Generalized Problem-Solving Methods

Rainer Perkuhn

Institute AIFB
University of Karlsruhe (TH)
D-76128 Karlsruhe, Germany
e-mail: perkuhn@aifb.uni-karlsruhe.de

Abstract

Problem-Solving Methods (PSMs) describe the dynamic reasoning behaviour of a knowledge-based system independent from a certain domain and application. Evidently, the reuse of PSMs across different domains or applications is a challenging issue.

The investigation presented in this paper was directed by the hypotheses that a library of PSMs should be essentially more than just a juxtaposition of its items, and that the fundamental principle for reuse is abstraction. To meet these goals the notion of a *Generalized Problem-Solving Method* (GPSM) is introduced. A GPSM is a model that collects information about similar PSMs, a family of PSMs. At the same time it describes the members of the family and their relations. The approach suggests these abstractions from several PSMs as a more appropriate candidate for reuse retrieval. A GPSM contains some degrees of freedom, interpretable as parameters, but it allows also to model separately how these are resolved. So, after the selection of a GPSM this approach supports a systematic way how to derive a member of the family. To this extent, the reuse of PSMs might already be seen as an engineering discipline.

The main aspects of the approach are incrementally developed in this paper and always related to a case study for the family of assignment (rsp. parametric design). This is finally presented completely.

Keywords: Problem Solving Methods, Reuse, Similarities, Categories of PSMs, Software Architectures, Meta modeling

1 Introduction

The notion of a problem-solving method (PSM) was inspired by a lot of different approaches (Generic Tasks [Chandrasekaran, Johnson, and Smith, 1992], CommonKADS [Schreiber et al., 1994], Method-to-Task Approach [Eriksson et al., 1995], Components of Expertise [Steels, 1990], GDM [Terpstra et al., 1993], MIKE [Angele, Fensel, and Studer, 1996]). PSMs describe the reasoning behaviour of an intelligent agent - be it a human being or a knowledge-based system. Though, suitable models are especially conceptual ones and “platform-independent” by providing modeling primitives on the knowledge level ([Newell, 1982]). Up to now the competing modeling frameworks converged and reached consensus on the fundamental issues a common (“unified”) theory has to cover. [Angele et al., 1996], [Perkuhn, 1997] summarize the synthesis of this development, the new proposal for UPML ([Fensel et al., 1999]) tries to capture the result in a unified modeling language.

Reuse of PSMs promises time, cost, and quality improvement in the development process of a knowledge-based system, incl. maintenance, and a more reasonable assessment of the quality of the resulting product. Mainly, investigations on the reuse of

PSMs focus on the development of libraries ([Motta, 1997], [Breuker and van de Velde, 1994]) but from a reuse process point of view these are useful only to a limited extent. Either they offer only a collection of items with no real support of how to select an appropriate one. Or they attempt to cover a more generalized structure, e.g. task-method-decomposition trees ([Benjamins, 1993]), but are very poor in showing up the relations between the possible specializations. The latter approach seems more promising but evaluations have shown their deficiencies ([Orsvärn, 1996]). The main critics is that the designer of the library did not consider (and is not able to represent) in his models of how to adapt the generalized structure to a special application. Although slight modifications could improve the suitability of the models in that case it remains unclear how many improvements are necessary for more cases and how useful the model still will be after that many modifications.

The tower-of-adapter approach ([Fensel, 1997]) is derived from the necessity to adapt general models, like e.g. basic search schemes, to more specialized circumstances, e.g. special PSMs like propose&revise. But unfortunately it abandoned the idea of the conceptual models. In principle, the approach is a constructive one but up to now it neither offers models that contain the information of the overall structure of the resulting system in a communicable form - as the conceptual models do - nor offers construction plans of how to combine the basic templates with some adapters to come to a certain overall conceptual structure. Of course, adapters might improve the reusability of a system like any other design pattern ([Gamma et al., 1994]) can do. But especially conceptual models of PSMs contain information that is closer to an architectural description ([Shaw and Garlan, 1996]) of the general structure of the target system.

The paper will show that the known approaches based on general structures suffer from the fact that they cannot separately capture aspects considering some kind of flexibility dimension of the models. This is needed on the one hand to express the degrees of freedom in this general models and on the other hand to show the way how to resolve this variability and to adapt the models. This dimension allows to formulate some kind of "strategic" decisions in general but also especially to handle the challenge of embedding optional components as introduced in [Perkuhn, 1997]. This paper suggests Generalized PSMs (GPSMs) as a means to pursue the discussed goals. The notion is based upon the conceptual and communicable models, it describes the overall problem-solving structure for a family of problem types and captures the flexibility dimension. It reconsiders and extends the notion of a component in a sense that it is an isolatable part that can be plugged in into a general architecture and increases, possibly as an add-on, the performance of an already "runnable" system.

The paper is organized as follows. After this introduction section 2 discusses the flexibility dimension and offers primitives to model respective aspects. In section 3 one way is presented how this new dimension can be exploited. Section 4 illustrates the dimension with the problem type of assignment rsp. parametric design. Finally, section 5 concludes and discusses related and further work.

2 Degrees of Freedom in Problem-Solving Structures

The common core of all PSM modeling frameworks consists basically of primitives

- for elementary steps in the problem solving process (subtasks rsp. elementary inference actions) and
- for used and produced knowledge (knowledge roles),

and connections between them as there is

- the task decomposition of a complex task into finer grained steps,

- the inference structure (or data flow), i.e. the overall structure of exchanging data between all the steps that is actually derived from the relation between one step and the knowledge roles it uses as input and produces as output,
- the control flow that describes which steps has to follow in which order, and
- the method ontology that describes the concepts the PSM uses and their relations.

The method ontology is out of the scope of this paper and will not be discussed in detail although it should be immediately clear how to apply the issues discussed to that notion. The inference structure and the control flow are always defined with respect to one task and its decomposition. The paper argues always on this level of granularity but without being committed to this restriction.

[Perkuhn, 1997] investigated the similarities between different PSMs and suggested a framework to represent all members of one family of PSMs in one model. This paper generalizes the notion of a family to that of a *Generalized PSM* (GPSM). In the first approach the family structure was illustrated exemplarily by colouring parts of inference structures - with the intended meaning that the members of the family differ depending on whether all parts of one colour are realized or not. But two slightly different PSMs could still be very similar to each other because the rest of the model might be identical. Actually, a degree of closeness in the family relationship can be estimated depending on how many optional parts appear the same way. Indeed, colouring parts of a model is a way to express meta information about several models, and are for this reason considered as an additional modeling primitive for GPSMs. By annotating nodes and edges of a model with a colour feature rsp. visualizing this feature literally a new dimension can be modelled, in this case the aspect that the parts in one and the same colour belong together and form a region. To know about coloured regions is not important for the GPSM itself but for the reuse process: They are the cornerstones for questionnaires about available knowledge and thus responsible whether the parts could be realized in the final system or not. The investigation on some problems with optional components yields that colouring is only one feature besides many. Similarly, other features are necessary for other kind of meta information. And in enriched models all these features can play important roles in the reuse process. This will be shown by elaborating the notion of optional components. But this serves only as motivation. Afterwards the deep impact of the flexibility dimension in general will be discussed.

2.1 Problems with optional components

First of all, allowing components to be considered optional seems to be a simple notion from the modeller‘s perspective. But to ensure well-defined model transformations in the reuse process the approach has to cope with the fundamental problem:

What happens if the component is not realized? Most examples look reasonable with the component but omitting the component causes fear that this transformation results in a gap and an inconsistency in the model. But for a GPSM both situations should be handled in one framework. Thus, a way is needed to describe the embedding of the optional component

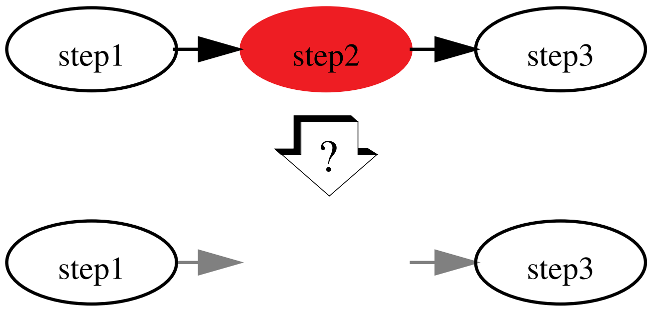

Figure 1. What happens if optional component is omitted?

so that the remaining model does not care about whether it is realized or not.

Taking optional as a synonym to omittable an unwanted gap might cause incompleteness and inconsistency. But since colouring is only an additional modelling primitive, declaring a part as optional is a creative and should be a reasonable act. In the long run the modeller him-/herself is responsible that the reduced model still makes sense. If the gap in the model is intended on purpose, the modeller can declare the connection as optional, too. This is done by colouring the respective edges with the same colour as the component.

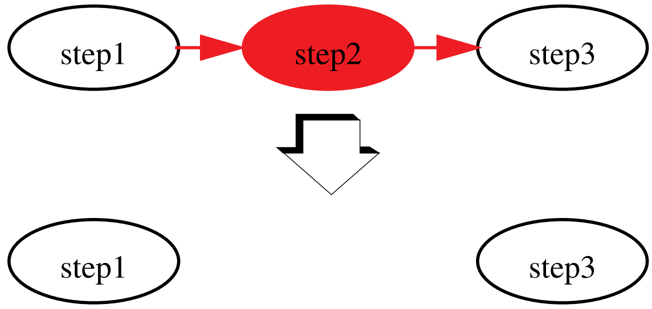

Figure 2. Optional connections

If the combination of the two relationships is feasible then the modeller can consider whether concatenating the two involved connections to their transitive closure makes sense. For this case the edges are not coloured. Now, if the component is wiped out the gap and the adjacent relations have to be replaced by one new relation that should express the transitive closure of the adjacent relations. Examples where the second rule can be applied are taxonomies but may be also in the control flow. In a taxonomy the concepts are related via an is-a-relationship. Of course, this relation is transitive. A concept in a chain of is-a-relations can be omitted but the gap should be bridged with a new is-a-relation that connects the former preceeding and succeeding concepts and replaces the dangling edges. Even the ordering relation in a control flow might be handled similarly. E.g. code optimizing in a compiler is a step that has to be considered in the control flow but in most cases during development of a system it is recommended to leave out this step, thus the result may be represented as in figure 3 (with code optimizing taken as step 2).

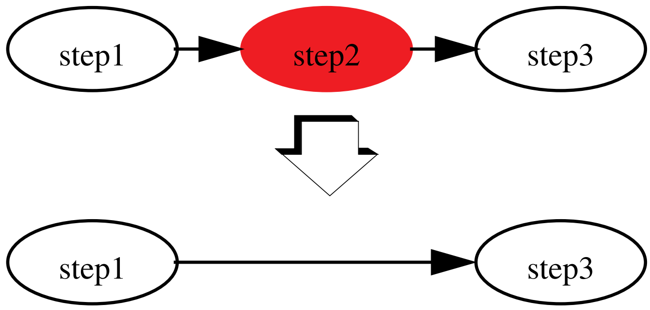

Figure 3. Transitive closure of adjacent relations

If neither of these two proposals work then it is questionable whether the component is really optional at all. To remember: Annotating the model with the colour feature is a creative modeling act and as an epistemological primitive on a higher cross-model level this feature expresses that the modeller believes that the component and possibly the connections are not necessarily part(s) of the model - and by this, part of any context of the model. But obviously if none of both solutions are appropriate the belief was faulty.

These two graph transformation rules handle nearly every GPSM modeling problem - except one: the description of an alternative control flow incorporating an optional component.

2.2 Alternative control flow incorporating optional components

The control flow specification has to prescribe which steps are performed in which order. If the reasoning process reaches a state where it can follow different succeeding paths, the control flow has to specify how to go on. In the ordinary PSM scenario the decision depends on boolean expressions. Resembling the manner of procedural programming languages if- or case-statement like expressions evaluate these boolean expressions and according to their truth value decide for one path.

In a part of the control flow specification of the GPSM for assignment the model has to express that processing can be continued with a propose step or with a generate step. Both deliver a value for a given variable (or attribute), generate in a random fashion, propose founded on some heuristic knowledge about good values but may be restricted only to a subset of all possible variables. Since propose is an optional component the control flow has to take into consideration that it cannot be guaranteed. Applying the idea of boolean expression and if-statements to this case would yield something like the following:

```
if (propose is realized)
  then value = propose(variable)
  else value = generate(variable)
```

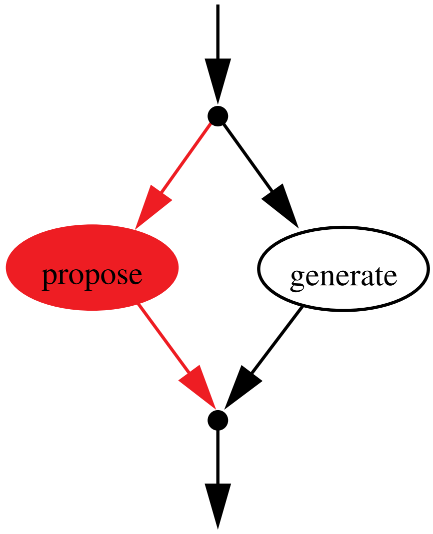

Figure 4. An alternative optional path

But this solution is quite not satisfactory due to two reasons. First, the if-statement is a construction normally used for the control flow specification inside one PSM. But in this case it is used for selecting one of several possible control flow specifications. Similarly and even worse, with the boolean expression it refers to a meta property of the GPSM. Since these are separate affairs it seems more appropriate to make the difference explicit by a different means. Second, although there is a choice between several possible paths - in the case that propose is realized - the if-statement hides this information by deciding for one of the paths. Of course, this decision might be reasonable and has to be made if the model should be expressed in the standard terminology of PSMs. But GPSMs are especially designed to allow this degree of freedom. And to be most useful in the reuse process the resolution of the choice should be made transparent and traceable. Therefore, this approach separates the two aspects

- what alternatives are really available? (→ meta property of GPSM)
- how should be selected if several alternatives are available? (→ resolution of choice rsp. degree of freedom)

At the decision point the problem-solving process is in a state similar to a person that wants to get from one place to another one. When the person reaches a point where the path splits and he/she can decide how to go on - assuming that both paths still lead to the target place -, normally, the selection is not arbitrary but depends on some properties of the alternatives. Even if one path cannot be guaranteed - may be it is closed due to higher circumstances (the short cut through the park is closed during the night) - the ordering of the alternatives can now be formulated independent of the presence or absence of an optional path. First, it has to be checked which path are actually available. Second, if the first check reveals some degree of freedom one path has to be selected due to the properties. These are annotated as features to the different paths. Boolean expressions are not appropriate at decision points because the next step does not depend on an internal state or variable that may vary during runtime like a system model that is complete (or not) or test that is successful (or not). The next step is determined by an external strategy once and normally not changed thereafter.

In the example the properties might describe the length of the path - or the beauty of the landscape while walking along. If both ways are available it seems reasonable to decide for the shorter one. But actually, this selection can only be explained and justified with respect to a certain strategy the person has in mind how the goal should be reached. The length of the path is important if the person wants to get very quickly to the final place. Otherwise, if the person just wants to get out and enjoy the beautiful landscape, he/she would accept a longer way if this is the nicer one. To summarize, alternative paths can be characterized with different properties, strategies express different attitudes to proceed at decision points and how one path should be selected.

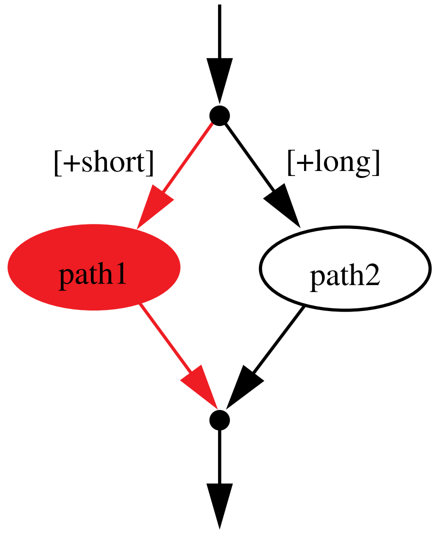

Figure 5. Decision criteria for different paths

Evidently, conflicts might occur if different strategies are combined, e.g. if the person wants to get as quick as possible and while enjoying the most beautiful landscape to his/her goal. In this case the degree of freedom cannot be resolved: one strategy prefers one way and the other strategy prefers the other one. What is needed is a partial ordering of all the alternatives, because at least the most favourite path must be known. If it cannot be determined due to a conflict, feedback should be provided. Then the importance of the properties has to be assessed, e.g. the person has to decide whether the length or the beauty is more important. According to this assessment, one strategy gets a higher weight and overrules the other.

In the example from the assignment GPSM the alternative paths can be annotated as founded on the one hand, and random on the other hand. Once again, the first step to determine the control flow is to collect all available paths. If only generate is present, then trivially this has to be executed next. If propose and generate are realized, then apart from the first step a second one has to select one of the both possible paths. A usual default strategy (cf. table 1) would reflect the superiority and prefer the founded alternative over the random one. Both strategies refine the annotated non-deterministic control flow of the GPSM to the non-annotated deterministic control flow as expected in conventional PSM terminology, namely propose is selected and has to be executed next.

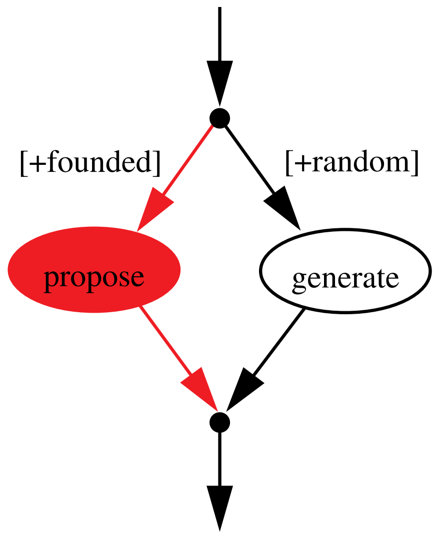

Figure 6. Decision criteria for different alternative next problem-solving steps

By introducing two modelling primitives, namely

- *features* in a somehow non-deterministic control flow on the one hand and
- *strategies* apart from the control flow on a “strategy layer” to resolve the non-determinism on the other hand

the two different concerns what alternatives are available and which one should be selected could be modelled separately. Thus, they fulfill the requirement that was postulated to capture alternatives with optional components. But they can be applied even

strategy “founded”	`prefer a [+founded] path over any other`
strategy “not random”	`prefer any path over a [+random] one`

Table 1 - Two default strategies

more generally. Regardless of optional components they can be used to describe different alternative control flows in general and enable to integrate similar PSMs with slightly different control flows into one common framework, as was intended for the notion of a GPSM. Before this claim can be supported with an example, first the notation has to be introduced completely in all forms it might appear. And of course, the meaning of all forms has to be explained.

2.3 Annotating the non-deterministic control flow for GPSMs

The basic elements of a control flow description of a GPSM are exactly the same as for an ordinary PSM: subtasks rsp. elementary inference actions as the steps in the reasoning process and the ordering relation to express that one steps follows another. In addition a GPSM considers a designated start and end. This is not really a crucial modification but may be makes explicit a possible choice how to begin. In the world of GPSMs this means: It reveals a degree of freedom and also the necessity to model how this is resolved based on features. Since it is the aim to work with the GPSM in the reuse process it is even more important to have as much information as possible (and of course, still reasonable) in the model to demand and to preserve design rationales.

If the ordering is not annotated then the relation is the unconditioned sequential order between the two involved steps. As in ordinary PSMs - having deterministic alternatives in mind - boolean expressions can be annotated to the ordering relation. This means that the path is chosen if the boolean expression is evaluated to the truth value true. Applying this to parallel paths captures if- and case-statement like expressions. If one of the paths is not really parallel but returns to a point that was already visited by the control flow, it describes literally and figuratively a loop. Since in principle all kinds of connections are possible in the control flow, jumps in an unmoderated fashion might result in “spaghetti models”. To avoid this, this approach recommends only restricted use of jumps and forbids jumps from outside into a loop.

For non-deterministic alternatives features are annotated to the ordering relation. This means that the path is considered for the control flow, if the path is the most preferred path due to a strategy that is based on this feature. If only a feature is annotated and the path is preferred then the relation becomes a unconditioned sequential order. But sometimes a combination of features and boolean expressions is necessary because the strategy might still prefer several paths that are discriminated via boolean expressions. For this the notation of a feature-conditioned boolean expression is introduced. This

boolean expression	$\xrightarrow{\text{boolean}}$
feature	$\xrightarrow{\text{[+feature]}}$
feature-conditioned boolean expression	$\xrightarrow{\text{[+feature] \| boolean}}$

Table 2 - Control flow annotations for GPSMs

means that if the path is not preferred rsp. the feature does not support the selection of this path, the whole expression can be ignored. But if and only if the path is preferred due to this feature, then the evaluation of the boolean expression determines the control flow. To sum up, if the strategy prefers this way, the relation becomes a conditioned ordering. Further, conjunctions/disjunctions of these expressions are allowed.

The following example shows that the new modelling primitives not only capture the problems with the optional components but allow in general more flexibility for describing generalized problem-solving structures - and especially the similarities between different PSMs.

2.4 Non-determinism without optional components

All PSMs of the assignment family can be categorized into two groups: those that work holistically, i.e. they first complete the system model (in an inner loop) before they test and revise it, and those that work incrementally, i.e. they already test and revise incomplete models (partial assignments) and extend them in an outer loop. This difference can be connected to a decision in a non-deterministic control flow. Executing propose or generate once yields one value for one variable. So, afterwards the control flow has the option to repeat this first step or to test the system model built up so far. This general property is expressed by the feature [+holistic] for completing first, and [-holistic] for the interleaving tests. The logical complement is used here to express the mutual exclusiveness of the two alternatives. The path back to the beginning (the inner loop) is only considered if the strategy prefers “holistic ways”. The path related to the test is considered if “incremental ways” are preferred but also at least finally in the holistic case (for the outer loop cf. section 4, figure 12). So, deciding for an incremental strategy has trivially the following effect: The loop can be ignored but the connection to test has to be realized as an unconditioned path. The holistic strategy still prefers both paths. But

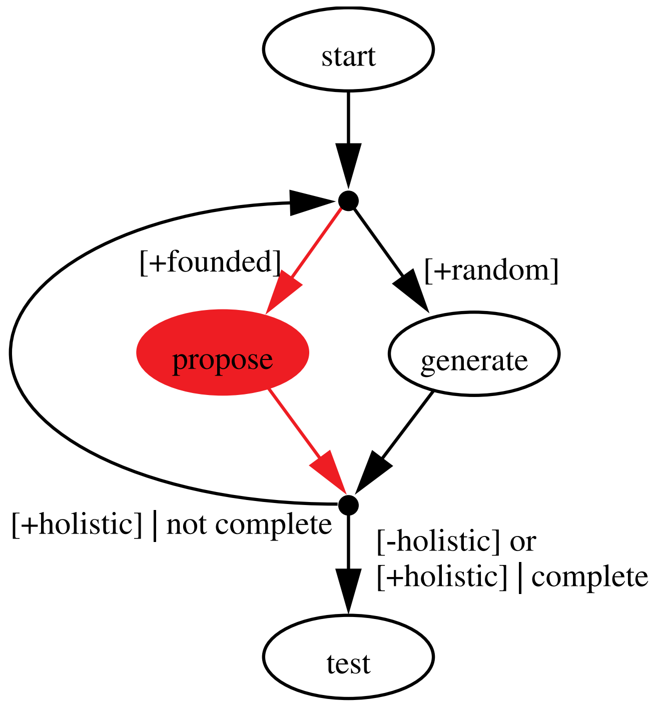

Figure 7. Another decision point in a problem-solving structure

strategy “holistic”	`prefer [+holistic] over [-holistic]`
strategy “incremental”	`prefer [-holistic] over [+holistic]`

Table 3 - Two complementary strategies to be selected by the user

it is clear that the inner loop should not be repeated infinitely but only until the model is complete. This is what may be expressed by an ordinary boolean expression like “complete” or “not complete”. Thus, the feature-conditioned boolean expression has to be read as: Even if the strategy preferred the path due to the feature, solely the evaluation of the boolean expression determines which path to follow.

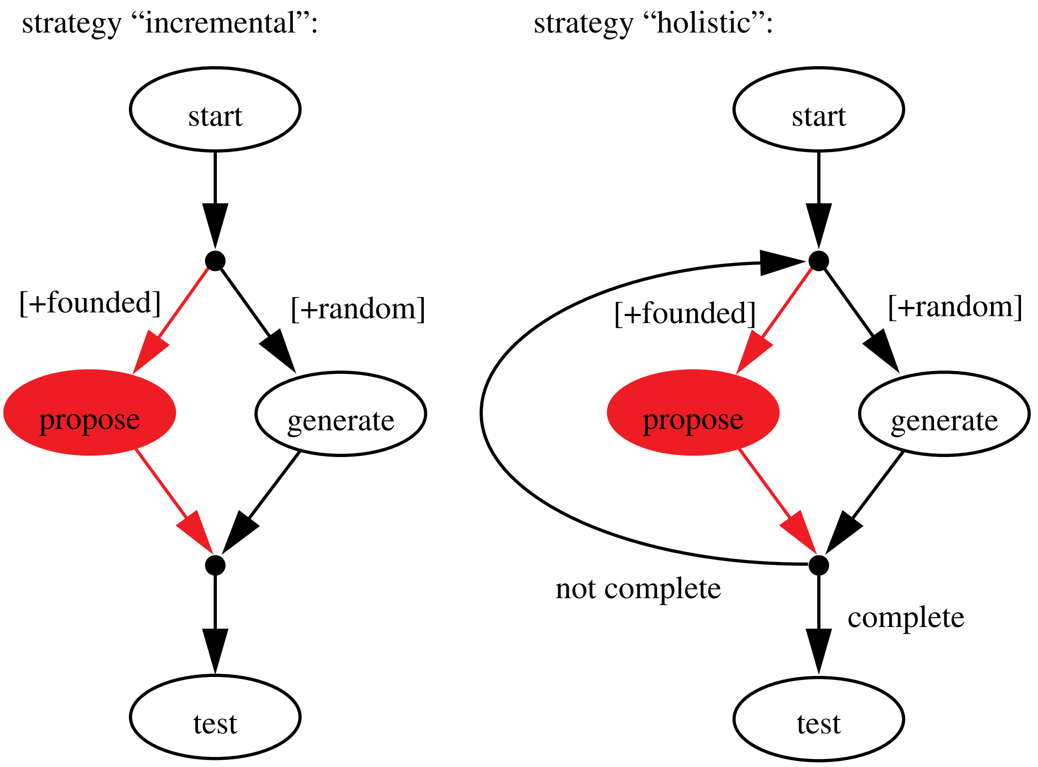

Figure 8. Different control flows determined by different strategies

After having shown informally how to exploit features and strategies the next section explains the rules according to them the GPSM can be reduced to an ordinary PSM. In the outlook of the final section alternative ways to exploit the full benefits of the features and strategies are shortly discussed.

3 How to Exploit the New Dimension

The approach outlined so far offers a systematic way how to get from a general structure, a GPSM, to the well-known specification of the reasoning behaviour of a knowledge-based system, a PSM.

Assuming that a GPSM could be selected first the optional parts are examined. After clarification which component could be realized, strategies have to be determined for possibly remaining non-deterministic alternatives. The colouring and the other features are special kinds of task features ([Aamodt et al., 1993]) that are correlated to and elicited by questions that are used in structured interviews. With this information, i.e. the GPSM, the optional parts, the features, and the strategies, the following algorithm can be applied to derive a deterministic PSM via graph transformation rules.

selection of GPSM
↓
examining optional parts
↓
determining strategies
↓
reducing GPSM to PSM

Figure 9. How to get from a GPSM to a PSM

In a nutshell, the algorithm removes all the fea-

```
for all feature-annotated paths p do
  for all features f annotated to p do
    decide whether p is preferred due to f or not
    if p is preferred due to f then
      replace f with true rsp.
        with the following boolean expression
        /* f -> true and
          f | bool -> bool */
    else
      replace f inclusively a following boolean
        expression with false
        /* f -> false and
          f | bool -> false */
    if path annotation a is a conjunction or
      disjunction then
      simplify a according to rules of
        propositional logic
        /* a and true -> a      a or true -> true
          a and false -> false  a or false -> a */
    if annotation is true then
      remove annotation
    if annotation is false then
      remove annotation and path
```

Algorithm 1 - to reduce GPSM to PSM

tures from the model. Supported features effect the persistence of paths and boolean expressions, not supported features may cause removal of paths and boolean expressions. In a disjunction the positive effect of supported features dominates over the not supported features, but in a conjunction the dominance is vice versa. This algorithm was applied in the last section for the "holistic" example. In the following tables some details are shown that were left out for simplification reasons.

	strategy	
	holistic	incremental
annotation	[+holistic] | not complete	
replacement	not complete	false
effect	edge and boolean expression "not complete" remain in the model	edge and boolean expression "false" removed from the model

Table 4 - Illustration of algorithm with simple annotation

Taking the essence of this algorithm the notion of "optional" has to be redefined with respect to each context. Optional means omittable in a sense that the remaining model still must be reasonable, e.g. in a control flow an alternative path from the start to the

	strategy	
	holistic	incremental
annotation	[-holistic] or [+holistic] | complete	
replacement	false or complete	true or false
simplification	complete	true
effect	edge and boolean expression "complete" remain in the model	boolean expression "true" removed from the model, edge remains in the model

Table 5 - Illustration of algorithm with annotated disjunction

end still has to exist.

With colours, features, strategies, and this algorithm a complete framework is provided to model a family of PSMs and to derive all its members from its prototype representation, the GPSM. The next section is intended to illustrate the complete GPSM model for the assignment family. Although most details were already explained in the previous sections some further issues will be discussed.

4 Case Study: A GPSM for Assignment (rsp. Parametric Design)

In the following a complete model of the GPSM for assignment is presented - only missing is an explicit method ontology. The first two contexts are very similar to the example in [Perkuhn, 1997] and just illustrate the use of colours.

The task decomposition concentrates on one level of refinement. Not shown here is the

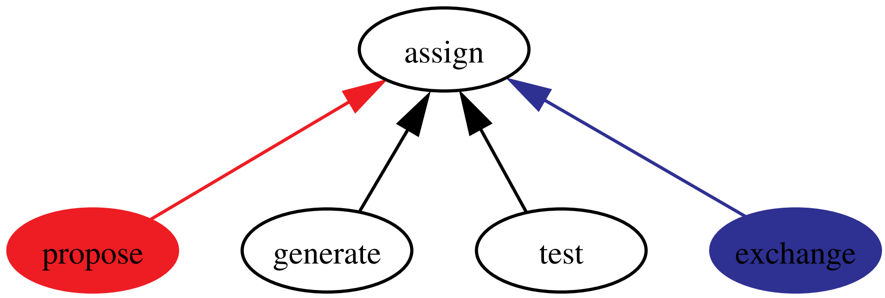

Figure 10. A coloured task decomposition for the assignment family (cf. [Perkuhn, 1997])

further refinement of the exchange step that could actually be performed as generating new assignments for a subset of the variables either randomly, based on special exchange knowledge, or by applying special revise knowledge that indicates how certain constraint violations can be fixed.

The inference structure shows partial assignment as central role that is filled from generate or propose and is used by test. Exchange connects the test result with this central role from the other side. For a reasonable exchange step the test must deliver information about constraints that are violated whereas a simple notion of violation may also be only a binary decision. Propose and exchange require special knowledge ("static roles") and all the parts of a coherent region that depend on it are coloured - red for the

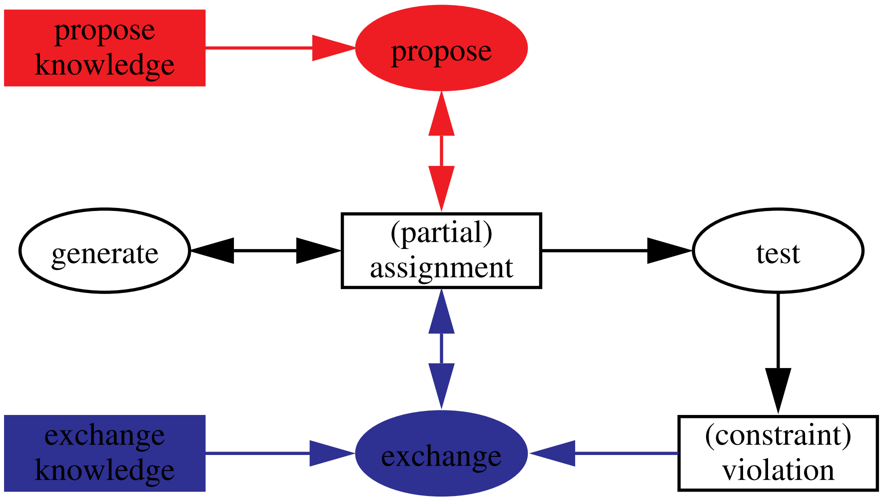

Figure 11. A coloured inference structure for the assignment family (cf. [Perkuhn, 1997])

propose and blue for the exchange region. Since not only the components but also the connections are coloured the whole regions are considered optional. Dependent on the availability of the special knowledge they presuppose the regions are either completely realized or not.

In the description of the non-deterministic control flow the three decision points are most interesting. The first enables the choice between propose and generate based on the features founded rsp. random (and was already explained in subsection 2.2, cf. figure 6). The second characterizes the difference between the way for a holistic strategy and the way for an incremental strategy (and was already explained in subsection 2.4, cf. figure 7). The third and last decision point is also correlated to the same features as the second one. This point reflects the circumstances under which the PSM can successfully terminate and especially it is responsible for the outer loop of the incremental variants of the family of assignment PSMs. Feature-conditioned boolean expressions enable the model to express that the test for completeness has to take place only once: with an incremental strategy in mind at the last decision point and with an holistic strategy in mind before the test. Since in the latter case this has to be asserted before the test is performed (and test has no effect on it) the test for the terminating path can be evidently reduced to a simple test of success. Actually, the combination of features and strategies is an instrument to capture global effects: The strategy determines which boolean expressions and which paths have to be considered at two different decision points. If the same effect is intended with ordinary control flow primitives the result looks very awkward if it is possible at all. The simple control flow has to compile strategic decisions into boolean expressions and thus makes the clear conceptual distinction nearly impossible. Maintenance and transparency of the model suffer from this mixing up of strategic and other aspects.

This GPSM describes a whole range of PSMs that could be understood as parametrized with respect to the optional components and strategies:

$\text{GPSM}_{\text{family}}$ ((optional components), (strategies)) = $\text{PSM}_{\text{optional components/strategies}}$

e.g.

$\text{GPSM}_{\text{assignment}}$ ((propose ✔, exchange ✔, revise ✔),
(strategy “not random”, strategy “holistic”))
= $\text{PSM}_{\text{propose\&revise/holistic}}$ = Complete-Model-Then-Revise

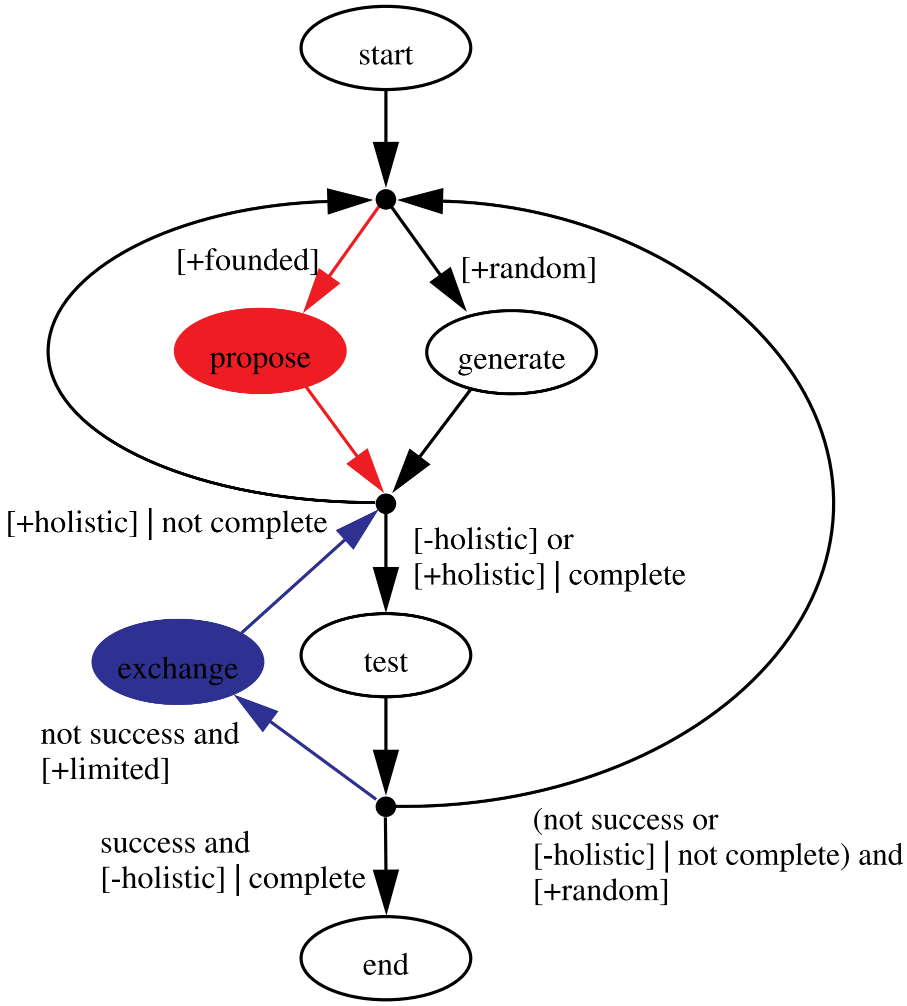

Figure 12. A coloured non-deterministic control flow for the assignment family

This framework offers the advantage in contrast to other approaches ([ten Teije, 1997]) that the parameters are very closely attached to the conceptual models and, by this, can be exploited in a communicative situation. All parameters can be expressed in a way that can be understood by an expert but they are also useful for the knowledge engineer.

5 Conclusion, Further and Related Work

A fundamental principle for reuse-oriented approaches is abstraction. And a critical success factor is how appropriate the abstraction is and which role it can play in the further (adaptation) process. GPSMs are abstractions of families of PSMs and thus well-suited candidates for retrieval. Furthermore, features and strategies provide a systematic way how to adapt the model to a PSM. A GPSM is a framework where questions can be tagged to the features, making decisions according to structured interviews traceable. In a way the features correspond to design rationales because they can be explained from their context.

Section 3 showed how to resolve the degree of freedom of a GPSM - mainly assumed at development time of the system. But it could be worth while considering to work with the GPSM at runtime, i.e. to keep the alternative paths and to resolve the non-determinism on the fly at runtime. This offers the possibility to switch the strategy at runtime and to take into account that availability of components may vary. In a distributed web-based, e.g. CORBA-oriented scenario (cf. e.g. [Benjamins et al., 1998]) a server providing an optional component (or the connection) may break down - but with the chance to switch to another previously not preferred component the system‘s robustness is improved. Of course strategy switching may cause inconsistencies, so the effect that a switch causes should not be allowed at any time but only at well-defined synchronization points. The switch between an holistic and an incremental strategy e.g. may only affect the control flow at the first decision point due to global interdependency between the inner and the outer loop.

A somehow similar argument supports the runtime management idea of GPSMs: Some components may work not for any input or situation, e.g. propose yields a value normally only for a subset of all possible variables. In the case that propose cannot deliver a reasonable value, generate could help and should be called. Actually, the communication in this case is an extraordinary one. Although conventional control flow has to tackle this problem by some kind of “undefined value” propose may return (and that triggers the call of generate) it would be more appropriate to handle this situation as an exception. Exceptions are a special distinct way to communicate ahead from return values. The above mentioned example could be modelled very adequately as a loop that tries alternatives according to the strategic (now required) total ordering until no exception is raised.

```
repeat
  try
    nextElement(alternatives, strategies)
until no exception is raised
```

Besides the investigation of further families and GPSMs (currently classification, diagnosis and assessment) the lazy evaluation of strategies will be an interesting and promising topic for future work.

The ambition to make reuse of knowledge (level) models more flexible is not new ([Geldof, 1994]). In the first version KADS ([Schreiber, Wielinga, and Breuker, 1993]) suggested a strategy layer for this purpose. But both only consider the possibility to chose between different methods with the same competence for one task. Similarly GDM ([Terpstra et al., 1993]) allows the application of alternative rewrite rules that may cause comparable effects to the graph transformation rules in the approach presented here. But none of these three frameworks allows to model explicitly decision criteria or resolution strategies. There is no flexibility related directly to the conceptual models and there is no way to capture global effects on different strategies.

Other approaches focus more on strategic aspects. TASK+ ([Pierret-Golbreich and Talon, 1997]) tries to describe these with abstract data types; DESIRE ([Brazier et al. 1997]) allows to specify multi agent systems. But both are only loosely coupled to the conceptual models in the sense presented here and they are not able to separate the concerns on different layers. DESIRE e.g. uses ordinary if-statements for activating agents, i.e. for selecting the control flow. Thus, the difference between ordinary control flow and strategic aspects is not obvious in these models.

GPSMs are somehow complementary to Towers-of-Adapters ([Fensel, 1997]). Both assume a componential architecture but although [Fensel, 1997] provides more details about basic search schemes and refinements it lacks the “construction plan” that expresses how to build the same tower several times. And this is exactly the information the GPSMs contain.

Very close in spirit is the idea of configuring PSMs via parametric design ([ten Teije et al., 1996], [ten Teije, 1997]) that is investigated for diagnosis. But the suggested parameters are only hardly to understand as underpinned by the conceptual model. The major weakness of the models is the insufficient expressiveness for specifying control flow especially for capturing alternatives.

The work presented in this paper is the only one that combines the strict relation to the conceptual models with an explicit layer to capture and specify alternatives and their resolution.

References

[Aamodt et al., 1993] A. Aamodt, B. Benus, C. Duursma, C. Tomlinson, R. Schrooten, and W. van der Velde: *Task Features and their Use in CommonKADS.* Deliverable 1.5. Version 1.0, Consortium, University of Amsterdam, 1993.

[Angele et al., 1996] J. Angele, S. Decker, R. Perkuhn, and R. Studer: Modeling Problem Solving Methods in New KARL. In: *Proceedings of the 10th Knowledge Acquisition for Knowledge-Based Systems Workshop (KAW'96)*, Banff, Canada, November 1996, 1-1 - 1-18.

[Angele, Fensel, and Studer, 1996] J. Angele, D. Fensel, and R. Studer: Domain and Task Modeling in MIKE. In: A. Sutcliffe, D. Benyon, F. van Assche (eds.): *Domain Knowledge for Interactive System Design*, Chapman & Hall, 1996, 149-163.

[Benjamins, 1993] R. Benjamins: *Problem Solving Methods for Diagnosis.* Ph.D. Thesis, University of Amsterdam, Amsterdam, 1993.

[Benjamins et al., 1998] R. Benjamins, E. Plaza, E. Motta, D. Fensel, R. Studer, B. Wielinga, G. Schreiber, and Z. Zdrahal: - IBROW3 - An Intelligent Brokering Service for Knowledge-Component Reuse on the World-Wide Web. In: *Proceedings of the 11th Banff Knowledge Acquisition for Knowledge Based Systems Workshop (KAW'98)*, Banff, Canada, April 18-23, 1998, SHARE-2- 1-12.

[Brazier et al. 1997] F.M.T. Brazier, B.M. Dunin-Keplicz, N.R. Jennings, and J. Treur: DESIRE: Modelling Multi-Agent Systems in a Compositional Framework. *International Journal of Cooperative Information Systems: Multiagent Systems*. 6 (1), 1997, 67-94.

[Breuker and van de Velde, 1994] J.A. Breuker and W. van de Velde (eds.): *The CommonKADS Library for Expertise Modeling*. IOS Press, Amsterdam, 1994.

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

[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, 1995, 293-326.

[Fensel, 1997] D. Fensel: The Tower-of-Adapter Method for Developing and Reusing Problem-Solving Methods. In: [Plaza and Benjamins, 1997], 97- 112.

[Fensel et al., 1999] D. Fensel, R. Benjamins, S. Decker, M. Gaspari, R. Groenboom, W. Grosso, M. Musen, E. Motta, E. Plaza, G. Schreiber, R. Studer, and B. Wielinga: The Component Model of UPML in a Nutshell. To appear in: *Proceedings of the 1st Working IFIP Conference on Software Architecture (WICSAI)*, San Antonio Texas, USA, February 22-24, 1999.

[Gamma et al., 1994] E. Gamma, R. Helm, R. Johnson, and J. Vlissides: *Design Patterns. Elements of Reusable Object-Oriented Software.* Addison-Wesley, Reading/Mass. 1994.

[Geldof, 1994] S. Geldof: Towards More Flexibility in Reuse. In: *Proceedings of the 14th International Conference in Artificial Intelligence, KBS, Expert Systems, Natural Language of Avignon*. Paris, 1994, 65-75.

[Gennari et al., 1994] J.H. Gennari, S. Tu, Th.E. Rothenfluh, and M.A. Musen: Mapping Domains to Methods in Support of Reuse. *International Journal of Human-Computer Studies (IJHCS),* 41, 1994, 399-424.

[Motta, 1997] E. Motta: *Reusable Components for Knowledge Modeling*. Ph.D. Thesis, Knowledge Media Institute, Open University, Milton Keynes, UK, 1997.

[Motta and Zdrahal, 1996] E. Motta and Z. Zdrahal: Parametric Design Problem Solving. In: *Proceedings of the 10th Banff Knowledge Acquisition for Knowledge Based Systems Workshop (KAW'96)*, Banff, Canada, November 1996, 9-1 - 9-20.

[Newell, 1982] A. Newell: The Knowledge Level. *Artificial Intelligence,* 18, 1982, 87-127.

[Orsvärn, 1996] K. Orsvärn: Principles for Libraries of Task Decomposition Methdos - Conclusions from a Case Study. In: N. Shadbolt, K. O'Hara, G. Schreiber (eds.): *Advances in Knowledge Acquisition. Proceedings of the 10th European Knowledge Acquisition Workshop (EKAW'96),* Nottingham, England, May 1996, Lecture Notes in Artificial Intelligence (LNAI), vol. 1076, Springer, Berlin, 1996, 48-65.

[Perkuhn, 1997] R. Perkuhn: Reuse of Problem-Solving Methods and Family Resemblances. In: [Plaza and Benjamins, 1997], 174-189.

[Pierret-Golbreich and Talon, 1997] C. Pierret-Golbreich, X. Talon: Specification of Flexible Knowledge-Based Systems. In: [Plaza and Benjamins, 1997], 190-204.

[Plaza and Benjamins, 1997] E. Plaza, R. Benjamins (eds.): *Knowledge Acquisition, Modeling and Management. Proceedings of the 10th European Workshop (EKAW'97),* Sant Feliu de Guixols, Catalonia, Spain, October 1997, Lecture Notes in Artificial Intelligence (LNAI), vol. 1319, Springer, Berlin, 1997.

[Puerta et al., 1992] A. R. Puerta, J. W. Egar, S. W. Tu, and M. A. Musen: A Multiple-Method Knowledge Acquisition Shell for the Automatic Generation of Knowledge Acquisition Tools. *Knowledge Acquisition*, 4, 1992, 171-196.

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

[Schreiber et al., 1994] A.Th. Schreiber, B.J. Wielinga, R. de Hoog, H. Akkermans, and W. van de Velde: CommonKADS: A Comprehensive Methodology for KBS Development. *IEEE Expert*, December 1994, 28-37.

[Shaw and Garlan, 1996] M. Shaw, D. Garlan: *Software Architectures. Perspectives on an Emerging Discipline.* Prentice Hall, Upper Saddle River, NJ, 1996.

[Steels, 1990] L. Steels: Components of Expertise. *AI Magazine*, 11(2), 1990, 29-49.

[ten Teije, 1997] A. ten Teije: *Automated Configuration of Problem Solving Methods in Diagnosis.* Ph.D. Thesis, University of Amsterdam, Amsterdam, 1997.

[ten Teije et al., 1996] A. ten Teije, F. van Harmelen, G. Schreiber, and B. Wielinga: Construction of Problem Solving Methods as Parametric Design. In: *Proceedings of the 10th Banff Knowledge Acquisition for Knowledge Based Systems Workshop (KAW'96)*, Banff, Canada, November 1996, 12-1 - 12-20.

[Terpstra et al., 1993] P. Terpstra, G. van Heijst, B. Wielinga, and N. Shadbolt: Knowledge Acquisition Support Through Generalized Directive Models. In: J.-M. David, J.-P. Krivine, and R. Simmons (eds.): *Second Generation Expert Systems*, Springer, Berlin, 1993, 428-455.