Reuse of Problem-Solving Methods in Knowledge Engineering

Dieter Fensel

Institut fur Angewandte Informatik und Formale Beschreibungsverfahren (AIFB)
University of Karlsruhe, P.O. Box 6980, 76128 Karlsruhe, Germany
Tel: 49-721-6084754
Fax: 49-721-693717
E-mail: fensel@aifb.uni-karlsruhe.de

Abstract

The paper proposes the combination of two kinds of software reuse in knowledge engineering. On the one hand, a lot of work has been done to develop generic problem-solving methods which can be applied to different domains and tasks. These reusable methods simplify the process of building a knowledge-based system by providing a predefined model of the problem-solving process which guides the further acquisition of domain knowledge. One the other hand, several formal knowledge specification languages have been developed which allow the description of a model of expertise at a high but precise conceptual level. In the paper it is argued that formal languages can also be used to improve the reuse of predefined problem-solving methods.

Keywords: Reuse, requirements specification, formal specifications, source-code libraries, problem-solving methods, knowledge engineering

Workshop Goals: Learning and networking, discussing reuse of formal specifications and the use of formal specifications for reuse.

Working Groups: Reuse and formal methods, Domain analysis.

1 Background

During the last four years I have worked in the domain of knowledge engineering. The main part of my work was concerned with the development of the formal and operational knowledge specification language KARL [1] (i.e., reuse by a very-high-level language cf. [2]). A second line of research on reuse was the formal specification of reusable problem-solving methods in KARL. Currently, the development and description of generic and reusable problem-solving methods is an important branch of research in knowledge engineering. The description of the problem-solving behaviour of an expert system by a generic problem-solving method characterizes the so-called second generation expert systems [3]. My work is part of the MIKE-project (**M**odel-based and **I**ncremental **K**nowledge **E**ngineering, cf. [4]) at the University of Karlsruhe which aims at developing methods and tools for the process of building knowledge-based systems.

2 Position and Comparison

First, the rationale of reusable problem-solving methods is discussed. Second, the development of formal and operational knowledge specifi

2.1 Reuse of Problem-Solving Methods

During the last decade, several problem-solving methods have been discovered in the domain of experts systems. [5] analysed several expert systems for diagnosis and remodelled the so-called problem-solving method heuristic classification, which was implicitly used by all of these expert systems without being explicitly represented. In spite of various differences all systems have three steps in common: a data-abstraction, i.e., concrete values are mapped on intervals; a heuristic match from abstract problem descriptions to abstract solutions; the hierarchical refinement of these solutions until a final solution has been found.
Thus, Clancey describes a problem-solving method without referring to implementational details of the used knowledge representation formalism of the different systems or to the domain specific knowledge they used. The description of a problem-solving method independently from its implementation and from an application domain are the essential necessities for its reuse. In [6] a whole set of different problem-solving methods is described at this level. These problem-solving methods are described by defining: the single steps called knowledge sources or inference actions; the structural dependencies of these single steps by a so-called inference structure; and the control flow, i.e. the sequence of these single steps. Such a problem-solving method is often called interpretation model because it can be used as an guideline for the knowledge acquisition process. Once selected, the method can guide the elicitation and interpretation of the experts domain knowledge required for solving the specific task. A more operational point of view on reuse is taken by theroblem-solving methods.
To overcome these shortcomings of application generators, [7] proposes a library of reusable mechanisms or generic tasks. These methods have a finer grain size than conventional expert system shells. A complete problem-solving process must be modelled by several mechanisms. These approaches are analogous to the *source-code library* idea in software engineering. The main characteristics of these approaches is that currently these mechanisms are only described by code and by informal descriptions of the code. Therefore, there is little support for the selection, specialization, and integration of these mechanisms. The lack of descriptions that abstract from implementational

details, but do not lead to imprecise natural-language descriptions, makes it difficult to compare such mechanisms, and to provide a precise description of their problem-solving capability. Mathemaows developers to address such components through their names. Currently, there exist neither exhaustive libraries of reusable mec

2.2 Reuse by Formal Specification Languages

A further line of software reuse is the development of specific formal or operational knowledge specification languages. Very high-level languages (VHLL) are executable like high-level languages but they aim to increase the level of abstraction. "The primary concern in VHLLs is not efficiency in program execution but rather efficiency in implementing and modifying programs" [2]. In VHLLs, the automatic code-generation is similar to that of application generators. Yet, the VHLLs are application-independent general-purpose languages of higher complexity (e.g., they use first-order logic). Languages like $(ML)^2$ [8], KARL, or MoMo [9] use the KADS-I model of expertise (cf. [10]) as conceptual model for their language constructs. In Europe the KADS-I model of expertise is widely accepted as the appropriate conceptual framework for the specification of knowledge-based systems. The languages differ in whether they aim more at formalization or operationalization of a model of ex

2.3 Reuse of Problem-Solving Methods by Formal Methods

Currently, reuse by problem-solving methods and reuse by Very-high-level languages converges. KADS-I proposed a set of standardized elementary inference actions. [11] formally describes this set of elementary inference actions in $(ML)^2$. The granularity of these reusable blocks is less or equal to the above mentioned problem-solving mechanisms. A second example is the specification of several problem-solving methods in KARL like the board-game method, chronological backtracking, cover-and-differenciate, propose-and-exchange, etc.
These formal descriptions of reusable components have the following advantages compared to pure informal descriptions by natural language and graphics: Natural-language documents have an ambiguous and vague semantics; there is a high cognitive distance between the specification and the implementation, i.e. the specification is not an appropriate means for defining the behaviour of the implementation; as the specification is not executable, there is little support in evaluating it; and as the specification is informal, there is no support in verifying its correctness and completeness by formal methods.
Secondly, formal descriptions of reusable componentResearch which aims at improving the usability of the mechanism library and the formal specification languages approach by their combination will have to address two main topics:
Study of the formal semantics of the languages: The selection and modification of problem-solving methods or mechanisms can be supported by formal reasoning based on the semantical properties of the specifications of the methods or mechanisms in these languages. It requires further research to decide which of the several declarative semantics for formal specification languages is best suited for that purpose. $(ML)^2$ and KARL both use dynamic logic for specifying procedural knowledge but differ significantly in their way of integrating this with the specification of static knowledge. A further possibility is exemplified by the language DESIRE [12] which employs partial temporal logic as semantics.
Study of the formal properties of the problem-solving methods: The formal specification of a problem-solving method is the precondition for formally supporting its reuse. The complexity of the search process for appropriate problem-solving methods according to a given formal description of a task

(cf. [13]) can be significantly reduced by deriving pre- and postconditions from these for

References

[1] D. Fensel, "The Knowledge Acquisition and Representation Language KARL," Tech. Rep. Ph D. thesis, University of Karlsruhe, Institut fuer Angewandte Informatik und Formale Beschreibungsverfahren, 1993.

[2] C. W. Krueger, "Software Reuse," *ACM Computing Survey*, vol. 24, no. 2, pp. 131–184, June 1992.

[3] J.-M. David, J.-P. Krivine, and R. Simmons, *Second Generation Expert Systems.* Berlin: Springer-Verlag, 1993.

[4] J. Angele, D. Fensel, D. Landes, S. Neubert, and R. Studer, "Model-based and Incremental Knowledge Engineering: The MIKE Approach," in *J. Cuena (ed.), Knowledge Oriented Software Design, IFIP Transactions, A-27*, (Amsterdam), North-Holland, 1993.

[5] W. Clancey, "Heuristic Classification," *Artificial Intelligence*, vol. 27, pp. 289–350, 1985.

[6] J. Breuker, B. Wielinga, M. Someren, R. de Hoog, G. Schreiber, P. de Greef, B. Bredeweg, J. Wielemaker, and J.-P. Billault, "Model-Driven Knowledge Acquisition: Interpretation Models," Tech. Rep. ESPRIT project P1098, report, University of Amsterdam, 1987.

[7] M. Musen, "Overcoming the Limitations of Role-Limiting Methods," *Knowledge Acquisition*, vol. 4, no. 2, pp. 165–170, June 1992.

[8] F. v. Harmelen and J. Balder, "$(ML)^2$: A Formal Language for KADS Conceptual Models," *Knowledge Acquisitionn*, vol. 4, no. 1, pp. 127–161, March 1992.

[9] A. Voss, H. Voss, J. Walther, and T. Hemman, "Model-Driven Prototyping - Prototyping-Driven Modelling in Knowledge-Based Systems," in *Proceedings of Requirements Engineering RE93 - Prototyping -*, (Bonn), H. Zllighoven (ed.), Teubner Verlag, Stuttgart, 25-27 April, 1993.

[10] B. Wielinga, A. Schreiber, and J. Breuker, "KADS: A Modelling Approach to Knowledge Engineering," *Knowledge Acquisition*, vol. 4, no. 1, pp. 5–53, March 1992.

[11] M. Aben, "CommonKADS Inferences," Tech. Rep. ESPRIT Project P5248 KADS-II, KADS-II/M2/TR/UvA/041/1.0, University of Amsterdam, 1993.

[12] I. A. van Langevelde, A. W. Philipsen, and J. Treur, "Formal Specification of Compositional Architectures," in *Proceedings of the 10th European Conference on Artificial Intelligence (ECAI92)*, (Vienna, Austria), August 3-7,1992.

[13] H. Akkermans, B. Wielinga, and G. Schreiber, "Steps in Constructing Problem Solving Methods," in *Knowledge Acquisition for Knowledge-Based Systems, 7th European Workshop, EKAW93*, (Toulouse and Caylus, France), Lecture Notes in AI, no 723, Springer-Verlag, Berlin, September 6-10, 1993.

[14] S. Marcus, *Automating Knowledge Acquisition for Experts Systems.* Boston: Kluwer Academic Pub., 1988.

[15] F. Puppe, *Systematic Introduction to Expert Systems: Knowledge Representation and Problem-Solving Methods.* Berlin: Springer-Verlag, 1993.

3 Biography

Since 1989 Dieter Fensel is working as a scientist at the University of Karlsruhe. He holds a Diploma in computer science from the Technical University of Berlin and a Diploma in social science from the Free University of Berlin. He received his Ph.D. in economic science from the University of Karlsruhe in 1993. His thesis was concerned with the development of the Knowledge Acquisition and Representation Language KARL. He developed the modelling primitives and the formal (i.e., declarative) semantics of KARL. Currently, he is working *on reuse of formal specifications and use of formal specifications for reuse.* Further activities are concerned with the development and application of machine learning algorithms.