Published in: Fensel, Dieter/Studer, Rudi (Eds.): Knowledge Acquisition, Modeling and Management. 11th European Workshop, EKAW'99 Dagstuhl Castle, Germany, May 26–29, 1999 Proceedings - Berlin/Heidelberg: Springer, 1999. Pp. 361-366.
(Lecture Notes in Artificial Intelligence 1621)

Describing Similar Control Flows for Families of Problem-Solving Methods

Rainer Perkuhn

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

Abstract

A library of software components should be essentially more than just a juxtaposition of its items. For problem-solving methods the notion of a family is suggested as means to cluster the items and to provide partially a structure of the library. This paper especially investigates how the similar control flows of the members of such a family can be described in one framework.

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. 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. 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. 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. Nevertheless the approach is an interesting alternative rsp. completion to indexing the library with simple keywords or logical formulae-based pre-/post-condition annotations. Actually, it is not far away from object-orientation - another view on reuse that claims that the inheritance hierarchy provides a reasonable structure of the reuse components and, thus, solves a good deal of the indexing problem. But a PSM cannot be captured completely by the notion of an object in this sense since it e.g. contains an explicit specification of the control flow. [Perkuhn, 1997] suggested the concept of a family of PSMs that describes the overall architecture for a class of similar methods. In the same fashion as in object-orientation it is intended to structure a part of the library. Thus, only a family has to be retrieved from the library by an additional mechanism. Afterwards the selection of a PSM corresponds to systematic browsing through the family. [Perkuhn, 1997] focussed on inference structures while this paper especially investigates how to describe similar control flows for a range of closely related PSMs.

2 Families of PSMs

Most aspects of a PSM that can be represented in a specific part of the model (layer rsp. view) can be distinguished between elements like concepts or roles and steps in the problem solving process on the one hand and relations between them on the other hand. [Perkuhn, 1997] introduced colouring as a means to express that they are not necessarily parts of the model. All elements in the same colour form a region. A coloured region possibly has to be omitted if certain conditions are not fulfilled. In the example of [Perkuhn, 1997] (a family for assignment tasks that covers generate-and-test, propose-and-exchange, and propose-and-revise) different regions depend on the availability of the knowledge for static roles namely propose, exchange, and revise. The former two are illustrated in figure 1. The revise region is part of the refinement of the exchange step that is not shown here. If one of these static roles cannot be filled, i.e. the knowledge is not available or cannot be acquired either, the corresponding region has to be removed from the model. For these cases it seems to be appropriate to colour and remove all related connections, too. But in other cases the resulting model has to be kept consistent rsp. coherent. E.g. to restore the coherence of a taxonomy or a sequential control flow the gap in the model has to be bridged with the transitive closure of the adjacent relations (cf. figure 2) - if possible and reasonable. Colouring is a creative modeling act and expresses as an epistemological primitive on a cross-model level that the

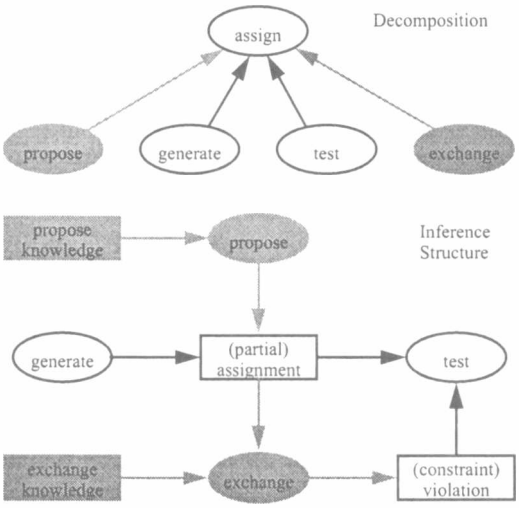

Figure 1. Coloured task decomposition and inference structure for the assignment family (cf. [Perkuhn, 1997])

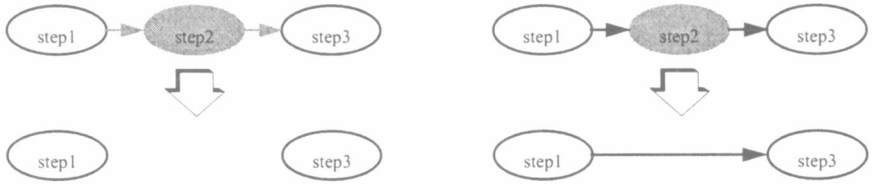

Figure 2. Optional connections vs. transitive closure of adjacent connections

model should still make sense in all variations with or without the regions in every context.

Capturing similar control flows imposes additional requirements on the modeling framework. 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 - normally expressing an internal state during computation. 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. Overlaying different control flows introduces a new aspect that has to be distinguished from these internal states.

In the example family some variants begin with a propose step, others with a generate step. Some other variants use the generate step as a fallback action if for a certain variable ("parameter") no propose knowledge is applicable. Since in most cases - if propose is realized - generate is not taken into consideration at all it is not appropriate to put these two steps into a sequential order. Rather they should be treated as alternatives. An additional mechanism is necessary to handle the fallback variant, but, actually, this distinction exactly reflects the difference of ordinary inter-process communication via return values and extra-ordinary exception handling. This new form of alternative does not depend on an internal state of the computation. It is a kind of non-deterministic decision point to be resolved with respect to the possible variants.

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. In this framework these are annotated as features to the different paths. The next step is then determined by an external strategy that weighs up the different properties. In the example of the assignment family the alternative paths can be annotated as "founded" on the one hand, and "random" on the other hand. A usual default strategy (cf. table 1) would reflect the superiority and prefer the founded alternative over the random one.

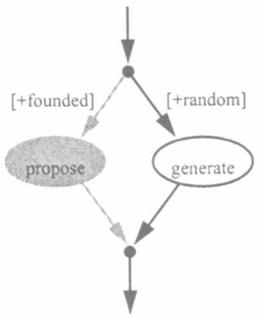

Figure 3. Decision criteria annotated as features

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

Table 1 - Two default strategies

The following example illustrates how the combination of properties and internal states is annotated as feature-conditioned boolean expression.

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. 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. 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

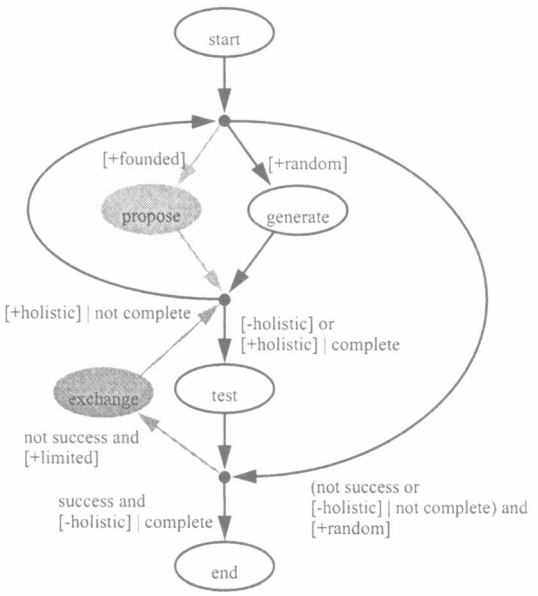

Figure 4. A coloured non-deterministic control flow for the assignment family

strategy "holistic"	`prefer [+holistic] over [-holistic]`
strategy "incremental"	`prefer [-holistic] over [+holistic]`

Table 2 - 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. It is worth while to mention that the completeness of the model has to be checked only once - either before or after the test. Thus, this framework is able to cope with different strategies that cause different global effects on the control flow.

By introducing two modeling 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, a family is a parametrized representation of several PMSs with respect to the optional components and the strategies that can be chosen. 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. After selecting a family the parameters can be checked systematically, e.g. with a questionnaire, to provide actual parameters for an instantiation of the generic family.

3 Conclusion and Related 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. GDM claims that meta knowledge helps them to cope with some of these problems. But similar to task features ([Aamodt et al., 1993]) there is no direct relation to the respective part of the model so that it could be explained and justified from its context.

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.

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: [KAW, 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.

[Brazier et al. 1997] F.M.T. Brazier, B.M. Dunin-Keplicz, N.R. Jennings, and J. Treur: DESIRE: Modeling 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.

[KAW, 1996] *Proceedings of the 10th Banff Knowledge Acquisition for Knowledge Based Systems Workshop (KAW'96)*, Banff, Canada, November 1996.

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

[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: [KAW, 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.