Adaptation of Generic Models in Model-driven Knowledge Acquisition

Klas Orsvärn
Swedish Institute of Computer Science
Box 1263, 164 28 Kista, Sweden
E-mail: klasorsv@sics.se, URL: http://www.sics.se/~klasorsv/

Abstract

Knowledge level modelling of knowledge-based systems is commonly facilitated with a library of generic models, that are specialized for classes of applications. The paper discusses guidelines for adapting such generic models, in particular task decompositions, to a given application. In this field of adaptation of reused knowledge structures, the kind of case for which reuse and adaptation is made is very complex and open – the development of a knowledge-based system. In the current state-of-the-art, the process cannot be automated, but must be controlled by a human knowledge engineer. Therefore, the results so far are informal guidelines.

Introduction

Our research on *Adaptation of Knowledge for Reuse* is done within the field of model-driven knowledge acquisition, where the major approaches today include COMMET (Steels 1990), CommonKADS (Schreiber *et al.* 1994), PROTÉGÉ-II (Puerta *et al.* 1992), and Spark-Burn-Firefighter (Yost *et al.* 1994). This research field addresses the question of how to support knowledge acquisition for development of knowledge-based systems (KBS), and the central idea in model-driven knowledge acquisition is to use a library of generic models, which are specialized for certain tasks and/or domains.

A common kind of generic model is a task decomposition method, which decomposes a task into subtasks that are simple enough to be solved directly by using domain models. Other generic models may also contain mechanisms for the necessary inferences over domain models. A generic model is based on assumptions about the task, the environment, the domain, and about available domain models containing application specific knowledge. Domain models describe objects and their relationships that the KBS has to reason about. The assumptions behind the generic models are used as criteria for selecting which one of them to use in a given application. The generic model prescribes which additional knowledge is needed in the application, in the form of domain models, which is used to drive the knowledge acquisition process. Generic models can be developed by generalizing from previous applications, or by more theoretical research on how to solve a given class of problems.

In the ideal case, knowledge acquisition is simply a matter of selecting a suitable generic model, and acquiring the domain knowledge required by the generic model. However, in practice it is very often the case that the generic model has to be adapted to the particular features of the application. In the current state-of-the-art, this adaptation has to be done by hand, and no guidance has previously been given in any of the approaches to model-driven knowledge acquisition.

Our research aims to fill this gap, specifically within the CommonKADS approach. So far, the support takes the form of written guidelines for the adaptation process (Orsvärn, Olsson, & Hassan 1994; 1995) for use by a knowledge engineer. The guidelines are mainly based on our own experiences of knowledge modelling with generic models (e.g. (Orsvärn 1992)). The library in CommonKADS contains many different kinds of generic modelling components (Breuker & Van de Velde 1994), but our guidelines only address the adaptation of a kind of generic model called interpretation models, which corresponds to what we have called a task decomposition method above.

We will first give an overview of the modelling process, decomposed into four activities, which is used as a framework for the guidelines. This overview will also indicate the specific features of adaptation in this task and domain. After that, we will describe guidelines for the two activities that are most central in adaptation of generic models. After that, we will briefly describe the empirical evaluation that has been made, draw some conclusions and indicate future work.

The Select-and-Modify Process

The development process is decomposed into the following four activities, which are illustrated in figure 1: **Select-IM**: Select an interpretation model (IM), on the basis of selection critera called task features in CommonKADS (Aamodt *et al.* 1992). **Evaluate-IM**: Investigate whether the selected interpretation model

is suitable in the application, or whether it is necessary to modify it. **Modify-IM**: Modify the interpretation model, to match the modification requirements identified in the evaluation. The result of the modification is again evaluated. **Domain KA**: After succcessful evaluation, acquire the domain knowledge that is required by the interpretation model.

It should be noted that although this is in the ideal case a directed flow of activities, as shown in figure 1, there will in practice often be iteration in the process, e.g. because it is very hard to capture all modification requirements in the evaluation.

At this level of description, the process of model-driven knowledge acquisition is very similar to that of case-based reasoning, or other reasoning methods where a candidate solution is first selected from a store, and then adapted to fit the current situation. A distinguishing feature of model-driven knowledge acquisition, in the current state-of-the-art, is that the process is not automated. A human (knowledge engineer) is in control of the process, although supported by software tools. The main reason for this is that the kind of case for which reuse and adaptation is made is a very complex one – the development of a knowledge-based system. Compared to the major classes of case-based reasoning, this makes it most similar to case-based design. Furthermore, it corresponds to design in an open world (Hinrichs 1992):

- The generic models are not selected only on the basis of selection criteria. The knowledge engineer may use any kind of evaluation to make a selection. In current libraries, selection criteria for a generic model are not sufficient conditions for applicability, but rather only the most distinguising necessary conditions. Furthermore, the knowledge engineer usually has no complete description of requirements in the current case (i.e. the current application).
- Since there is usually no complete description of requirements, the need for adaptation is often not discovered as violation of an explicit requirement, but rather because the generic model does not work on some test cases. However, failure on test cases often reflects violation of a previously tacit requirement, which should be made explicit.

Guidelines for evaluation

This section will describe guidelines for the evaluation activity in figure 1.

A cornerstone of the guidelines is that the modification requirements should be carefully investigated, and made explicit, before making the modifications. The modification requirements are those distinguishing features of the current application that explain why the selected generic model is not suitable. This recommendation may sound all too obvious. However, it is most often not practised, except in a tacit and incomplete way. The adaptation process is hard to control, since modifications interact, in the sense that a modification made to address one discrepancy, between the selected model and the current application, may be destroyed by a modification made later to address another discrepancy. The resulting modifications are often *ad hoc*, which makes the model hard to understand and maintain when requirements change in the future.

In order to handle the interactions between modifications, it is important to identify as many modification requirements as possible before making the modifications. As will be discussed later, the interactions can be handled by planning the sequence of modifications, and by addressing interacting requirements simultaneously in the same modification.

There are two dimensions to consider, when trying to identify as many modification requirements as possible:

- Identify as many as possible of the *discrepancies* between the selected generic model and the current application. Discrepancies may for example appear as difference in behaviour, between the generic model and the desired behaviour, e.g. protocols of expert behaviour. The search for discrepancies can, for example, be made by simulating the generic model with test cases (which requires that partial domain models are constructed), or by walkthrough of the model with domain experts.
- Identify as many as possible of the *modification requirements*, i.e. distinguishing features of the current application, that in some sense cause the above mentioned discrepancies.

The modification requirements behind a discrepancy are detected by explaining why the selected model is not suitable, and by phrasing the explanation in terms of requirements. This is not entirely easy to do. The main reason is that people often have difficulties justifying their beliefs. A contributing reason for this is that they are not used to distinguishing between requirements and other justifications, e.g. operational ones, like "I always do this before that". This difficulty can be mitigated somewhat by using a predefined categorization of requirements, and an inventory of common requirements, which gives abstract and concrete examples of what to look for. Such a framework has been developed in CommonKADS (Aamodt *et al.* 1992), for indexing reusable components in the library. The requirements in this framework are called task features, and describe properties of the real world, and domain knowledge requirements. Although the kinds of requirements that are used for indexing reusable components are in practice not always the same as those that necessitate modification of retrieved components, we have found it fruitful to use this framework for guidance in identifying modification requirements.

In the task feature framework, there are four main categories of task features, three of which we have found especially useful:

Task purpose This includes the goal of the task, and

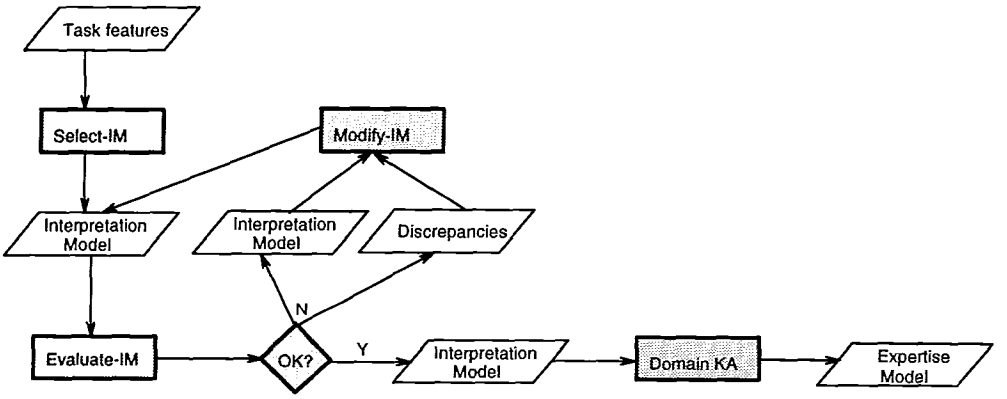

Figure 1: An overview of the activities involved in model-driven knowledge acquisition. (The generic model is here a so called interpretation model.)

the input and the output of the task. An example feature could be that the set of observations input to a specific diagnostic KBS is very large.

Task domain This is the set of abstract or physical objects that the task is about, i.e. that are the subject of the task. An example feature could be that the object (of e.g. diagnosis) is an electronic device.

Task environment This is the part of the task reality that the agent performing the task is interacting with. An example feature could be that the user is providing discriminating observations, and the cost of data-gathering should be minimized.

These three categories in turn have a hierarchy of sub-categories, which is partly illustrated in figure 2. The rest of the hierarchy is found in (Aamodt *et al.* 1992).

It would be very valuable to find guidelines for associating classes of discrepancies with classes of requirements. One example of such an association is based on a larger category of task features that we call task setting features. The term task setting refers to the reality in which the agent, whose expertise we are modelling, is (or will be) working to solve its task, i.e. external to the agent. The task setting includes the task assigned to the agent, the environment that the agent will be working in, and the domain that the agent has to reason about, but it does not include properties of the agent itself, such as the nature of its knowledge. The notion of task setting feature resembles the notion of functional requirement in the field of Requirements Engineering.

The guideline is that a significant behavioural discrepancy, i.e. a difference between the behaviour of the selected generic model and the desired behaviour in the application, should be due to task setting features. As mentioned above, a common kind of non-functional requirement imposed by generic models is domain knowledge requirements. Hence, it is natural to explain a discrepancy by violations to domain knowledge requirements. Suppose we conclude that a difference in behaviour between a generic model and the desired behaviour in a specific diagnosis application is due to difference in domain knowledge, e.g. that the generic model is based on a structure/behaviour device model, whereas the domain expert in the application appears to be using a causal model. Using the guideline, we can see that the use of a causal model is not a sufficient modification requirement, since a significant behaviour discrepancy must be due to a task setting feature. We may then discover that the real requirement is to minimize the total cost of observations to reach a final diagnosis, and fault models can be used to select the most likely hypothesis. This gives additional guidance in modification, since it is then apparent that the requirements can be satisfied without replacing the use of structure/behaviour models with causal fault models, but also by complementing the use of structure/behaviour models with fault models.

A general guideline for determining whether all the modification requirements behind a discrepancy have been made explicit, is to consider whether any modification satisfying the modification requirements would eliminate the discrepancy. Or to rephrase the question: will a solution in another application with the same requirements be a solution in this application as well? If not, there must be additional distinguishing requirements of the current application that need to be made explicit.

When requirements are made explicit, there is a basis for determining the necessity of adapting the selected model to satisfy them, since it is often the case that requirements are desirable but not vital.

Guidelines for modification

This section will describe guidelines for the modification activity in figure 1.

An interpretation model often consists of a hierarchy of modular task decomposition methods. At the top, a high level method decomposes the task assigned to

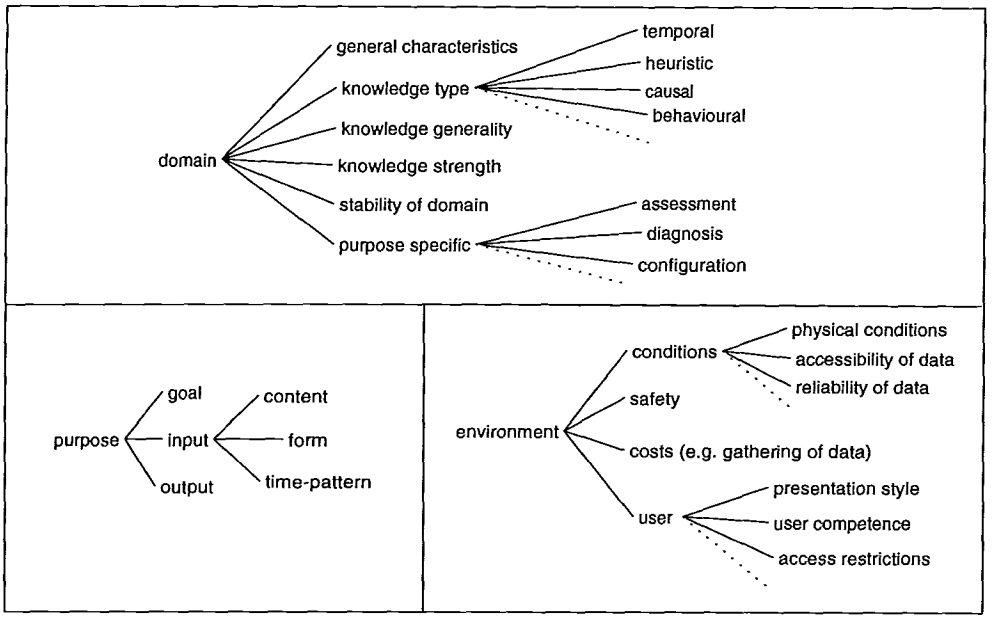

Figure 2: Selected parts of the hierarchies of sub-categories associated to three of the main task feature categories: domain, purpose, and environment.

the KBS into subtasks. These subtasks are in turn decomposed further by other methods into lower level subtasks, until a level is reached where the subtask can be achived directly using domain models.

With such generic models, the hierarchical decomposition can be used to determine where to make modifications, i.e. for localizing blame (Hinrichs 1992). This is done by determining for each task decomposition whether it is consistent with the modification requirements. If not, it should also be possible to determine which requirements are violated in this task decomposition. Figure 3 shows an example of such an analysis, from a case-study (Orsvärn 1995) of using these guidelines for reverse engineering of an existing diagnosis application, using a library of task decomposition methods for diagnosis tasks (Benjamins & Jansweijer 1994). An example of a violated task feature is that "knowledge exists for ranking hypotheses according to probability, based on association to specific symptoms". The decompositions of the tasks diagnosis, hypothesis discrimination, and select hypothesis, address the task feature to minimize the total cost of observations to reach a diagnosis. They need to use all available to knowledge to this end, e.g. the associations mentioned in the violated task feature. However, on inspection it turns out that the current set of observations, including symptoms, is not input to the hypothesis discrimination subtask of the top-level diagnosis tasks. So the associative knowledge cannot be used. This means the top-level decomposition violates the example task feature. The same holds for the lower level decompositions of the tasks hypothesis discrimination and select hypothesis.

This kind of assignment of modification requirements to parts of the generic model substantially simplifies the task of planning the modifications to handle interactions between them, to avoid backtracking in the modification process. One heuristic is to begin by modifying decompositions high up in the hierarchy. This is especially useful if all modification requirements have been detected in the evaluation. If we begin by making changes to decompositions of low level subtasks, there is a high risk that these subtasks will change later when higher level decompositions are modified, which may destroy the lower level modifications. Another heuristic is that all modification requirements pertaining to one task decomposition, should be addressed simultaneusly in the same modification.

An important reason why discrepancies should be described in terms of requirements, rather than in terms of properties of the generic model that needs modification, is that the latter description may no longer be applicable after making modifications to eliminate other discrepancies.

An important reason for using task features to describe requirements is that they can provide guidance in the modification activity. For example, if we find a behaviour discrepancy in a diagnosis application, which shows that the desired troubleshooting strategy is different from that which is embodied in the selected generic model, the work to understand and define the desired troubleshooting strategy would be facilitated by knowing that it is based on a single-fault assump-

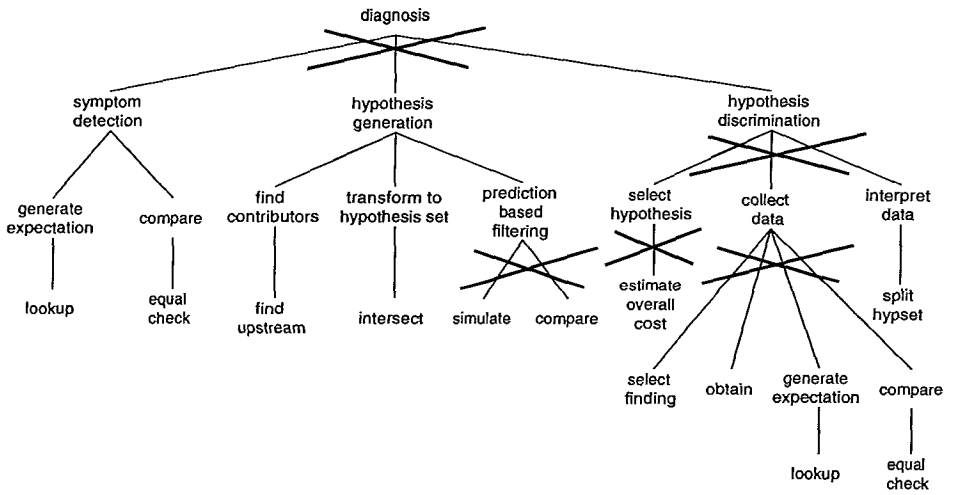

Figure 3: The figure illustrates which parts of the generated generic model needed to be modified, in case-study of modelling a diagnosis application using a library of task decomposition methods for diagnosis tasks. The sub-task branches are crossed out, for those task decompositions that were inconsistent with the modification requirements.

tion (a task feature of the category called purpose specific domain feature). This would also reveal relevant limitations of the modification.

When the modification requirements are explicit, there is also a potential for using a repertoire of stereotypical bugs, with associated modifications. A difficulty here is that the required modification always depends on what the selected generic model looks like. We have given some examples of such stereotypes, for task features that often cause modifications in libraries of task decomposition methods (Orsvärn, Olsson, & Hassan 1994). However, more work on this topic is needed.

Approaches to automated design adaptation often make use of replacing components that are to blame with other predefined components that satisfy the requirements, e.g. replacing lasagne with spinach lasagne to make a meal vegetarian (Hinrichs 1992). In model-driven knowledge acquisition, it is most often necessary to make structural modifications. But with better libraries, i.e. the selected generic model is appropriately structured, and based on components that are organized in such a way that alternative components can be identified, it is likely that substitution could be used more often.

Another difficulty in modification is that requirements may be conflicting, which calls for trade-off and compromise solutions. Johan Vanwelkenhuysen has given examples of concrete guidelines for compromise solutions to specific conflicting requirements in diagnosis applications (Vanwelkenhuysen 1995).

In a manual adaptation process like this one, it is very valuable to record design decisions and their rationale, so that developers and maintainers later in the system's life-cycle can understand what motivated the specific solutions in the system, e.g. so that they do not accidentally make modifications that violate the original requirements. This can be facilitated in the proposed process, since modification requirements are made explicit and attributed to parts of the generic model and the corresponding model after modification.

Evaluation of the Guidelines

The guidelines described in this paper were first tested by preparing a detailed example scenario (Orsvärn & Wells 1994) of developing an expertise model for a task of software assessment according to the guidelines. That model was originally developed using a select-and-modify approach, but without any guidelines. The scenario is a reconstruction of the process, which naturally deviates substantially from the original process. The main difference is the explicit introduction of the evaluation activity and the description of discrepancies in terms of task features. Although the scenario is mainly intended to illustrate the use of the guidelines, it is based only on real facts of the case, and it supports our belief that a model can be developed according to the guidelines. Since the scenario is a reconstruction, it is not possible to draw any definite conclusions from it about the benefits of the guidelines, but it clearly indicates that the guidelines support developers by providing a rigorous method.

Later, the guidelines have also been applied in a case-study (Orsvärn 1995) of knowledge modelling using a library of task decomposition methods methods for diagnosis tasks (Benjamins & Jansweijer 1994). The target of knowledge modelling was reverse engineering of an existing diagnosis application. The analysis illustrated in figure 3 was a big help in controlling the complex task of characterizing the discrepancies and

making the necessary modifications, in particular to reduce the need for backtracking. But it still remains to be shown that the guidelines can be applied by others than the authors.

Conclusions and Future Work

Although these guidelines are quite informal, we claim that they provide guidance for knowledge engineers in the complex process of adapting a selected generic model to an application. The two case-studies mentioned above seem to support this claim, although no definite conclusions can be drawn yet.

There is clearly a need to provide stronger support for the modification activity. The present guidelines are focused on identifying the requirements for modification, but more could be said about how to find the appropriate modification. In a related work, (Schreiber & Wielinga 1993) described examples of how to drive refinement of a task decomposition in a seemingly systematic way by task features. We believe that other modifications (structural change rather than refinement) can be described in a similar way, as generalization followed by the same kind of refinement. If the principles behind such a systematic adaptation process could be articulated, it would be a great potential for guidance.

We have found that the task feature framework of CommonKADS (Aamodt *et al.* 1992) provides guidance in characterizing the requirements of modifications, but we believe it would be possible to provide stronger support by extending that framework, e.g. the classification and inventory of task features (the notion of task setting feature, introduced here, is one example). A comprehensive task feature framework is also needed to build better libraries (note that the CommonKADS task feature framework was hardly used at all in the CommonKADS library (Breuker & Van de Velde 1994)). This brings us to another interesting prospect for the future; to extend these guidelines to cover how to incorporate new generic models into the library, i.e. models that have been created by modifying a selected generic model in an application.

In the work towards the above mentioned extensions to our guidelines, we will attempt to integrate results from other fields of AI, which address the issue of adapting previously developed solutions to new situations, into our guidelines, e.g. from the field of case-based design. Similarly, we hope that our analysis of the adaptation process in model-driven knowledge acquisition, and the specific guidelines, that we have presented here, can also be a source of inspiration for researchers in these other fields.

The case-study mentioned above (Orsvärn 1995), of knowledge modelling using a library of task decomposition methods methods for diagnosis tasks (Benjamins & Jansweijer 1994), also lead to identification of a number of principles that should be satisfied by libraries of generic task decomposition methods, in order to reduce probability and difficulty of having to modify a generic model in a given application. This indicates a close relationship between the problem of adapting a generic model to an application, and the problem of creating good generic models. The generic models are better, the less modifications are necessary in order to use them.

Acknowledgments Hesham A. Hassan and Olle Olsson have assisted in the development of these guidelines. Bob Wielinga and Steve Wells provided valuable feedback on the original guidelines report, and Johan Vanwelkenhuysen has also given valuable comments on this work. The work was to a large extent performed within the KADS-II project, which was partially funded by the CEC as Esprit project 5248.

Appendix: Describing the contribution

A. Reasoning framework:

1. *What is the reasoning framework?* The contribution is an informal description of an adaptation process performed by humans, with informal guidelines. It is not yet set within an automated reasoning framework.
2. *What benefits can be gained that motivated using adaptation? reuse?* The use of generic (reusable) models to drive knowledge acquisition can save a lot of work in the development of a KBS, and lead to better quality solutions. This often holds also when the best generic model is not quite suitable, i.e. when adaptation is necessary.
3. *What are the specific benefits and limitations of your approach?* The approach is very general within the scope of model-driven knowledge acquisition with task decomposition methods, and gives guidance in this process. The approach does not assume that requirements are completely specified, neither for the case (the application) nor for the stored generic models. However, the guidance is limited in strength, e.g. no strong guidance is given for determining which modifications to make.
4. *What invariants does it guarantee?* Since a human is in charge of the process, we cannot guarantee any invariants.
5. *What are the roles of adaptation, knowledge, and reuse in your approach?* The knowledge that is reused and adapted is problem solving knowledge for KBS, more specifically task decomposition methods, as opposed to domain knowledge. It is assumed that a library of reusable generic models exists. Retrieval is done by matching task features associated with the generic models with actual task features in a given application. Adaptation is a matter of adapting a generic model of problem solving knowledge to satisfy all requirements (which often are not known from the start) of the given application.

B. Task:

1. *What is the task? The domain?* The task and domain is development of a knowledge level model of a KBS.

2. *What are the inputs?* The initial input is a (incomplete) description of the application requirements in terms of task features.

3. *What are the outputs?* The output is a knowledge level model of a KBS.

4. *What constraints are on the outputs?* The output knowledge level model should satisfy the real requirements in the application, of which some are input initially.

5. *Are there characteristics of the domain that the method requires, relies on, or exploits in some way?* If the reused generic model is modular (hierarchical task decomposition) this is used to determine which part to modify in adaptation.

C. Evaluation:

1. *What hypotheses were explored?* Our hypothesis is that requirements analysis is a key to success in this process, and this is the issue we have focused on.

2. *What type of evaluation?* The task we address is very complex, so empirical evaluation is very expensive. We have so far described two scenarios of using the guidelines to develop two models of real KBS. These show that the guidelines are feasible to use, but no definite conclusions can yet be drawn empirically about the benefits.

3. *What comparisons were made with other methods?* Another paper (Orsvärn, Olsson, & Hassan 1995) compares with related reuse-based work on the same task, but noone has to our knowledge directly addressed the adaptation process before. No comparison has been made with approaches not based on reuse.

4. *How does the evaluation validate or illuminate your theory of adaptation of knowledge for reuse?* The scenarios illustrate how adaptation in our approach is intended to work.

5. *What are the primary contributions of your research?* The primary contribution is an analysis of the critical factors involved in adaptation of generic models in model-driven knowledge acquisition, and an outline of a principled adaptation process.

References

Aamodt, A.; Benus, B.; Duursma, C.; Tomlinson, C.; Schrooten, R.; and Van de Velde, W. 1992. Task features and their use in CommonKADS. ESPRIT Project P5248 Deliverable D1.5 KADS-II/T1.5/VUB/TR/014/1.0, Free University of Brussels & University of Amsterdam & Lloyd's Register.

Benjamins, R., and Jansweijer, W. 1994. Towards a competence theory of diagnosis. *IEEE Expert* 9(5).

Breuker, J., and Van de Velde, W., eds. 1994. *CommonKADS Library for Expertise Modelling: Reusable Problem Solving Components.* Amsterdam: IOS-Press.

Hinrichs, T. R. 1992. *Problem Solving in Open Worlds: A Case Study in Design.* Hillsdale, New Jersey: Lawrence Erlbaum Associates.

Orsvärn, K., and Wells, S. 1994. Modelling scenario: SAT expertise model. In Wielinga, B., ed., *Expertise Model Definition Document.* University of Amsterdam.

Orsvärn, K.; Olsson, O.; and Hassan, H. A. 1994. Guidelines for the Select-and-Modify approach. In Wielinga, B., ed., *Expertise Model Definition Document.* University of Amsterdam.

Orsvärn, K.; Olsson, O.; and Hassan, H. A. 1995. Guidelines for adapting an interpretation model in an application. In *Proc. Knowledge Engineering Forum 95*, number 903 in Arbeitspapiere der GMD, 95–106. Sankt Augustin, Germany: GMD.

Orsvärn, K. 1992. Adaptation of a task specific problem solving method to the features of an application. In *Proc. 7th Banff Knowledge-Acquisition for Knowledge-Based Systems Workshop*, 19–1 – 19–. SRDG Publications, University of Calgary.

Orsvärn, K. 1995. Case-study in knowledge modelling with Benjamins' library of diagnosis methods. In *Proc. Knowledge Engineering Forum 95*, number 903 in Arbeitspapiere der GMD, 83–94. Sankt Augustin, Germany: GMD.

Puerta, A.; Egar, J.; Tu, S.; and Musen, M. 1992. A multiple-method knowledge-acquisition shell for the automatic generation of knowledge-acquisition tools. *Knowledge Acquisition* 4:171–196.

Schreiber, A. T., and Wielinga, B. J. 1993. Model construction. In Schreiber, A. T.; Wielinga, B. J.; and Breuker, J. A., eds., *KADS: A Principled Approach to Knowledge-Based System Development.* London: Academic Press. 93–118.

Schreiber, A. T.; Wielinga, B. J.; Akkermans, J. M.; Van de Velde, W.; and de Hoog., R. 1994. Commonkads: A comprehensive methodology for kbs development. *IEEE Expert* 9(6).

Steels, L. 1990. Components of expertise. *AI Magazine* 11(2):29–49.

Vanwelkenhuysen, J. 1995. Embedding non-functional requirements analyses in conceptual knowledge system designs. In *Proc. 9th Banff Knowledge-Acquisition for Knowledge-Based Systems Workshop.*

Yost, G.; Klinker, G.; Linster, M.; Marques, D.; and McDermott, J. 1994. The SBF framework, 1989-1994: From applications to workplaces. In Steels, L.; Schreiber, G.; and Van de Velde, W., eds., *Proc. EKAW'94*, number 867 in LNAI, 318–339. Springer-Verlag.