Proceedings of the 10th Banff Knowledge Acquisition for Knowledge-Based System Workshop (KAW´96), Banff, Canada, November 9-14, 1996.

[Nebel, 1996] B. Nebel: Artificial intelligence: A Computational Perspective. In G. Brewka (ed.), *Essentials in Knowledge Representation*, 1996.

[O'Hara & Shadbolt, 1996] K. O'Hara and N. Shadbolt: The Thin End of the Wedge: Efficiency and the Generalized Directive Model Methodology. In N. Shadbolt (eds.), *Advances in Knowledge Acquisition*, LNAI 1076, Springer-Verlag, Berlin, 1996.

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

[Reif, 1992] W. Reif: The KIV-System: Systematic Construction of Verified Software, *Proceedings of the 11th International Conference on Automated Deduction*, CADE-92, Lecture Notes in Computer Science (LNCS), no 607, Springer-Verlag, Berlin, 1992.

[Reif, 1995] W. Reif: The KIV Approach to Software Engineering. In M. Broy and S. Jähnichen (eds.): *Methods, Languages, and Tools for the Construction of Correct Software*, LNCS 1009, Springer-Verlag, 1995.

[Smith & Lowry, 1990] D. R. Smith and M. R. Lowry: Algorithm Theories and Design Tactics, *Science of Computer Programming*, 14:305—321, 1990.

[Schreiber et al., 1993] A. Th. Schreiber, B. J. Wielinga, and J. A. Breuker (eds.): *KADS: A Principled Approach to Knowledge-Based System Development, vol 11 of Knowledge-Based Systems Book Series*, Academic Press, London, 1993.

[Schreiber et al., 1994] A. TH. Schreiber, B. Wielinga, J. M. Akkermans, W. Van De Velde, and R. de Hoog: CommonKADS. A Comprehensive Methodology for KBS Development, *IEEE Expert*, 9(6):28—37, 1994.

[Steels, 1990] L. Steels: Components of Expertise, *AI Magazine*, 11(2), 1990.

[ten Teije, 1997] A. ten Teije: *Automated Configuration of Problem Solving Methods in Diagnosis*, PhD thesis, University of Amsterdam, Amsterdam, NL, 1997.

[Terpstra et al., 1993] P. Terpstra, G. van Heijst, B. Wielinga, and N. Shadbolt: Knowledge Acquisition Support Through Generalised Directive Models. In M. David et al. (eds.): *Second Generation Expert Systems*, Springer-Verlag, 1993.

[van Heijst and A. Anjewerden, 1996] G. van Heijst and A. Anjewerden: Four Propositions concerning the specification of Problem-Solving Methods. In *Supplementary Proceedings of the 9th European Knowledge Acquisition Workshop EKAW-96*, Nottingham, England, May 14-17, 1996.

[Wielinga et al., 1995] B. J. Wielinga, J. M. Akkermans, and A. Th. Schreiber: A Formal Analysis of Parametric Design Problem Solving. In *Proceedings of the 9th Banff Knowledge Acquisition Workshop (KAW-95)*, Banff, Canada, January 26 - Feruary 3, 1995.

[Wirsing, 1990] M. Wirsing: Algebraic Specification. In J. van Leeuwen (ed.), *Handbook of Theoretical Computer Science*, Elsevier Science Publ, 1990.

[Bylander et al., 1991] T. Bylander, D. Allemang, M. C. Tanner, and J. R. Josephson: The Computational Complexity of Abduction, *Artificial Intelligence*, 49, 1991.

[Chandrasekaran et al., 1992] B. Chandrasekaran, T.R. Johnson, and J. W. Smith: Task Structure Analysis for Knowledge Modeling, *Communications of the ACM*, 35(9): 124—137, 1992.

[David et al., 1993] J.-M. David, J.-P. Krivine, and R. Simmons (eds.): *Second Generation Expert Systems*, Springer-Verlag, Berlin, 1993.

[de Kleer & Williams, 1987] J. de Kleer and B. C. Williams: Diagnosing Multiple Faults, *Artificial Intelligence*, 32:97-130, 1987.

[de Kleer, 1992] J. de Kleer, K. Mackworth, and R. Reiter: Characterizing Diagnoses and Systems, *Artificial Intelligence*, 56, 1992.

[Eriksson et al., 1995] H. Eriksson, Y. Shahar, S. W. Tu, A. R. Puerta, and M. A. Musen: Task Modeling with Reusable Problem-Solving Methods, *Artificial Intelligence*, 79(2):293—326, 1995.

[Fensel & Benjamins, 1996]D. Fensel and R. Benjamins: Assumptions in Model-Based Diagnosis. In *Proceedings of the 10th Banff Knowledge Acquisition for Knowledge-Based System Workshop (KAW´96)*, Banff, Canada, November 9-14, 1996.

[Fensel, 1995] D. Fensel: Assumptions and Limitations of a Problem-Solving Method: A Case Study. In *Proceedings of the 9th Banff Knowledge Acquisition for Knowledge-Based System Workshop (KAW-95)*, Banff, Canada, January 26 - February 3, 1995.

[Fensel et al., 1996] D. Fensel, H. Eriksson, M. A. Musen, and R. Studer: Developing Problem-Solving by Introducing Ontological Commitments, *International Journal of Expert Systems: Research & Applications*, vol 9(4), 1996.

[Fensel & Straatman, 1996] D. Fensel and R. Straatman: The Essence of Problem-Solving Methods: Making Assumptions for Efficiency Reasons. In N. Shadbolt et al. (eds.), *Advances in Knowledge Acquisiiton, LNAI* 1076, Springer-Verlag, 1996.

[Fensel & Groenboom, 1997] D. Fensel and R. Groenboom: Specifying Knowledge-Based Systems with Reusable Components. In *Proceedings of the 9th International Conference on Software Engineering & Knowledge Engineering (SEKE-97)*, Madrid, Spain, June 18-20, 1997.

[Fensel & Schönegge, 1997a] D. Fensel and A. Schönegge: Assumption Hunting as Development Method for Knowledge-Based Systems. In *Proceeedings of the Workshop on Problem-Solving Methods for Knowledge-based Systems at the 15th International Joint Conference on AI (IJCAI-97)*, Nagoya, Japan, August 23, 1997.

[Fensel & Schönegge, 1997b] D. Fensel and A. Schönegge: Specifying and Verifying Knowledge-Based Systems with KIV. In *Proceedings of the European Symposium on the Validation and Verification of Knowledge Based Systems EUROVAV-97*, Leuven Belgium, June 26-28, 1997.

[Fensel et al., 1997] D. Fensel, E. Motta, S. Decker, Z. Zdrahal: Using Ontologies For Defining Tasks, Problem-Solving Methods and Their Mappings. To appear in *Proceedings of the European Knowledge Acquisition Workshop (EKAW-97)*, Sant Feliu de Guixols, Catalonia, Spain, October 15-18,LNAI, Springer-Verlag, 1997.

[Harel, 1984] D. Harel: Dynamic Logic. In D. Gabby et al. (eds.), *Handbook of Philosophical Logic, vol. II*, Extensions of Classical Logic, Publishing Company, Dordrecht (NL), 1984.

[Klinker et al., 1991] G. Klinker, C. Bhola, G. Dallemagne, D. Marques, and J. McDermott: Usable and Reusable Programmin Constructs, *Knowledge Acquisition*, 3:117—136, 1991.

[Marcus, 1988] S. Marcus (ed.). *Automating Knowledge Acquisition for Experts Systems*, Kluwer Academic Publisher, Boston, 1988.

[Motta & Zdrahal, 1996] E. Motta and Z. Zdrahal: Parametric Design Problem Solving. In

worries that (1) methods specified for one problem type can be applied to other problems (requiring some renaming) and (2) not only the methods that are provided for a problem class in the library can be applied to it.

During the paper we used simple examples to make the paper easy to understand. The simple examples allowed us to present most of the details necessary to follow our arguments. However this does not at all imply that we designed our framework for block worlds. Currently we are applying our concept to the family of PSMs for parametric design that are developed by [Motta & Zdrahal, 1996]. It turns out that it is quite easy to scale up our approach. For example, propose & revise turns out to be a loop of two local searches applied in different contexts. One proposes extensions of design models (i.e., the propose step) and one revises design models in cases when they are incorrect. So simply put it is a loop of two instantiations of hill climbing. The different instantiations can be achieved by defining different successor relationships (one via extension and one via correction of models). A formalization of the problem definition of parametric design is already provided in [Fensel et al., 1997].

Acknowledgement. The paper conceptualizes work that was done together with Arno Schönegge on specifying and verifying problem-solving methods. Also I would like to thank Stefan Decker, Enrico Motta and Zdenek Zdrahal with whom I am currently experimenting in applying the ideas to parametric design and Rudi Studer and two anonymous reviewers for helpful comments.

References

[Akkermans et al., 1993] J. M. Akkermans, B. Wielinga, and A. TH. Schreiber: Steps in Constructing Problem-Solving Methods. In N. Aussenac et al. (eds.): *Knowledge-Acquisition for Knowledge-Based Systems*, Lecture Notes in AI, no 723, Springer-Verlag, 1993.

[Angele et al., 1996] J. Angele, D. Fensel, and R. Studer: Domain and Task Modelling in MIKE. In A. Sutcliffe et al. (eds.), *Domain Knowledge for Interactive System Design*, Chapman & Hall, 1996.

[Benjamins, 1995] R. Benjamins: Problem Solving Methods for Diagnosis And Their Role in Knowledge Acquisition, *International Journal of Expert Systems: Research and Application*, 8(2):93—120, 1995.

[Benjamins & Pierret-Golbreich, 1996] R. Benjamins and C. Pierret-Golbreich: Assumptions of Problem-Solving Method. In N. Shadbolt et al. (eds.), *Advances in Knowledge Acquisition*, Lecture Notes in Artificial Intelligence (LNAI), no 1076, Springer-Verlag, Berlin, 1996.

[Benjamins et al., 1996] R. Benjamins, D. Fensel, and R. Straatman: Assumptions of Problem-Solving Methods and Their Role in Knowledge Engineering. In *Proceedings of the 12. European Conference on Artificial Intelligence (ECAI-96)*, Budapest, August 12-16, 1996.

[Beys et al., 1996] P. Beys, R. Benjamins, and G. van Heijst: Remedying the Reusability-Usability Tradeoff for Problem-solving Methods. In *Proceedings of the 10th Banff Knowledge Acquisition for Knowledge-Based System Workshop (KAW´96)*, Banff, Canada, November 9-14, 1996.

[Breuker, 1997] J. Breuker: Problems in Indexing Problem Solving Methods. In *Proceeedings of the Workshop on Problem-Solving Methods during the IJCAI-97*, Japan, August 24, 1997.

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

[Benjamins, 1995], [Terpstra et al., 1993]. The former deal only with a very limited aspects of the methods as a method is essentially a description of *how* to achieve some goals. The latter assume that adapting a PSM to a given problem is an activity of decomposing a problem in subproblems and defining control over the solution of the subproblems (and recursively refining the subproblems). However it was often reported in the literature that different control regimens can be applied to solve the same problem and the same control regime can be applied to very different problems [Breuker, 1997]. Therefore, we think that adapting a control regime is neither the only nor the central point in adapting a PSM to a given problem. When defining the overall control schema (i.e., local search, branch and bound, etc.) one mainly refers to properties of the domain knowledge. For example, only when a useful successor relation is provided local search can be applied and A* can only be applied if a useful heuristic estimation function is provided. Besides this, they can be applied to any type of problem that can be solved by a search process. An important distinction between [Benjamins, 1995], [Terpstra et al., 1993] and our approach is that [Benjamins, 1995], [Terpstra et al., 1993] express a PSM immediately in problem-specific terms (like symptom detection, hypothesis generation, hypothesis discrimination, etc.) whereas we describe general algorithmic schemas that become instantiated to a specific class of problem via adapters. Therefore we can discuss these algorithmic schemas of PSMs independently from specific problems reflecting the fact that the same PSM (or better the same algorithmic schema) can be applied to different problem classes.

For organizing a library of PSMs we made the following proposals:

- Extending the library by including problem definitions and assumptions necessary to relate the competence of method with the problems. By including problem definitions in the library we can use them to select appropriate PSMs as proposed by [Breuker, 1997]. We argue for inclusion of assumptions within our library because they are as important as PSMs. Assumptions define the parts of the problem that cannot be solved by the PSM but must be assumed as given (either as domain knowledge or as problem restriction [Benjamins et al., 1996]). They are the complement of the PSM relative to a selected problem. A kind of library of assumptions for a specific type of problems is described in [Fensel & Benjamins, 1996].
- Using adapters to relate units of problems, PSMs and assumptions of different specificity to get rid of the usability-reusability trade-off of [Klinker et al., 1991].
- Defining two orthogonal dimensions for organising such a unified library: (1) the specificity of problems, PSMs and assumptions (see Fig. 8) and (2) the algorithmic schema used to derive the PSM (i.e., local search, branch and bound etc., see Fig. 1).[5] Current libraries of PSMs [Benjamins, 1995], [Breuker & Van de Velde, 1994], [Chandrasekaran et al., 1992], [Motta & Zdrahal, 1996], [Puppe, 1993] interweave these two dimensions. As a consequence the literature is full of

5. One reviewer was wondering whether abduction should be solved by local search like we illustrate in this paper. Especially he makes the point that there are many other search and optimization techniques. Actually this circumstance is the reason for the second dimension in our framework that allows different algorithm schemas as core of PSMs.

realizing an implementation and computing an actual solution. They define a part of the entire problem for which no implementation has to be provided. Therefore, the system can be realized more easily and in system runtime this part of the problem does not need to be computed.[4]

In general, there is no distinction between problem specification, assumption specification and the specification of the competence of a method. They are all specification units. Only their roles within the entire specification differ. That is, it is their context that creates their distinction:

- A *problem* defines a specifications that becomes "realized" by two other specification. One of these specifies the part that is actually solved (the competence of a PSM) and the other specifies the part that is assumed to be solved (an assumption specification).
- An *assumption* is a specification that has *no* realization at all. Its realization is *assumed* only.
- Finally, the *competence* of a PSM is a specification that is directly realized by a module that defines an operationalization. Again this operationalization relies on a specification defining its knowledge requirements and inferences. Their realization are either external to the specification or introduce a new level of hierarchical refinement by being formulated as new problems.

4 Conclusion, Related and Future Work

We have show how to use adapters for developing PMSs and for organizing a library of PSMs including problem definitions and assumptions. The development process of PSMs is viewed as a refinement process that:

- introduces ontological commitments used to characterize initial, intermediate and terminal states of the method;
- uses ontological commitments to specialize the state transitions of a method; and
- introduces assumption to bridge the gap between competence of a method and a problem definition.

All these refinement were achieved by adding adapters to existing elements. A number of authors [Beys et al., 1996][van Heijst and A. Anjewerden, 1996] have proposed that PSMs should be described not only in a domain-independent, but also task-independent way, so that they can become more broadly reusable. However, there is a known trade-off between usability and reusability [Klinker et al., 1991]. With our approach this dilemma disappears. PSMs can either be reused in their generic or more problem-specific variant. The latter does not modify the former but adds only an external description to it.

Existing approaches for developing PSMs either stop at the level of the competence of the methods [Akkermans et al., 1993], [Wielinga et al., 1995], [ten Teije, 1997] or view PSM development as process of hierarchically refining inference actions

4. See [O'Hara & Shadbolt, 1996] for a discussion of different dimensions of efficiency.

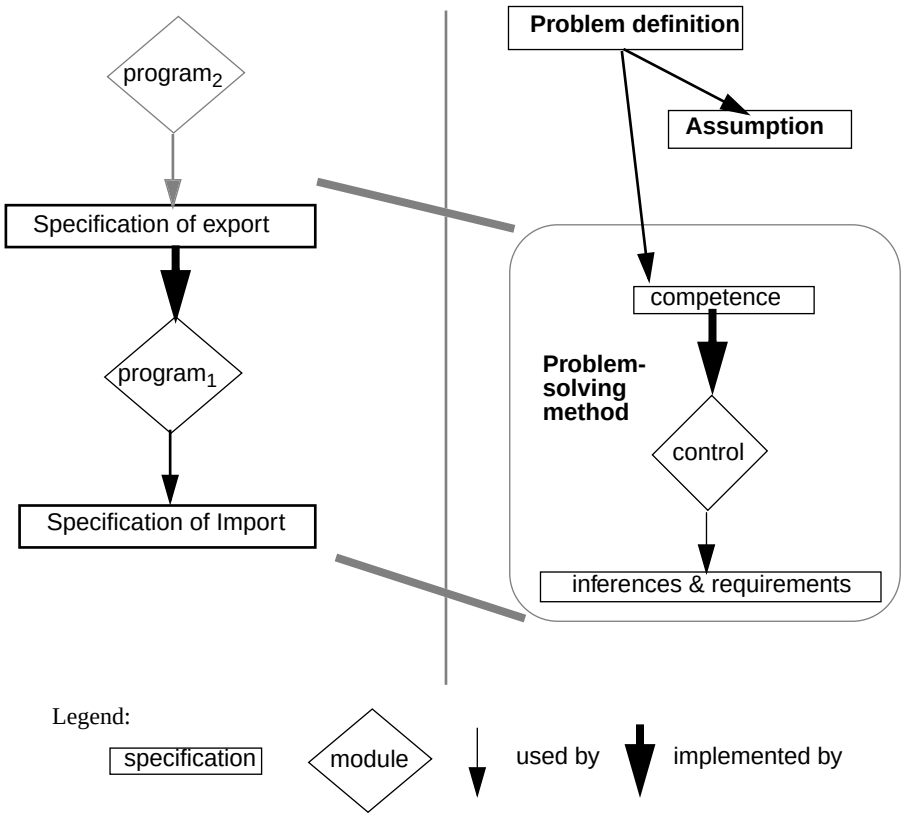

Fig. 9 Development graphs in KIV.

execution of imported operations (knowledge and inferences).

- Finally the competence of a PSM corresponds to the specification of the functionality of a module.

A distinction between typical development graphs in KIV and our specifications is introduced by problem definitions and assumptions. Usually in software engineering it is assumed that the problem that should be solved is identical with the functionality of the program. However most problems tackled with KBSs are inherently complex and intractable (cf. [Nebel, 1996], [Fensel & Straatman, 1996]). A PSM can only solve such problems with reasonable computational effort by introducing assumptions that restrict the complexity of the problem, or by strengthening the requirements on domain knowledge. Therefore, a specification of the problem independent from the specification of the competence as well as the specification of assumptions are introduced in our context. A method does not have the direct competence to solve the problem. Only when adding assumptions that limit the problem can this be guaranteed. The entire problem is therefore decomposed into a part that can be solved by the PSM and for which an operationalization is provided and a second part of which the solution is only assumed. This makes quite clear *how assumptions help to reduce the effort of*

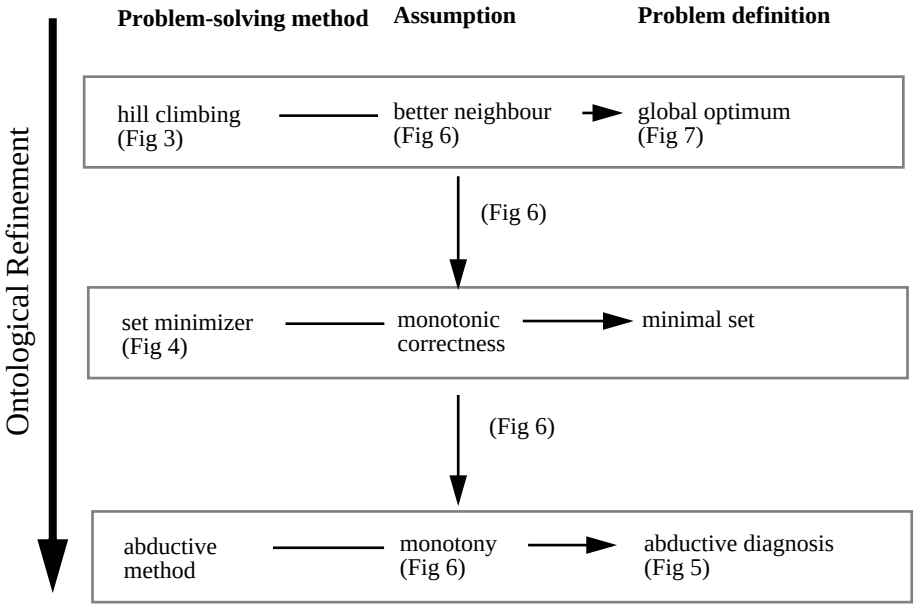

Fig. 8 Refining PSMs, assumptions, and problem definitions.

procedures provided in an export specification. It internally describes the algorithmic realization of these procedures. It uses, through an import specification, operations that are specified by other specifications (and realized by their modules). Usually a specification in KIV has the following pattern:

- A specification defines the functionality of operations that are imported by a module, i.e. by an implementation.
- A module imports some operations und uses them to implement new operations that are exported.
- A second specification defines the required functionality of the exported operations. Then a new implementation would use these operations as import.

The left side of Fig. 9 provides the structure of such a typical development graph. The single specifications are the rectangles in the graph and modules are modelled by the rhomboid units in the graph. Such a typical development graph corresponds with the specification and operationalization of a PSM (viewed at the right side of Fig. 9).

- The inference actions are defined in the import specification of the PSM. No algorithmic realization of the inference action is provided.
- The algorithmic realization of the knowledge requirements is also not part of the PSM because this aspect is assumed to be covered by the domain knowledge or by other agents of the entire problem-solving process. Therefore, its realization is beyond the scope of the specification of the PSM.
- The operationalization of a PSM corresponds to an implementation module. It describes how a specific competence can be achieved by defining control over the

3 An Integrated Library of Problem Definitions, Problem-Solving Methods and Adapters

So far, we have illustrated the stepwise refinement of PSMs. We started with a generic search strategy that became instantiated to hill-climbing that became later on refined to set minimizer that became later on refined to a method for abductive problems. In the same way we refined PSMs we can also refine problem definitions and assumptions necessary to link PSMs and problem definitions. In [Fensel & Schönegge, 1997a] we showed *that the monotony assumption is a problem-specific refinement of an assumption that is necessary and sufficient to prove that hill climbing finds a global optimum*. Hill climbing can solve the problem of finding a global optimum if we assume the *better neighbour assumption*, i.e. each entity that is not a global optimum has a better neighbour. Fig. 7 provides the definition of the corresponding problem and Fig. 6 of the better-neighbour assumption. Therefore, it is not only possible to refine PSMs but also problem definitions and assumptions. Fig. 8 summarizes our problem definitions, assumptions and methods. With simple extensions of the mappings of the two refinement adapters in Fig. 6 (by defining mappings for terms that were only used in problem and assumption definitions) we can use them to refine corresponding problem definitions and assumptions necessary to relate PSMs and problems as we did in section 2 for PSMs.

The dimension of this refinement is the ontological commitments made by problem definition, assumptions and PSMs. Notice that this specializations are kept separate via adapters. Therefore, it is always possible to reuse very specific or very generic entities from our library. We get an unified library of problem definitions, assumptions, and PSMs where the refinements have a *virtual* existence via adapters.

A second dimension of the library are the algorithmic schemas that are used to derive the PSMs. Instead of local search we can also choose branch and bound or A* as schemas that define the basic search algorithm of a method.

When organising such a library we have to go into the question of what are the differences between problems, assumptions, and PSMs and whether these differences have consequences in organising the library. We realized some significant properties of these elements when specifying them with KIV. KIV is designed for the specification and verification of programs. The entire specification of a system can be split into smaller and more tractable pieces. A specification defines the functionality of a program using algebraic specification techniques. In addition to specifications, KIV provides modules to describe implementations. A module defines a collection of

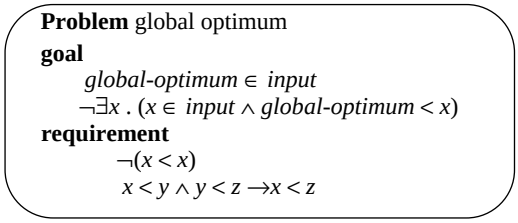

Fig. 7 The specification of the problem finding a global optimum.

PSM refinement adapter hill-climbing -> set-mimimizer

/* The input set must be correct. */

$correct(input)$

/* *select- start* must select the input set. */

$select\text{-}start(x) = \{x\}$

/* Successors are subsets that contain one element less.*/

$successor(x,y) \leftrightarrow \exists z\ .\ (z \in x \wedge y = x \setminus \{z\})$

/* We prefer smaller sets if they are still correct. */

$x < y \leftrightarrow correct(y) \wedge y \subset x$

PSM refinement adapter set-mimimizer -> abduction-method

$correct(x) = complete(x)$;

$input = \{h \mid h \text{ is } hypothesis\}$;

PSM refinement adapter abduction-method -> abductive diagnosis

$H_1 \subseteq H_2 \rightarrow expl(H_1) \subseteq expl(H_2)$

Assumption adapter hill climbing -> global optimum

$x \in input \rightarrow (\exists y\ .\ (y \in input \wedge successor(x,y) \wedge x < y) \vee \neg\ \exists z\ .\ (z \in input \wedge x < z))$

Fig. 6 The adapters.

average-case effort of finding all parsimonious and complete explanations with GDE.

2.4 Resume

We discussed four steps of the derivation of a refined PSM for abductive problems from a generic local search frame via adapters (cf. Fig. 6):

- hill climbing := PSM-refinement-adapter$_{\text{local search -> hill-climbing}}$(local search)
- set minimizer := PSM-refinement-adapter$_{\text{hill climbing -> set minimizer}}$(hill climbing)
- abductive method := PSM-refinement-adapter$_{\text{set minimizer -> abductive method}}$(set minimizer)
- abductive diagnosis :=
  Assumption-adapter$_{\text{abductive method -> abductive diagnosis}}$(abductive method)

The first adapter[3] refines mainly the definition of state transitions and introduces knowledge requirements that allows the definition of the competence of the method in terms of finding a local optimum. The second adapter refines the notion of states (entities) to sets and refine the definition of state transition via defining a successor relationship between sets. The third adapter adds some simple terminological mappings that express the method in terms of abduction. Finally, an assumption is added to guarantee that this methods achieves the goal as given by the problem definition.

3. For reasons of limited space we do not provide the formal definition of this adapter in the paper.

method it is easy to overcome what was viewed as the *usability/reusability trade-off* of PSMs [Klinker et al., 1991]. The original version of hill-climbing can be reused for different problems requiring different kinds of refinement. The combination of hill-climbing and the set-minimizer adapter can be used for problems that can be expressed in terms of minimizing sets. While this combined version is less reusable, it is much more usable for cases it can be applied to. For achieving a problem-specific variant of a method it is not necessary to change the method itself. Instead, a problem-specific adapter is added. These adapters can also be piled up (stacked) to increase the problem specificity of methods. We will show this in the following section where we adapt set minimizer to abductive diagnosis.

2.3 Abductive Diagnosis

The task of abductive diagnosis receives a set of observations as input and delivers a complete and parsimonious explanation (see e.g. [Bylander et al., 1991]). An explanation is a set of hypotheses. A complete explanation must explain all input data (i.e., observations) and a parsimonious explanation must be minimal (that is, no subset of it has the same or a greater explanatory power). The goal of the task is such a complete and parsimonious explanation (see Fig. 5).

The problem-specific refinement of set-minimizer is straightforwards (Fig. 6). The set of all hypotheses is the set that must be minimized and correctness is defined in terms of completeness. However we have to introduce assumptions to ensure that the local-minimal-set found by set-minimizer is a complete and parsimonious explanation. First, we have to require that the input of the method is a complete explanation. Second, based on the adapter we can prove that our method set-minimizer finds a set that is parsimonious in the sense that each subset that contains one element less is not a complete explanation. However, we cannot guarantee that it is parsimonious in general. There may exist smaller subsets that are complete explanations. The adapter has to introduce a new requirement on domain knowledge or an assumption (in the case that it does not follow from the domain model) to guarantee that the competence of the PSM is strong enough to achieve the goal of the task. The *monotony assumption* (cf. Fig. 6) is sufficient and necessary [Fensel & Schönegge, 1997a] to prove that the (global) parsimonious of the output of the PSM follows from its local parsimoniality. It defines a natural subclass of abduction. For example [de Kleer, 1992] examine their role in model-based diagnosis. The assumption holds for applications, where no knowledge that constrains fault behaviour of devices is provided or where this knowledge respects the *limited-knowledge-of-abnormal behaviour assumption*. This is used by [de Kleer & Williams, 1987] as a *minimal diagnosis hypothesis* to reduce the

Problem Abductive diagnosis
goal
$goal(x) \leftrightarrow complete(x) \wedge parsimonious(x)$
$complete(x) \leftrightarrow expl(x) = observables$
$parsimonious(x) \leftrightarrow \neg\exists x'.\ (x' \subset x \wedge expl(x) \subseteq expl(x'))$

Fig. 5 The specification of abductive diagnosis.

set minimizer adds additional ontological commitments used to characterize states of the search process.

- The successor relationship is hard-wired into set minimizer. A set is a successor of another set if it is a subset with one element less. The ontological commitment used to characterize states is used to refine the definition of state transitions.
- A preference on entities is only defined implicitly. Smaller sets are preferred if they are still correct.

Set minimizer describes only one of several possible problem-specific adaptations of hill climbing. Traditionally for each variant the specification has to be re-done, all termination and correctness proofs of the method have to be re-done, and the method has to be re-implemented. Our approach provides adapters as means to add the problem-specific refinement to hill-climbing, however keeping the adaptation separate. Therefore, the complete specifications, proofs, and implementations of hill-climbing can be reused. Only the problem-specific aspects have to be specified, proven and implemented by an adapter. Fig. 6 provides the definition of such an adapter. Its main proof obligations is to prove that the way the preference is defined fulfils the requirement on such a relation (cf. [Fensel & Schönegge, 1997b]).

By keeping the problem-specific refinement separate from the generic core of the

Problem-solving method Set minmizer
competence
output $\subseteq$ *input*,
correct(*output*),
$x \in output \rightarrow \neg correct(output \setminus \{x\})$
control
output := *set-minimizer*(*input*)
set-minimizer(*X*)
begin
successors := *generate*(*X*);
if $\neg \exists x . (x \in successors \wedge correct(x))$
then *output* := *X*
else
new := *select-successors*(*successors*)
set-minimizer(*new*)
endif
end
inference actions
generate
/* *generate* creates subsets that contain one element less.*/
$x \in generate(y) \leftrightarrow \exists z . (z \in y \wedge x = y \setminus \{z\})$
select-successors
/* A selected successor has to be correct. */
$\exists x . (x \in y \wedge correct(x)) \rightarrow$
$(correct(select\text{-}successors(y)) \wedge select\text{-}successors(y) \in y)$
input requirement
correct(*input*)

Fig. 4 The specification of set minimizer.

Problem-solving method Hill climbing
competence
output ∈ *input*
∃*x* . (*x* ∈ *input* ∧ *select-criterion*(*x* ,*input*) ∧ (*x* < *output* ∨ *x* = *output*))
¬∃*x* . (*successor*(*output*,*x*) ∧ *x* ∈ *input* ∧ *output* < *x*)
control
hill-climbing(*input*)
begin
current := *select-start*(*input*);
output := *recursion*(*current*)
end
recursion(*X*)
begin
successors := *generate*(*X*);
new := *select-successors*(*X*,*successors*)
if *goal*(*X*,*new*)
then *output* := *X*
else *recursion*(*new*)
endif
end
inference actions
select-start
/* *select-start* must select an element of input and uses a selection criterion. */
select-start(*x*) ∈ *x* ∧ *select-start*(*x*) ∈ *select-criterion*(*select-start*(*x*),*x*)
generate
/* *generate* selects input elements that are in successor relation with the current object.*/
x ∈ *generate*(*y*) ↔ *x* ∈ *input* ∧ *successor*(*y*,*x*)
select-a-best
/* *select-successors* selects the current object if no better successors exist or a successor if a better successor exists. In the latter case the selected successor must be better than the current object and there need not to be another successor that is better than the selected successor. */
¬∃*z* . (*z* ∈ {*y*} ∪ *y´* ∧ *select-successors*(*y*,*y´*) < *z*)
¬∃*z* . (*z* ∈ *y´* ∧ *y* < *z*) → *select-successors*(*y*,*y´*) = *y*
∃*z* . (*z* ∈ *y´* ∧ *y* < *z*) → *select-successors*(*y*,*y´*) ∈ *y´* ∧ *y* < *select-successors*(*y*,*y´*))
goal
goal(*x*,*y*) ↔ *x* = *y*
requirements
input requirement
∃*x*. (*x* ∈ *input* ∧ *x* ∈ *select-criterion*)
knowledge requirement
¬(*x* < *x*)
x < *y* ∧ *y* < *z* →*x* < *z*

Fig. 3 The specification of hill-climbing.

one element less. This method is obviously a local search method specialized for a specific type of problems. Set minimizer refines hill climbing with the following refinements:

- A generic state of hill climbing is characterized as a set in set minimizer. That is,

local search strategy can be described by an initialization and a recursion (see Fig. 2). Unlike to all other PSMs that will be discussed in this paper we cannot prove termination of this general algorithmic structure. Therefore, we do not regard it is a PSMs. Instead it is a starting point for deriving PSMs via refinement.[2]

2.1 Hill Climbing

A local search strategy is necessarily incomplete because it checks only a subset of the transitive closure of the successor relationship of an initial node. Fig. 3 provides a refined specification of a local search algorithm by introducing competence, control, inferences and the knowledge requirements of hill-climbing. It can be used to find a local optimum of a set of elements. The proof that the specified algorithm terminates and has the competence as defined is given in [Fensel & Schönegge, 1997a]. The specification of hill climbing will be the backbone of all our examples during the paper. All other refined versions will be achieved by combining hill climbing with adapters.

Hill climbing refines the generic local search strategy to an algorithm (1) for which termination can be proven and (2) the competence of the method being defined in terms of a preference and a successor relation. The output is a local optimal element in the sense that it does not have a successor that is preferred. However this method is very generic and can be applied to nearly any type of task with a successor relationship (a local search structure) and a preference relation. In the next step we will specialize this method to a method for minimizing sets. We will see that the preference relation need not to be defined explicitly. It can also be provided implicitly by ontological commitments used to characterize states and state transitions of the method.

2.2 Set Minimizer

[Fensel & Schönegge, 1997b] present a method called set minimizer that can be used to find a minimal but still correct subset of a given set. The specification of this method is provided in Fig. 4. Its only requirement is that the original set is correct. It returns a correct set which is locally minimal in the sense that there is no correct subset that has

Local search

current := *select-start*(*input*); *output* := *local search*(*current*)
local search(*X*)
begin
successors := *generate*(*X*);
new := *select-successors*(*X*,*successors*)
if *goal*(*X*,*new*)
then *output* := *X*
else *recursion*(*new*)
endif
end

Fig. 2 The specification of local search.

2. What we can prove is that if it terminates its output fulfils its goal.

commitments, it is necessary to add assumptions to close the gap between a method and a problem. In section 3 it is shown how these methods that search for a local optimum can be applied to problems that define a global optimum and how this leads to a refinement of problem definitions and assumptions similar to the refinement of PSMs. We draw conclusions on how to organise a unified library of methods, problem definitions and assumptions. Finally, section 4 provides conclusions, related work and outlines possible directions of future work.

2 Refining Problem-Solving Methods

We start by describing the development process of PSMs with a very generic search schema. During the following sections this schema will become refined to more concrete control of the search process and the states it searches through. However, we do not make a commitment to this top-down like development process. The process can start at any level and can take the direction of specialization or generalization because we provide a library containing these generic schemas and their adaptations. *Specialization* is achieved by adding an adapter to an existing PSM-adapter combination and *generalization* is achieved by deleting an adapter from an existing PSM-adapter combination.

[Smith & Lowry, 1990] present a theory of search algorithms to support the transformation of problem definitions into implementations. Fig. 1 shows their hierarchy of search methods providing local search as an instance of generate&test-like approaches working on local structures. The general algorithmic structure of a

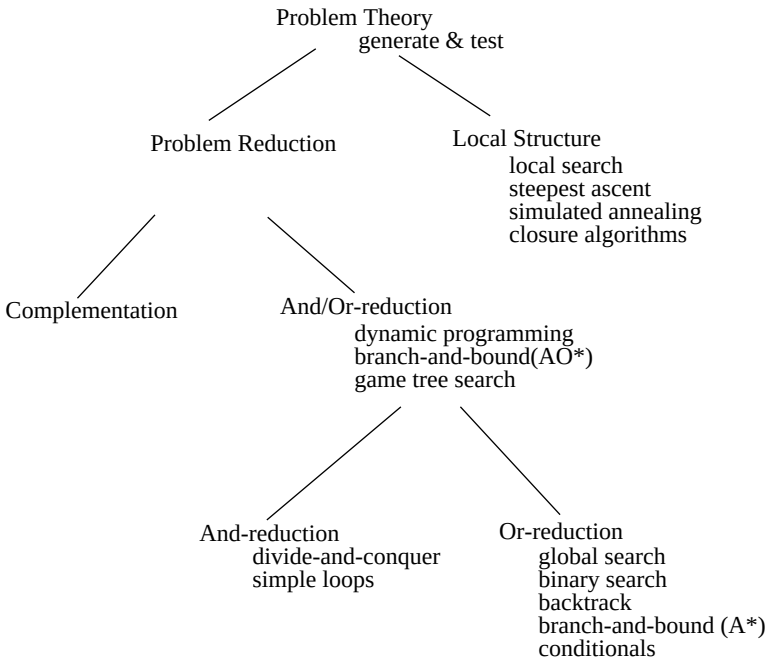

Fig. 1 Refinement hierarchy of algorithm theories.

PSMs, and domain knowledge. Building knowledge-based systems from reusable elements requires adapters that properly link these elements and adapt them to the application-specific circumstances. Because these elements should be reusable, they must abstract from application-specific circumstances and because they are specified independently from each other there is a need to introduce their mappings. Originally intended as glue that brings other elements together, we will give adapters a much more prominent role during this paper. They will play a central role in refining PSMs. Actually, a refined version of a PSMs is achieved by combining it with an adapter.

The stepwise introduction of adapters can be used to stepwise refine generic PSMs. This process can be used to *develop* and to *adapt* PSMs. More specifically, three processes are supported by our approach:

- the terminological structure of the states of a method can be refined by introducing ontological commitments;
- the refined terminological structure in describing states can be used to refine state transitions of a method; and
- assumptions can be introduced to link the competence of a method with problem definitions and domain knowledge.

This also leads to a entirely new organisation of a unified library providing problem definitions, PSMs and assumptions organised at different levels of refinement.

The technical machinery for specifying and verifying our examples is provided by the KIV approach (Karlsruhe Interactive Verifier) [Reif, 1995]. In KIV, the entire specification of a system can be split into smaller and more tractable pieces to support understandability and reuse of the different parts. Each *elementary specification* introduces a signature and a set of axioms. The semantics of such a specification is the isomorphic class of all algebras that satisfy the first-order axioms and that are generated by the operations indicated in the generation clauses (i.e., *loose* semantics is applied [Wirsing, 1990]). In addition to (elementary) specifications, KIV provides *module* to describe implementations in a Pascal-like style. A module defines a collection of procedures provided in an export specification. Internally, the module describes the algorithmic realization of these procedures. A more detailed discussions of KIV and most of the proofs our examples rely on are given in [Fensel & Schönegge, 1997b], [Fensel & Schönegge, 1997a].

The content of the paper is organised as follows. In section 2, we discuss the problem-specific refinement of methods. First, we present the specification of the incomplete search strategy hill-climbing, a common search strategy for local optima. Then we add an adapter that transforms the method into a method specialized on minimizing sets. Here, the search strategy remains the same but the ontological commitments of the methods become refined. States and state transitions are described with an enriched vocabulary. A further adapter transforms this method into a method for abductive diagnosis (by adding additional ontological requirements). Besides adding ontological

---

1. We used originally the term *task* definition. However our task definitions define only goals and requirements and not a way to achieve a goal. Therefore, the term *problem* definition is more appropriate, see [Breuker, 1997].

The Tower-of-Adapters Method for Developing and Reusing Problem-Solving Methods

Dieter Fensel

University of Karlsruhe, Institute AIFB, 76128 Karlsruhe, Germany.
fensel@aifb.uni-karlsruhe.de, http://www.aifb.uni-karlsruhe.de/WBS/dfe

Abstract. The paper provides three novel contributions to knowledge engineering. First, we provide a structured approach for the development and adaptation of problem-solving methods. We start from very generic search strategies with weak data structures and add adapters that refine the states and state transitions of the search process and that add assumptions necessary to link the competence of a method with given problem definitions and domain knowledge. Second, we show how the usability-reusability trade-off of task-specific versus task-independent problem-solving methods can easily be overcome by the *virtual* existence of specific methods. Third, we provide the concept of an integrated library combining reusable problem definitions, problem-solving methods, and adapters.

1 Introduction

Problem-solving methods (PSMs) are used by most of the current knowledge-engineering frameworks (e.g. Generic Tasks [Chandrasekaran et al., 1992]; Configurable Role-Limiting Methods [Puppe, 1993]; CommonKADS [Schreiber et al., 1994]; the Method-To-Task approach [Eriksson et al., 1995]; Components of Expertise [Steels, 1990]; GDM [Terpstra et al., 1993]; MIKE [Angele et al., 1996]). Libraries of PSMs are described in [Benjamins, 1995], [Breuker & Van de Velde, 1994], [Chandrasekaran et al., 1992], [Motta & Zdrahal, 1996], and [Puppe, 1993]. Despite the strong agreement on the usefulness of PSMs and the large body of documented PSMs there is still a lack of clear methodological support in developing PSMs and in (re-)using them. Recent work [Akkermans et al., 1993], [Fensel, 1995], [Wielinga et al., 1995], [Benjamins & Pierret-Golbreich, 1996], [Benjamins et al., 1996], [Fensel & Benjamins, 1996], [Fensel et al., 1996], [Fensel & Straatman, 1996], [Motta & Zdrahal, 1996], [Fensel & Schönegge, 1997a], [ten Teije, 1997], and [Breuker, 1997] provide in-depth analysis of the essence and main rationales of some PSMs. Some of these papers also outline general steps that have to be taken in developing PSMs. However, it still remains rather unclear how to develop PSMs, how to adapt PSMs to given problems and domain-specific circumstances and how to select PSMs from a library, i.e. how to organise such a library.

Our contribution is concerned with these three problems. We show a principled way of developing and adapting PSMs and provide a new way of organising a library of PSMs to support their reuse. This is mainly achieved by using *adapters* as a means of expressing the refinement of PSMs. Adapters were originally introduced in [Fensel & Groenboom, 1997] to allow the independent specifications of problem definitions[1],

To appear in
Proceedings of European Knowledge Acquisition Workshop (EKAW-97), Lecture Notes in Artificial Intelligence (LNAI), Springer-Verlag, 1997.