The Knowledge Engineering Review, Vol. 12:2, 1997, 209–217

Trends in knowledge modelling: report on the 7th KEML Workshop

ENRICO MOTTA

Knowledge Media Institute, The Open University, Walton Hall, Milton Keynes, MK7 6AA, UK (email:e.motta@open.ac.uk)

1 Background: knowledge modelling and KEML workshops

The KEML (Knowledge Engineering Methods and Languages) workshop which took place on 22–24 January at the Open University in Milton Keynes (UK) was the seventh in a series of workshops on methods and languages for knowledge engineering. Although the KEML acronym suggests a broad knowledge engineering connotation, in practice the main emphasis of these workshops is on the construction, formalisation, verification and use of *knowledge models*. The term "knowledge model" originates from the work of Allen Newell (1982), who proposed a level of description—the *knowledge level*—which abstracts from implementation-related considerations to focus on the actions, goals and knowledge embodied by a problem solving agent.

Among the various research enterprises which set to configure Newell's ideas in the practice of knowledge engineering, the EC-funded KADS projects (Wielinga et al., 1992; Schreiber et al., 1994b) were especially important and influential, and led to the creation of the KEML series of workshops. These started in 1991 with the goal of bringing together researchers and practitioners who were interested in the specification and use of KADS models. In recent years, the scope of these workshops has expanded from the original KADS-centred focus to include all research associated with knowledge modelling languages and techniques. This change of scope reflects the increased importance of knowledge modelling technology, both within and outside knowledge engineering. Within knowledge engineering, knowledge modelling is now an established research area, whose importance goes beyond a particular approach even one as influential as KADS. In particular, knowledge modelling technology is currently used to support several activities in the knowledge engineering life-cycle, including system specification, knowledge acquisition and validation and verification. Within a broader context, it is worth mentioning the increasing relevance of knowledge modelling technology to business management research. Nowadays it is almost as easy to find people talking about "building knowledge models" in business schools as it is in computer science departments. The reason can be found in the crucial role being played by *knowledge assets* in today's skill-intensive, service-oriented economy, where knowledge is not so much an asset among others, but the *key competitive asset* (Nonaka and Takeuchi, 1995). To make sense of knowledge assets, researchers in the emerging field of *knowledge management* are reusing, adapting and developing knowledge modelling solutions, thus contributing not only to the establishment and progress of the field, but also to its decoupling from system development. In a nutshell, Newell's vision of a knowledge level which abstracts from implementation considerations is being pursued even beyond the original idea of providing the "right" level of description of a problem solving agent. Today, knowledge modelling is a mature, distinct technology whose domain of investigation is the analysis and construction of both "static" (knowledge-intensive resources) and "dynamic" (knowledge-intensive processes) models.

2 Research issues in knowledge modelling

During the first half of the nineties the European research agenda in knowledge modelling was mainly defined by the two big EC-funded research projects, KADS-II (Schreiber et al., 1994b) and VITAL (Shadbolt et al., 1993), and much of the research focused on harnessing knowledge modelling techniques to produce robust, structured process models for knowledge-based system

(KBS) development. During the past 2–3 years, the focus has shifted away from comprehensive KBS development methodologies to centre on languages and techniques for the specification of reusable knowledge components. There are a number of reasons which explain this trend. First, as already pointed out, knowledge modelling technology is now becoming to some extent independent of system development scenarios. Secondly, on the software construction side the trend is to move away from expensive and error-prone "start-from-scratch" development activities and adopt reuse-centred models, which can take advantage of the reusable libraries of *domain ontologies* (Farquhar et al., 1996) and *problem solving methods* (O'Hara, 1995; Benjamins, 1993; Breuker and Van de Velde, 1994), which are becoming available. Finally, the explosion of the World Wide Web has had the dual benefit of (i) facilitating the interchange of ideas and models, thus stimulating the development of a "reuse-conscious" culture, and (ii) proposing new models of software architectures, which are based on a plug-and-play approach to system development.

Hence, at a coarse-grained level of analysis, we can characterise current knowledge modelling research as providing languages, tools and techniques to support the identification, specification, evaluation, and reuse of knowledge-based components. More specifically, there are a number of research issues which are currently being addressed:

- **Modelling languages**. Reusable knowledge-based components are described at the knowledge level. This approach is consistent with current views in the wider context of software reuse. As Krueger (1992) points out, "Abstraction is the essential feature in any reuse technique". While many researchers use informal notations, which are much easier to use, formal and operational descriptions are also needed to support validation and verification and to guarantee the robustness of a model. Several languages have been proposed over the past few years—see, for instance, the review of KADS languages by Fensel and van Harmelen (1994). Nevertheless, there is still much work to do in particular with respect to formal modelling of control knowledge, providing seamless integration of different levels of description (formal, informal and operational), and raising the level of support, i.e., providing high-level modelling constructs.
- **Libraries of reusable components**. A number of libraries of reusable components have been produced in recent years (Farquhar et al., 1996; O'Hara, 1995; Benjamins, 1993; Breuker and Van de Velde, 1994; Marcus, 1988), which differ in many respects. Some focus on ontologies (Farquhar et al.), others on problem solving methods (Benjamins, Marcus, O'Hara). Among the latter, some are collections of complete problem solvers (Marcus), others are characterised as partial, configurable specifications of inference and control structures (Breuker and Van de Velde, Benjamins, O'Hara). Some libraries are described only informally (Benjamins), others are mainly described at the symbol level (Marcus). Finally, these libraries differ in the support they provide for configuring a method to an application. For example, O'Hara's library is specified in terms of a set of rewrite rules and method configuration is characterised as a refinement process in which domain knowledge is used to drive the selection of the appropriate model refinement rule (Motta et al., 1994, 1996). In contrast with this approach, Marcus' library is essentially static. A fixed control and inference structure is associated with each method, and the knowledge acquisition process consists of eliciting and representing the domain knowledge required by the *knowledge roles* associated with each method. In general, much additional research is still needed to support effective reuse of model components. The World Wide Web provides a software infrastructure for knowledge sharing and reuse but to-date there is very little tool support for selecting and configuring pre-existing components and for collaborative model development. Moreover, it will be difficult to make significant progress on the interoperability of modelling components until a standard specification language for problem solving methods is produced.
- **Modelling frameworks**. A modelling framework provides an organisation of the basic types of knowledge types which constitute an application. In recent years, in part because of the impact of KADS-centred research projects and because of cross-fertilisation between different groups of researchers, there is a broad consensus that task, method and domain models provide useful classes of reusable components. In other words, they provide a useful epistemology for

structuring a library, and useful general categories for a coarse-grained analysis of application domains. However, despite this "broad consensus" there are many fine-grained differences between different approaches. To give just one example I will mention here the so-called *interaction problem* (Bylander and Chandrasekaran, 1988). This states that the nature and the representation of domain knowledge depends on the choice of a problem solving model. If one takes a radical interpretation of this view, then the consequence is that there is very little use for task-independent domain ontologies, given that both the knowledge and the representation decisions required for a particular application will depend on the specific problem solving model which has been chosen. Of course, those researchers working on task-independent ontologies, e.g., Guarino (1996), criticise such a view and argue that the "interaction effects" have only to do with symbol-level efficiency issues, and therefore do not apply to abstract, knowledge-level application models. The bottom line here is that consensus on how to model applications and structure a reuse-centred development process is still skin-deep. Important differences exist in the knowledge modelling community, especially with regard to the nature and role of domain knowledge, and the relationship between ontologies and problem solving models.

3 Contributions from KEML '97

Twenty research papers were submitted to KEML '97, of which twelve were accepted for presentation at the workshop. In addition, Guus Schreiber from the University of Amsterdam and Dieter Fensel from the University of Karlsruhe gave invited talks. The workshop was divided into five sessions, where each session comprised one or more presentations thematically related. Within each session much time was devoted to discussion.

From a general point of view, the papers accepted at KEML '97 can be characterised in terms of one or more of three core KEML research areas: modelling languages, modelling frameworks, and problem solving methods. Specifically, the papers in this year's workshop address research issues which include: the formal treatment of control knowledge; the specification of task-specific or general-purpose modelling languages; the integration of problem solving and learning; the evaluation of alternative modelling frameworks; and the formalisation, development and reuse of problem solving methods.

In the following sections I will give an overview of the presentations given at KEML '97. This is structured in terms of the five sessions held at the workshop.

3.1 Problem solving methods

The first session of the workshop was dedicated to research on problem solving methods and included presentations by Frank van Harmelen (Free University of Amsterdam), Pascal Beys (University of Amsterdam), and myself. Following these three presentations the session was concluded by Dieter Fensel's invited talk.

In my presentation I discussed the work on parametric design problem solving carried out in the context of the EC-funded Encode project, in collaboration with my colleague Zdenek Zdrahal. In the talk I illustrated both the design-related legacy of the work and its wider implications for research on knowledge modelling. From a design point of view, this work has produced a number of reusable resources, which include *task* and *method ontologies* for design, a library of design-specific problem solving methods, and a method-independent, but task-specific *problem solving framework*. The latter specifies the generic tasks which occur in parametric design problem solving. This framework has been implemented in Common Lisp and provides a kind of task-specific expert system shell, configurable for design methods and applications. From a modelling point of view, our work provides a novel approach to both the development and organisation of a class of problem solving methods. This "novel approach to development" centres around the use of the search paradigm as a kind of "methodological device" to bridge the gap between a task ontology and a class of problem solving methods. With respect to organisation issues, our approach shows that it is

possible to organise a library of heterogeneous methods—where some are heuristic-based, other case-based, other search-based—as alternative refinements of a common, generic model. This organisation makes it easier to compare, evaluate and contrast problem solving methods. In particular, in my talk I illustrated an example in which the redescriptions of a number of problem solving methods in terms of the generic framework allowed us to explain the reasons for the different competence patterns exhibited by alternative problem solving methods.

The next presentation was given by Frank van Harmelen, who discussed an approach to the automatic configuration of problem solving methods for an application. This approach is based on the idea that once a class of methods is specified as a parametrised, declarative structure, the problem of configuring a problem solving method can be characterised as a parametric design task. Obviously, the approach assumes that, given a class of problem solving methods, it is possible to build a generic, parametrised model subsuming each method in the class. This assumption shows important similarities between the work I illustrated in my presentation and van Harmelen's, as both approaches are predicated on the development of a task-specific, but method-independent problem solving model. In particular, van Harmelen illustrated his approach in the area of diagnostic systems, and showed that a number of diagnostic problem solving methods can be characterised as instantiations of a generic inference structure, parametrised along six dimensions. He then discussed a process model for configuring such generic inference structure, which is based on the Propose-Critique-Modify model of design problem solving (Chandrasekaran, 1990). This model requires domain-specific heuristic knowledge, and van Harmelen emphasised that such reliance on domain knowledge especially distinguishes this approach to automatic method configuration from alternative proposals (Benjamins, 1993).

The third presentation in the session on problem solving methods was given by Pascal Beys, who illustrated an approach to generalising from specific problem solving methods which is significantly different from those presented in the two earlier talks. While both van Harmelen and I illustrated the advantages derived by abstracting a task-specific problem solving model from a class of problem solving methods, Beys' approach generalises from task-specific methods by reformulating them in a *task-neutral* way. This redescription increases the reusability of methods across classes of applications, at least in theory. Of course, the problem with task-neutral methods is that, while they are in theory applicable to a wider range of domains, they are specified at a higher level of abstraction than task-specific ones and are in practice less easily reusable. To ameliorate this problem Beys showed a technique for automating the mapping from task-neutral methods to domain. This technique is based on the idea that a method can be characterised in terms of a number of assumptions and knowledge requirements. These can be automatically verified and acquired by analysing a domain knowledge base.

The first day of the workshop was concluded by an invited talk given by Dieter Fensel, who discussed a formal approach to the specification of reusable components for knowledge-based systems. His framework distinguishes between task definition problem solving method and domain model. These components are specified separately and are therefore independent of each other. To configure them to build an application model Fensel introduces the notion of *adapter*. This is a structure which (i) maps the different terminologies used by the various components, and (ii) often introduces additional *assumptions*, e.g., in some case a task definition might have to be weakened to meet the requirements of a method. An important aspect of this work is that the entire framework is described formally, and as a result it is possible to verify the resulting application model. In particular, Fensel discussed the results obtained by automatically verifying application models—using the KIV theorem prover—and showed that these formal analyses typically lead to the discovery of additional, hidden domain assumptions.

3.2 Graph-centred models

The initial session of the second day of the workshop was dedicated to graph-centred conceptual modelling languages. The first presentation was given by Corinne Bos (ISEN, France) who

extended conceptual graphs to represent "stereotyped behaviour in human organisations". This has typically to do with codified guidelines such as those which describe actions to take in case of emergencies. Bos illustrated the problems with informal specifications of codified guidelines: these are typically under-specified, given that much background knowledge, such as prerequisites and intensions, is assumed on the part of the reader. Moreover, there are often complex temporal relationships between actions which are not represented in the guidelines. Therefore, there is a need for modelling these guidelines in a formal or semi-formal way. Bos illustrated a number of requirements on the modelling language, some related to representation issues, e.g., being able to represent knowledge about time, some related to practical ones, e.g., it should be possible for domain experts to understand the formalised guidelines. Bos argued that the conceptual graph representation developed by Sowa (1984) fulfils the relevant formal and practical requirements, and therefore she adopted it as the underlying modelling language. Then, she discussed the various formal and ontological issues which she had to address to represent behavioural models in conceptual graphs. In particular, her ontology for modelling behavioural expertise characterises behaviour in terms of *action models*. These consist of *states*, *preconditions*, *progresses* and *effects*. Complex temporal relations can exist between actions, which are often described inaccurately by domain experts. To address this problem, Bos' ontology provides schema for modelling temporal relations, which take into account both the formal and pragmatic issues relevant to modelling stereotyped behaviour.

The next presentation was given by Alexandra Galatescu (ICI-Research Institute in Informatics, Rumania), and was closely related to the one given by Bos. While Galatescu's goal of providing a unique notation for the control, data and functional perspectives used in software specifications differed from Bos', both researchers followed a similar approach, based on the use of conceptual graphs. In particular, Galatescu proposed and formalised the notion of *activity-centred graph* as a way of abstracting from data, functional and control perspectives. Next, she illustrated a set of operations on activity graphs—such as *abstraction*, *contraction*, *expansion* and *join*—and she showed that the resulting notation could be used to model data, as well as functional and control models. Finally, she compared her work to other specification languages in both software (OMT) and knowledge engineering (KADS).

The final presentation of the graph-centred session was given by Gilles Kassel (Universite' de Technologies de Compiegne), who discussed extensions to a semantic-network-like modelling language, which make it "strongly intensional". In particular, Kassel showed that his proposed language made it possible to model both the extensional and intensional aspects of KADS metaclasses, specifically the association between metaclasses and domain objects.

3.3 CML 2.0: a new version of the KADS modelling language

Following the session on graph-centred models, Guus Schreiber (University of Amsterdam) presented a new release of the CML conceptual modelling language (Schreiber et al., 1994a), and illustrated the many and often conflicting requirements which had to be addressed when defining the new version of the language. These requirements include the need to be compatible with mainstream software engineering approaches, the need to support the modelling of alternative viewpoints, the need to facilitate the operationalisation of CML models, and the need to provide downward compatibility with previous releases of the language. A particular interesting conflict occurred when trying to reconcile the relatively sophisticated modelling demands arising from the knowledge modelling community with the need to provide a user-friendly and "conservative" modelling language, which should maintain compatibility with standard software engineering notations.

In the context of the workshop, this talk turned out to be particularly interesting because in contrast with all the other talks, which presented state-of-the-art research issues, it mainly focused on the many pragmatic issues which are faced by those who try to deploy knowledge modelling technology in the commercial sector. In particular, Schreiber emphasised the importance of

providing modelling languages which are consistent with the technical solutions and (especially) the standards used in mainstream computing.

3.4 Formal modelling languages

The session on formal modelling languages comprised two presentations. Christine Pierret-Golbreich (University Paris-Sud) discussed a formal approach, based on preference logic, to modelling *strategic knowledge*. Stefan Decker (University of Karlsruhe) presented a new version of the KARL modelling language.

Pierret-Golbreich's work addresses a topic which has received much attention in recent years: conceptual and formal aspects of *flexible problem solving*. This is defined as the capability of a problem solver to select dynamically problem solving methods when attempting to solve a problem. A typical, simple scenario is one in which a fast method with limited competence is applied on a (sub-)problem and, if this fails, a more expensive one is tried. While a number of frameworks supporting flexible problem solving have been proposed (Benjamins, 1993; Laird et al., 1987), Pierret-Golbreich's paper is one of the few attempts to tackle this problem formally. Her approach is based on the use of *default logic* to express the context in which a method can be preferred to an alternative one. In her presentation Pierret-Golbreich introduced the *Task* modelling framework (Pierret-Golbreich, 1994), illustrated a number of strategies for specifying flexible problem solving control, and defined a formal semantics for strategic knowledge, which is based on *abstract data types*.

In the following presentation Stefan Decker illustrated a number of improvements to the KARL language, which especially aim to facilitate the representation of reusable problem solving methods. In particular, KARL now includes operators for representing state changes formally and provides constructs for representing and integrating method and task ontologies. An interesting trade-off concerns the restriction to Horn clauses, enforced by KARL. On the one hand this restriction ensures that KARL can be given both a formal and an operational semantics. On the other hand, it can be in some cases a "modelling straight jacket". Decker made the point that for validation reasons, they decided to stick to an operational specification language, and therefore only a limited extension to Horn clauses has been incorporated into the new version of the language.

3.5 Agents, planning and learning

The final session of the workshop was dedicated to two papers discussing the application of modelling languages to agent and plan modelling, and to a third one which described a modelling framework integrating problem solving with learning.

The first presentation was given by Frank van Harmelen, who described—on behalf of Rineke Verbrugge, who had to withdraw from the workshop at short notice—work carried out by the Desire group at the Free University of Amsterdam. In his talk, van Harmelen discussed a formal approach to modelling the dynamic behaviour of an agent in the context of the Desire framework (Brazier et al., 1995). In addition to the "usual" modelling support for specifying domain, task and problem solving knowledge, Desire also provides constructs for representing other types of knowledge, such as the association between tasks and agents, the internal behaviour of an agent, and communication between agents. An agent in Desire is viewed as a cognitive entity, which is described in terms of a number of attitudes such as beliefs, desires and intentions. The presentation focused on the specification of *commitment strategies*, a number of which were discussed. For instance, blindly committed agents drop a goal only when it has been achieved, while open-minded agents drop a goal also when they no longer desire to achieve it. An important aspect of the work on Desire is that it provides a framework for applying formal modelling technology to multi-agent systems. On the one hand, this work is relevant to those researchers who carry out cognitively-motivated research on multi-agent systems. On the other hand, Desire can be used in the software

engineering practice, to specify software components in today's distributed, agent-based software architectures.

The second presentation was given by Silvia Miksch (Vienna University of Technology) who discussed a language for representing *skeletal plans*. These are schema that capture the essence of a procedure but are only partially specified (i.e., they leave room for customisation in attaining a goal). Skeletal plans are used extensively in health care, where clinical guidelines and protocols ensure consistent approaches to therapy throughout organisations and enable statistical comparisons between health care centres. However, guidelines and protocols are often incomplete and ambiguous and for this reason it is important to try and represent them in some formalism which enables their verification and validation. Miksch discussed the requirements associated with designing a language for representing skeletal plans. In particular, she focused on the need to have a language which could express a rich set of temporal relations, as well as allowing the user to express intentions and annotate plans. The proposed formalism is called Asbru and is centred around the key concept of *temporal pattern*. This is a representation structure which characterises the value of a parameter (or an abstraction) in a particular context and with a particular *time annotation*. A time annotation is a construct which supports the representation of time information with various degrees of uncertainty. Asbru also provides concepts which support the specification of intentions, conditions (such as abort condition and setup precondition), preferences, and effects (of an action). Finally, a control language makes it possible to specify parallel, sequential and cyclical plans.

The last presentation of the workshop was given by Josep Lluis Arcos (AI Research Institute, Spain), who illustrated the Noos framework for problem solving and learning. The Noos approach distinguishes between domain, problem solving, and metalevel knowledge. Metalevel knowledge is similar to (although strictly speaking more generic than) the strategic knowledge discussed by Christine Pierret-Golbreich and is used to model flexible problem solving. Knowledge representation in Noos is based on *feature terms*, which, in contrast with their typical use in natural language research, are used to model not just domain but also problem solving knowledge. The most important aspect of the Noos approach concerns the integration of a number of learning methods in the overall framework. Learning in Noos is modelled as a process with three main subtasks: introspection (finding interesting examples in the episodic memory); construction (generating new knowledge); and revision (integrating new knowledge in the system memory). Within this task structure Noos supports a number of learning methods, which include induction methods, case-based reasoning and analytical learning methods.

4 Conclusions

A number of general points can be abstracted from the various presentations and discussions which have taken place at KEML '97:

- **Formal modelling**. The "traditional" emphasis on formal approaches is still present in the KEML community, and also this year many papers discussed formal modelling issues. In particular, researchers are currently focusing on higher-level formalisation problems such as modelling strategic knowledge, integrating the specification of ontologies with that of tasks and methods, and modelling additional modalities, such as agent behaviour.
- **Conceptual graphs**. As shown by the session devoted to graph-centred representations, semantic network representations—in particular conceptual graphs—are still on active area of research in this community. This class of formalisms is favourite by many researchers who require both user-friendliness and a formal semantics from a modelling language.
- **From task-to-method architectures to generic problem solving models**. At the beginning of the knowledge modelling "revolution" the notion of task analysis was introduced and researchers moved from symbol-level models (typically rule-based) of problem solving to knowledge-level, task-centred analyses. While originally the notion of task comprised both a task specification and

a body (i.e., a "hardwired" problem solving method), the two notions were eventually decoupled and we saw the emergence of libraries of methods indexed by task features. A couple of presentations at this year's KEML outlined a new approach to the development, analysis, and organisation of problem solving methods which goes beyond simple task-to-methods associations. This new approach is based on the development of a generic problem solving model, which is associated to a generic task. Thus, problem solving methods are not just "ways to solve a task" but are precisely characterised as refinements of this partially specified generic model. This approach makes it possible to provide homogeneous characterisations of heterogeneous problem solving methods, which facilitate developing, configuring, and comparing them. Moreover, this approach provides a novel solution to the problem of organising a library of problem solving methods.

In conclusion this year's workshop was a successful event and confirmed both the vitality of knowledge modelling research in Europe and the continued validity and relevance of the KEML series of workshops. Next year the workshop will be held in May at the University of Karlsruhe, and will be chaired by Dieter Fensel. Additional information on the KEML research community can be found at the KEML web site, `ftp://swi.psy.uva.nl/pub/keml/keml.html`.

References

Benjamins, R, 1993. "Problem solving methods for diagnosis" PhD Thesis, Department of Social Science Informatics, University of Amsterdam.

Brazier, FMT, Dunin-Keplicz, B, Jennings, NR and Treur, J, 1995. "Formal specifications of multi-agent systems: a real-world case" In: V Lesser (ed) *Proc. Ist International Conference on Multi-Agent Systems—ICMAS-95* 25–32, MIT Press.

Breuker, JA and Van de Velde, W, 1994. *CommonKADS Library for Expertise Modelling* IOS Press.

Bylander, T and Chandrasekaran, B, 1988. "Generic tasks in knowledge-based reasoning: the right level of abstraction for knowledge acquisition" In: B Gaines and J Boose (eds) *Knowledge Acquisition for Knowledge-Based Systems—volume 1* 65–77, Academic Press.

Chandrasekaran, B, 1990. "Design problem solving: a task analysis" *AI Magazine* **11**(4), Winter, 59–71.

Farquhar, A, Fikes, R and Rice, J, 1996. "The Ontolingua server: a tool for collaborative ontology construction" In: B Gaines and M Musen (eds) *Proc. 10th Banff Knowledge Acquisition for Knowledge-Based Systems Workshop* Banff, Alberta, Canada.

Fensel, D and van Harmelen, F, 1994. "A comparison of languages which operationalize and formalize KADS models of expertise" *The Knowledge Engineering Review* **9**(2).

Guarino, N, 1996. Understanding, building, and using ontologies" In: B Gaines and M Musen (eds) *Proc. 10th Banff Knowledge Acquisition for Knowledge-Based Systems Workshop* Banff, Alberta, Canada.

Krueger, CW, 1992. "Software reuse" *ACM Computing Surveys* **24**(2) 131–183.

Laird, J, Newell, A and Rosenbloom, P, 1987. "Soar: an architecture for general intelligence" *Artificial Intelligence* **33** 1–64.

Marcus, S (ed), 1988. *Automatic Knowledge Acquisition for Expert Systems* Kluwer Academic.

Motta, E, O'Hara, K and Shadbolt, N, 1994. "Grounding GDMs: a structured case study" *International Journal of Human-Computer Studies* **40**(2) 315–347.

Motta, E, Stutt, A, Zdrahal, Z, O'Hara, K and Shadbolt, N, 1996. "Solving VT in VITAL: a study in model construction and knowledge reuse" *International Journal of Human-Computer Studies* **44**(3/4) 333–371.

Newell, A, 1982. "The knowledge level" *Artificial Intelligence* **18**(1) 87–127.

Nonaka, I and Takeuchi, H, 1995. *The Knowledge Creating Company: How Japanese Companies Create the Dynamics of Innovation* Oxford University Press.

O'Hara, K, 1995. "The GDM Grammar, v.4.6" *VITAL Project Report NOTT/T252.3.3*. (Available from the author at AI Group, Department of Psychology, University of Nottingham, UK.)

Pierret-Golbreich, C, 1994. "TASK MODEL: a framework for the deign of models of expertise and their operationalization" In: B Gaines and M Musen (eds) *Proc. 8th Banff Knowledge Acquisition for Knowledge-Based Systems Workshop* Banff, Alberta, Canada.

Schreiber, ATh, Wielinga, BJ, Akkermans, H, van de Velde, W and Anjewierden, A, 1994a. "CML: The CommonKADS Conceptual Modelling Language" In: L Steels, ATh Schreiber and W van de Velde (eds) *A Future for Knowledge Acquisition, Proc. of the 8th European Knowledge Acquisition Workshop: LNAI 867* Springer-Verlag, 283–300.

Schreiber, ATh, Wielinga, BJ, de Hoog, R, Akkermans, H, van der Velde, W and Anjewierden, A, 1994b. "CommonKADS: a comprehensive methodology for KBS development" *IEEE Expert* 28–37, December.

Shadbolt, N, Motta, E and Rouge, A, 1993. "Constructing knowledge based systems" *IEEE Software* November.

Sowa, J, 1984. *Conceptual Structures: Information Processing in Mind and Machine* Addison-Wesley.

Steels, L, 1990. "Components of expertise" *AI Magazine* Summer issue.

Wielinga, BJ, Schreiber, ATh and Breuker, J, 1992. "KADS: a modelling approach to knowledge engineering" *Knowledge Acquisition* **4**(1) 5–53.