott, 1989], RIME [Bachant and McDermott, 1984], PROTEGE-II [Eriksson et al., 1995]...

Some offer open-ended PSM definitions: PROTEGE-II, ?SBF

Others are harder to customize for a new PSM; e.g. SALT

Debatable point: are single-PSM tools (e.g. RIME based on SOAR, page 36) more or less flexible?

Other Acquire Techniques

Index on post-conditions [van Harmelen and Aben, 1996]

Expert critiquing systems:

> . . . programs that first cause their user to maximize the falsifiability of their statements and then proceed to check to see if errors exist. A good critic program doubts and traps its user into revealing his or her errors. It then attempts to help the user make the necessary repairs [Silverman, 1992].

Critics offer *cues* to the user:

- Coaxes the user from useless issues they are considering to useful issues that they are ignoring.
- E.G. Help text, special colors for highlights, "Wizards" in PC applications.

Two types or critics:

- *Influencers* cue: prevents an error happening.
- *Debiaser* cues: fixes after errors have happened (see the *Fix* activity, below). Debiasers less useful without positive feedback (influencers) [Silverman and Wenig, 1993].

Ptype: Acquire for Specialised KTypes

QARCC [Boehm, 1996] goal graphs [Mylopoulos et al., 1999] the acquiring of non-functional requirements quality knowledge (recall page 40).

VIEWER [Easterbrook and Nuseibeh, 1996] allows for the expression of inconsistencyK.

HT4 [Menzies and Compton, 1997]: tables for storing behaviouralK from experiments in the literature.

Case-based reasoning: builds behaviouralK via the incremental caching of parts of previous inferences (page 30).

Test generation tools: explore dependencies looking for inputs that exercise (e.g.) all branches: [Ginsberg, 1990,Zlatareva, 1992].

RDR: behaviouralK via *incremental capture*. In context of specific error, collect new knowledge. Remember the case that caused that error [Compton and Jansen, 1990].

BRAHMS: storage and organization of such socialK [Clancey et al., 1996].

Ptype: Operationalisation

"Operationalisation": the process of executing a KB either by direct interpretation or via compilation to some internal form.

E.g. SBF, PROTEGE-II, ..., TINA [Benjamins, 1994]. TINA KB:

```
rule1:diagnosis
      when prime_diagnostic_method
then symptom_detection and hypothesis_generation  and
     hypothesis_discrimination.

rule2:symptom_detection
      when ask_user_method
then apply_user_judgment.

rule3:symptom_detection
      when compare_symptom or detection_method
then generate_expectation and compare.

rule4:hypothesis_generation
      when empirical_hypothesis_generation_method
then associate and prediction_filter.

rule5:hypothesis_generation
      when model_based_hypothesis_method
then find_contributors and transform_to_hypothesis_set
      and prediction_based_filtering.

rule6:hypothesis_generation
      when hypothesis_generation_method
then select_hypothesis and collect_data
      and interpret_data.
```

Ptype: Operationalisation (2)

TINA output (after some queries to the user's db regarding the rules on the previous page).

```
model_based_hypothesis_generation_method {
   trace_back_method;
   intersection_method;
   corroboration                          }

trace_back_method {
   find_upstream  }

intersection_method {
   intersection     }

corroboration_method {
   select_random;
   simulate_hypothesis;
   compare;
   delete            }
```


Ptype: Fault

"Fault": recognizing that an operationalized KB has produced a behaviour that needs changing (bugs or enhancements).

Fault techniques for wordK: see repertory grids, page 45.

Fault techniques for sentenceK and PSMs:

- Expert inspection: use knowledge not captured in the 7 Ktypes
- BehaviouralK
- Using QualityK, inconsistencyK
  - Heuristic knowledge, may need maintenance.
  - No research known into how to fault qualityK, inconsistencyK (the recursive maintenance problem, see page 48).

Fault techniques for behaviouralK:

1. (Via the recursive maintenance problem): use some KB2 to model expectations of values in KB1's behaviouralK.
2. Use some algorithm to detect if new data does not fit into old data; e.g. [Colomb, 1999].
3. Via coverage of some feature X. Demand extensions to behaviouralK if coverage inadequate?

Ptype: Fault, subtype: Browse-around

Let the user manually generate their own explanations of why a variable was/was not set via:

- Fault localization (see below).
- MYCIN's "how" and "why" queries.
  - How= path to this point.
  - Why= current goal of backward chaining.
- SALT's "how" and "why not" and "what if".
  - "Why not X"= given conclusion Y, find a path to X blocked by some contribution to Y.
  - "What If X"= A hypothetical look downstream of some temporary setting. Variables set in what-if mode must be reset. ?HT4 a better approach (page 31).
  - No "why" since SALT was a forward-chainer.

Ptype: Fault, subtype: Browse-around (2)

Fensel & Schoenegge PSM assumption browser [Fensel and Schoenegge, 1997, Fensel and Schonegge, 1997]

- KIV (an interactive theorem prover) cannot solve a problem using a PSM.
- Identify, automatically, the missing logical formula that blocks PSM completion.
- Offer same to user as an assumption A.
- Let a user browse-around a first-order theory representing the PSMs in a "what if A"?

Ptype: Fault, subtype: Fault localization

Via backwards search of wordK dependencies (sentenceK):

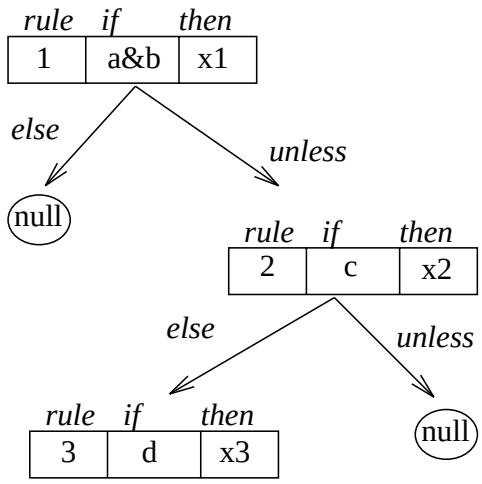
- Used in the MYCIN rule editor [Davis, 1976], Darden's theory anomaly localization [Darden, 1990], model-based diagnosis [Hamscher et al., 1992], RDR [Compton and Jansen, 1990],...
- Important technique

RDR (ripple down rules): optimized for fault localization and repair (see below) in KBS without PSMs [Compton et al., 1989, Compton and Jansen, 1990, Compton et al., 1992, Gaines and Compton, 1992, Mulholland et al., 1996, Preston et al., 1993, Richards and Compton, 1997, Richards and Menzies, 1997]

- Patch tree of rules
- Each rule has an *unless* patch (which is another rule and so on recursively).
- Patches stored with case that prompted patch creation.
- Final conclusion= conclusion of the last satisfied rule.
- Fault= the path over the patch tree to that rule.

Note: little or no recursive maintenance problem in RDR.

An RDR Tree



Ptype: Fix

"Fix": the process of removing a fault in an operationalized KB.

- Special class of "fix"=conflict resolution for knowledge collected from different sources.

Some general mechanisms for fixing:

- Ripple-down rules
- Conflict negotiation
- Specialization and/or generalization
- Machine learning
- Case-based reasoning
- KA scripts
- Others as described in [Menzies, 1999].

Fix via Ripple-down-rules

RDR = fault localization + repair:

- Current case = C= $Feature_1, Feature_2, ..$
- Path to faulty rule = P = $Feature_3, Feature_4, ...$
- Candidates for repair = difference list = C - P.
- Show difference list to an *oracle of repair.* Oracle picks X items off the difference list. These are added to new patch.

Patch only in the context of the last error? Crazy! Crazy?

- PIERS: St. Vincent's Hospital, Sydney,
- Modeled 20% of human biochemistry sufficiently well to make diagnoses that are 99% accurate [Preston et al., 1993].
- Development blended seamlessly with maintenance
- 2000-rule system, maintenance = a few minutes each day by domain experts.

RDR: Pros, Cons

Advantages:

- Practical implementation technique for the reflective [Schon, 1983] or situated cognition view [Menzies and Clancey, 1998]:
  - Design mostly happens when some concrete artifact *talks back* to the designer- typically by failing in some important situation.
  - Less concerned with the creation of some initial artifact than the on-going re-interpretation and adjustment of that artifact.
- Supports “preserve”: new fixes don’t break old fixes.
- Mostly avoids the recursive maintenance problem.
- Contrast with [de Brug et al., 1986]: large expert systems are notoriously hard to maintain.

Disadvantages:

- RDR tree not compatible with other common representation types (e.g. isa hierarchies, state charts)- but see [Richards and Compton, 1997,Lee and Compton, 1996,Colomb, 1999].
- Can’t process meta-knowledge such as PSMs - but see [Menzies and Mahidadia, 1997].

Fix via Negotiation

Explicitly represent the different viewpoints of different users.

Automatically detect conflicts between these viewpoints.

- Each inconsistency detection rule has an associated repair action.

Offer tools for resolution support.

Side effect of resolution = knowledge refinement [Easterbrook and Nuseibeh, 1996].

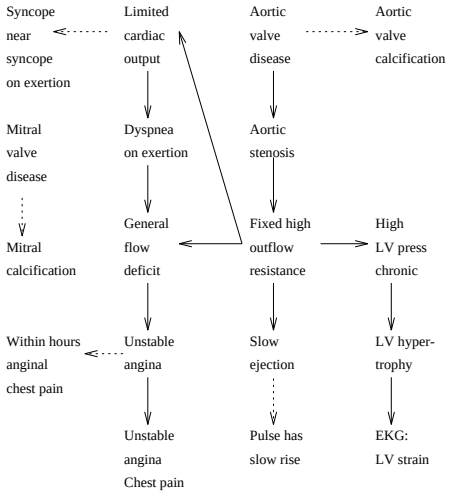
Note: conflicts may not be resolvable *now*.

- Need to be able to reason onwards in the presence of inconsistency.
- E.G. HT4, page 31.

Fixing via Case-Based Reasoning

Recall our heart attach patient, page 30 [Kolodner, 1993, p419]

New case: Dashed= added edges. Deleted: “murmur of as”.



CBR: Pros/Cons

CBR: strong on fixing cases (behaviouralK); weak on maintaining:

- Maintaining fixK: algorithms for turning old into new;
- SentenceK: the background causal knowledge or cardiac behaviour used above.

Fixing via specialization/ generalization

[Shapiro, 1983]. Undesired behaviours can be removed by specializing a pre-condition;

- e.g. increasing the number of tests in a conjunction.

Desired behaviour which was not achieved can be reached via generalizing a pre-condition;

- e.g. decreasing the number of tests in a conjunction.

Human-in-the-loop spec/gen:

- RDR: start with ultimate generalization: true. Specialize with each patch.
- SEEK: specialized by adding tests/symptoms or deleting exclusions or decreasing its confidence level. Generalized by removing tests/symptoms or adding exclusions or increasing its confidence level.

Fully automatic spec/gen:

- SEEK2 (page 46).
- Machine learning

Fix via Machine Learning

Machine learning (ML) algorithms [Michalski, 1993]:

- Input:
  - BehaviouralK: positive and negative examples (E^+,E^-) of what you want, don't want.
  - Background theory (B) which may be empty
- Output:
  - A new theory which covers more of E^+ or less of E^- than the initial background theory.
- Two types: inductive and deductive.

Inductive Learners

E.G. genetic algorithms, neural nets, decision tree learners, belief networks

Create a summary theory from the behaviouralK presented to them.

Data hungry: best with large E (100s to 1,000s)

Problems with very large E ($10,000s$): need pre-processors to prep the data (the data mining problem).

Typically ignore background knowledge (exception: inductive logic programming [Muggleton, 1991]):

- A user may be presented with a totally novel theory at the end of an inductive learning session.
  - BAD IDEA?
  - Users treasure their favorite portions of their KB (typically, the bits they defended from all critics).
  - Learners should not scribble all over treasured knowledge.

Deductive Learners

E.G. explanation-based generalization [van Harmelen and Bundy, 1988,Mitchell et al., 1986], chunking in rule-based systems [Laird et al., 1986]

Deductive learners input the steps taken by some inference engine and output a better set of inference steps.

- May cull the middle portions of a long inference procedure to connect inputs directly to outputs.

No scribble problem.

Less data hungry and make extensive use of the background theory.

- BAD IDEA?
- Less information to learn from = ? local minima.

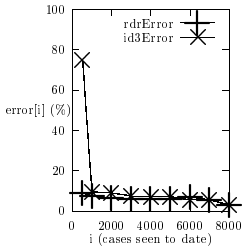
Machine Learning and KM

Deductive learning less reliable than inductive (the local minima problem).

Inductive learners have practical drawbacks:

- The scribble problem (see above).
- Data hungry = more expensive. May take months to collect data.

Manual KA using RDR $<$ inductive learning when $E > 1000s$ [Mansuri et al., 1991]



Fixing via KA Scripts

Databases: transaction management ensures completion of all table updates/deletes.

Knowledge bases: EXPECT Transaction Manager (ETM) ensures completion of all PSM knowledge updates/deletes:

- Triggered when EXPECT's partial evaluation strategy detects a fault.

	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

Recursive maintenance problem: how to check the KA scripts?

Ptype: Preserve

Stop a change for issue D from introducing problems into changes to made for A,B,C, . . .

Logic-based preserve:

- Initially, structure to support preserve.
- E.G. RDR (page 64), database and knowledge normalization (page 55).

Network-based preserve

- At all times, have access to the dependency network to check implications of a change.
- E.G. HT4, page 31.

Procedural-based preserve:

- Use a transaction manager to ensure completions
- E.G. KA scripts, page 77.

Other Preserve Tools

Design rationale [Moran and Carroll, 1996]:

- An annotation that describes the reasons for a fix.
- Current state-of-the-art:
  - Annotation tools for wordK and sentenceK only.
  - Creating such argumentation structures very costly to build. Interesting approach: Use CBR as a secretary to compare enrich new arguments by reflecting over old arguments [Fischer et al., 1996]

Data schema evolution (DSE).

- After some change to the logical model of the program, some parts of the program comply to the former version of the schema.
- Pressing problem in OODBMS (complex structures).
- Future problem for OO KBs (ontologies)
- Chain different schema versions, class to class.
- Coercion functions map instances along the versions.
- Current commercial state-of-the-art: can't handle changes that transcend class boundaries [Odberg, 1995, chpt2].

Discussion: How to Commission a KM Tool

4 Issues

1. Where in the 5*7 points of KM space does this system work?
2. What tools are offered for maintaining the maintenance KB? (the recursive maintenance problem).
3. Theoretically: Does the tool support preserve?
4. Practically, check that:
   - QualityK can assess the KB;
   - We can track $\frac{dQ}{dT}$ (quality): very useful for project management.

Search-Space Reflection

"Browse-around" core technique for KM. Used everywhere:

- Normalization = browse the dependency network
- Fault-localization
- "How", "why", "why not" and "what if" = browse dependencies around some word.
- Deductive learners = browse and edit pathways leading to a conclusion.
- CBR = browse which bits of the KB were used before.
- HT4=browse around and sort out what can be believed together.
- Requirements engineering= browse around to find fixes to stake-holder conflicts. ...

Does the tool support browse-around?

References

For the tutorial: Knowledge Maintenance: The State of the Art by Tim Menzies `tim@menzies.com`

1

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

[Angele et al., 1996] Angele, J., Fensel, D., and Studer, R. (1996). Domain and task modelling in mike. In et.al., A. S., editor, Domain Knowledge for Interactive System Design. *Chapman & Hall.*

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

[Benjamins, 1994] Benjamins, R. (1994). On a role of probem solving methods in knowledge acquisition- experiments with diagnostic strategies. In Proceedings of the European Knowledge Acquisition Workshop, 1994.

[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.*

[Boehm, 1996] Boehm, B. (1996). Aims for indentifying conflicts among quality requirements. In IEEE Software.

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

[Booch et al., 1997] Booch, G., Jacobsen, I., and Rumbaugh, J. (1997). Version 1.0 of the Unified Modeling Language. *Rational. `http://www.rational.com/ot/uml/1.0/index.html`.*

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

[Buschmann et al., 1996] Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., and Stal, M. (1996). A System of Patterns: Pattern-Oriented Software Architecture. *John Wiley & Sons.*

[Catlett, 1991] 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, *pages 53–67.*

[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., and 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.*

[Clancey et al., 1996] Clancey, W., Sachs, P., Sierhuis, M., and van Hoof, R. (1996). *Brahms: Simulating practice for work systems design. In Compton, P., Mizoguchi, R., Motoda, H., and Menzies, T., editors,* Proceedings PKAW '96: Pacific Knowledge Acquisition Workshop. *Department of Artificial Intelligence.*

[Clarke et al., 1986] Clarke, E., Emerson, E., and Sistla, A. (1986). *Automatic verification of finite-state concurrent systems using temporal logic specifications.* ACM Transactions on Programming Languages and Systems, *8(2):244–263.*

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

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

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

[Colomb, 1999] Colomb, R. (1999). *Representation of propositional expert systems as partial functions. Artificial Intelligence (to appear). Available from* `http://www.csee.uq.edu.au/~colomb/PartialFunctions.html`.

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

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

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

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

[Corporation, 1997] Corporation, R. S. (1997). *Rational-rose.* `http://www.rational.com`.

[Darden, 1990] Darden, L. (1990). *Diagnosing and fixing faults in theories. In Sharager, J. and Langley, P., editors,* Computational Models of Scientific Discovery and Theory Formation. *Morgan Kaufmann Publishers Inc.*

[Date, 1995] Date, C. (1995). An Introduction to Database Systems, *volume 6. Addison-Wesley.*

[Davis, 1976] Davis, R. (1976). Applications of Meta-Level Knowledge to the Construction, Mainteance and Use of Large Knowledge Bases. *PhD thesis, Stanford.*

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

[Debenham, 1995] Debenham, J. (1995). *Understanding expert systems maintenance. In* Proceedings Sixth International Conference on Database and Expert Systems Applications DEXA'95, London, September.

[Debenham, 1998] Debenham, J. (1998). Knowledge Engineering: Unifying Knowledge Base and Database Design. *Springer-Verlag.*

[Easterbrook and Nuseibeh, 1996] Easterbrook, S. and Nuseibeh, B. (1996). *Using viewpoints for inconsistency management.* BCS/IEE Software Engineering Journal, *pages 31–43.*

[Eriksson et al., 1995] Eriksson, H., Shahar, Y., Tu, S. W., Puerta, A. R., and Musen, M. A. (1995). *Task modeling with reusable problem-solving methods.* Artificial Intelligence, *79(2):293–326.*

[Fensel and Schoenegge, 1997] Fensel, D. and 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 and Schonegge, 1997] Fensel, D. and Schonegge, A. (1997). *Using kiv to specify and verify architecture of knowledge-based systems. In* Proceedings of the 12th IEEE International Conference on Automated Software Engineering (ASEC-97), Incline Village, Nevada, Nov 3-5.

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

[Fischer et al., 1996] Fischer, G., Lemke, A., McCall, R., and Morch, A. (1996). *Making argumentation serve design. In Moran, T. and Carroll, J., editors,* Design Rationale: Concepts, Techniques, and Use, *pages 267–293. Lawerence Erlbaum Associates.*

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

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

[Gaines and Compton, 1992] Gaines, B. and Compton, P. (1992). *Induction of ripple down rules. In* Proceedings, Australian AI '92, *pages 349–354. World Scientific.*

[Gaines and Shaw, 1989] Gaines, B. and Shaw, M. (1989). *Comparing the conceptual systems of experts. In* IJCAI '89, *pages 633–638.*

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

[Gil and Melz, 1996] Gil, Y. and Melz, E. (1996). *Explicit representations of problem-soving strategies to support knowledge acquisition. In* Proceedings AAAI' 96.

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

[Ginsberg, 1990] Ginsberg, A. (1990). *Theory reduction, theory revision, and retranslation. In* AAAI '90, *pages 777–782.*

[Ginsberg et al., 1988] Ginsberg, A., Weiss, S., and Politakis, P. (1988). *Automatic knowledge base refinement for classification systems.* Artificial Intelligence, *35:197–226.*

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

[Hamscher et al., 1992] Hamscher, W., Console, L., and DeKleer, J. (1992). Readings in Model-Based Diagnosis. *Morgan Kaufmann.*

[Harel, 1995] Harel, D. (1995). On visual formalisms. In Glasgow, J. and N.H. Narayanan, B. C., editors, Diagrammatic Reasoning, pages 235–271. The AAAI Press.

[Kahn et al., 1985] Kahn, G., Nowlan, S., and McDermott, J. (1985). Strategies for knowledge acquisition. IEEE Trsansactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-7:511–522.

[Kolodner, 1991] Kolodner, J. (1991). Improving Human Decision Making Through Case-Based Decision Aiding. AI Magazine, page 68.

[Kolodner, 1993] Kolodner, J. (1993). Case-Based Reasoning. Morgan Kaufmann.

[Laird et al., 1986] Laird, P. S., J. E., R., and Newell, A. (1986). Chunking in SOAR: The anatomy of a general learning mechanism. Machine Learning, 1(1):11–46.

[Lee and Compton, 1996] Lee, M. and Compton, P. (1996). From heuristic to causality. In Proceedings of the 3rd World Congress on Expert System (WcES'96).

[Levins and Puccia, 1985] Levins, R. and Puccia, C. (1985). Qualitative Modeling of Complex Systems: An Introduction to Loop Analysis and Time Averaging. Harvard University Press, Cambridge, Mass.

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

[Mansuri et al., 1991] Mansuri, Y., Compton, P., and Sammut, C. (1991). A comparison of a manual knowledge acquisition method and an inductive learning method. In Boose, J., Debenham, J., Gaines, B., and Quinlan, J., editors, Australian workshop on knowledge acquisition for knowledge based systems, Pokolbin, pages 114–132. University of Technology, Sydney.

[Marcus and McDermott, 1989] Marcus, S. and 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., and 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., and Tung, D. (1992). Easy Programming: Empowering People to Build Their own Applications. IEEE Expert, pages 16–29.

[Menzies, 1996] Menzies, T. (1996). Applications of abduction: Knowledge level modeling. International Journal of Human Computer Studies, 45:305–355. Available from `http://www.cse.unsw.edu.au/~timm/pub/docs/96abkl1.ps.gz`.

[Menzies, 1997] Menzies, T. (1997). OO patterns: Lessons from expert systems. Software Practice & Experience, 27(12):1457–1478. Available from `http://www.cse.unsw.edu.au/~timm/pub/docs/97probspatt.ps.gz`.

[Menzies, 1998] Menzies, T. (1998). Towards situated knowledge acquisition. International Journal of Human-Computer Studies, 49:867–893. Available from `http://www.cse.unsw.EDU.AU/~timm/pub/docs/98ijhcs`.

[Menzies, 1999] Menzies, T. (1999). Knowledge maintenance: The state of the art. The Knowledge Engineering Review, 14(1):1–46. Available from `http://www.cse.unsw.EDU.AU/~timm/pub/docs/97kmall.ps.gz`.

[Menzies, 2000] Menzies, T. (2000). Critical success metrics: Evaluation at the business-level. To appear, International Journal of Human-Computer Studies, special issue on evaluation of KE techniques.

[Menzies et al., 1992] Menzies, T., Black, J., Fleming, J., and Dean, M. (1992). An expert system for raising pigs. In The first Conference on Practical Applications of Prolog. Available from `http://www.cse.unsw.EDU.AU/~timm/pub/docs/ukapril92.ps.gz`.

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

[Menzies and Compton, 1997] Menzies, T. and Compton, P. (1997). Applications of abduction: Hypothesis testing of neuroendocrinological qualitative compartmental models. Artificial Intelligence in Medicine, 10:145–175. Available from `http://www.cse.unsw.edu.au/~timm/pub/docs/96aim.ps.gz`.

[Menzies and Mahidadia, 1997] Menzies, T. and Mahidadia, A. (1997). Ripple-down rationality: A framework for maintaining psms. In Workshop on Problem-Solving Methods for Knowledge-based Systems, IJCAI '97, August 23. Available from `http://www.cse.unsw.edu.au/~timm/pub/docs/97rdra.ps.gz`.

[Michalski, 1993] Michalski, R. (1993). Toward a unified theory of learning: Multistrategy task-adaptive learning. In Buchanan, B. G. and Wilkin, D. C., editors, Readings in Knowledge Acquisition and Learning: Automatic Construction and Improvement of Expert System. Morgan Kaufmann Publishers.

[Mitchell et al., 1986] Mitchell, T., Keller, R., and Kedar-Cabelli, S. T. (1986). Explanation-based generalization: A unifying view. Machine Learning, 1:47–80.

[Moran and Carroll, 1996] Moran, T. and Carroll, J. (1996). Design Rationale: Concepts, Techniques, and Use. Lawerence Erlbaum Associates.

[Motta and Zdrahal, 1996] Motta, E. and Zdrahal, Z. (1996). Parametric design problem solving. In Proceedings of the 10th Banff Knowledge Acquisition for Knowledge-Based System Workshop.

[Muggleton, 1991] Muggleton, S. (1991). Inductive logic programming. New Generation Computing, 8:295–318.

[Mulholland et al., 1996] Mulholland, M., Preston, P., Hibbert, B., and Compton, P. (1996). An expert system for ion chromatography developed using machine learning and knowledge in context. In Proceedings of the Sixth International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, Edinburgh.

[Myers, 1977] Myers, G. (1977). A controlled experiment in program testing and code walkthroughs/inspections. Communications of the ACM, 21:760–768.

[Mylopoulos et al., 1999] Mylopoulos, J., Cheng, L., and Yu, E. (1999). From object-oriented to goal-oriented requirements analysis. Communications of the ACM, 42(1):31–37.

[Neches et al., 1991] Neches, R., Fikes, R., Finin, T., Gruber, T., Patil, R., Senator, T., and Swartout., W. R. (1991). Enabling technology for knowledge sharing. AI Magazine, 12(3):16–36.

[Newell, 1993] Newell, A. (1993). Reflections on the Knowledge Level. Artificial Intelligence, 59:31–38.

[Odberg, 1995] Odberg, E. (1995). MultiPerspectives: Object Evolution and Schema Modification Management for Object-Oriented Databases. PhD thesis, Division of Computer Systems and Telematics, Norwegian Institute of Technology. 408 pages.

[Olle et al., 1991] Olle, T., Hagelstein, J., MacDonald, I., Rolland, C., Sol, H., Assche, F. V., and Verrijn-Stuart, A. (1991). Information Systems Methodologies: A Framework for Understanding. Addison-Wesley.

[Politakis, 1985] Politakis, P. (1985). Empirical Analsis for Expert Systems. Pitman.

[Preece and Shinghal, 1992] Preece, A. and Shinghal, R. (1992). Verifying knowledge bases by anomaly detection: An experience report. In ECAI '92.

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

[Quinlan, 1986] Quinlan, J. (1986). Induction of decision trees. Machine Learning, 1:81–106.

[Ramesh and Dhar, 1992] Ramesh, B. and Dhar, V. (1992). Supporing systems development by capturing deliberations during requirements engineering. IEEE Transactions on Software Engineering, 18(6):498–510.

[Rich and Feldman, 1992] Rich, C. and Feldman, Y. (1992). Seven layers of knowledge represeentation and reasoning in support of software development. IEEE Transactions on Software Engineering, 18(6):451–469.

[Richards and Compton, 1997] Richards, D. and Compton, P. (1997). 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.

[Richards and Menzies, 1997] Richards, D. and Menzies, T. (1997). Extending knowledge engineering to requirements engineering from multiple perspectives. In Menzies, T., Richards, D., and Compton, P., editors, Third Australian Knowledge Acquisition Workshop, Perth.

[Schon, 1983] Schon, D. (1983). The Reflective Practioner. Harper Collins/ Basic Books.

[Schreiber et al., 1994] Schreiber, A. T., Wielinga, B., Akkermans, J. M., Velde, W. V. D., and de Hoog, R. (1994). Commonkads. a comprehensive methodology for kbs development. IEEE Expert, 9(6):28–37.

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

[Shapiro, 1983] Shapiro, E. Y. (1983). Algorithmic program debugging. MIT Press, Cambridge, Massachusetts.

[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.

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

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

[Silverman and Wenig, 1993] Silverman, B. and Wenig, R. (1993). Engineering expert critics for cooperative systems. The Knowledge Engineering Review, 8(4):309–328.

[Soloway et al., 1987] Soloway, E., Bachant, J., and 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.

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

[Swartout and Gill, 1996] Swartout, B. and 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.

[Tallis, 1998] Tallis, M. (1998). A script-based approach to modifying knowledge-based systems. In Banff KAW '98 workshop. Available from `http://ksi.cpsc.ucalgary.ca/KAW/KAW98/tallis`.

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

[van Harmelen and Aben, 1996] van Harmelen, F. and Aben, M. (1996). Structure-preserving specification languages for knowledge-based systems. International Journal of Human-Computer Studies, 44:187–212.

[van Harmelen and Bundy, 1988] van Harmelen, F. and Bundy, A. (1988). Explanation-based generalisation = partial evaluation. Artificial Intelligence, pages 401–412.

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

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

[Winograd and Flores, 1987] Winograd, T. and Flores, F. (1987). On understanding computers and cognition: A new foundation for design: A respose to the reviews. Artificial Intelligence, 31:250–261.

[Yost, 1993] Yost, G. (1993). Acquiring knowledge in soar. IEEE Expert, pages 26–34.

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

[Zdrahal and Motta, 1996] Zdrahal, Z. and Motta, E. (1996). Improving competence by intergrating case-based reasoning and heuristic search. In 10th Banff Knowledge Acquisition for Knowledge-Based Systems Workshop, November 9-14, 1996, Banff, Canada.

[Zlatareva, 1992] Zlatareva, N. (1992). Ctms: A general framework for plausible reasoning. International Journal of Expert Systems, 5:229–247.