An Ontology-based Broker: Making Problem-Solving Method Reuse Work

Dieter Fensel
University of Karlsruhe, Institute AIFB, 76128 Karlsruhe, Germany.
E-mail: dieter.fensel@aifb.uni-karlsruhe.de

Abstract. We present the architecture of an intelligent broker for enabling the use of problem-solving methods via the World Wide Web (WWW). The core component of such a broker is realised by an ontologist and an adapter. Ontologies mediate between domain-specific requirements and knowledge, task-specific problem descriptions and method-specific terms describing the competence and requirements of the reasoning components. The ontological reasoning for relating the different ontologies is supported by the ontologies. Adapters are necessary to provide domain knowledge and case data to problem-solving methods and to rephrase the output of problem-solving methods into domain-specific terms. Therefore, the ontologist mediates the *selection* and *adaptation* process of PSMs whereas the adapter mediates the *execution* of them.

1 Introduction

The slogan level is the level above the knowledge level
where you can introduce CIFs (i.e., community interchange formats).

The concept *problem-solving method* (PSM) is present in a large part of current knowledge-engineering frameworks (e.g. GENERIC TASKS [CJS92]; ROLE-LIMITING METHODS [Mar88], [Pup93]; CommonKADS [SWA+94]; the METHOD-TO-TASK approach [EST+95]; COMPONENTS OF EXPERTISE [Ste90]; GDM [THW+93]; MIKE [AFS96]). Libraries of PSMs are described in [Ben95], [BV94], [CJS92], [MoZ96], and [Pup93]. In general a PSM describes in a domain-independent way which reasoning steps and which types of knowledge are needed to perform a task. Problem solving methods are used in a number of ways in knowledge engineering: as a guideline to acquire problem-solving knowledge from an expert, as a description of the main rationale of the reasoning process of the expert and the knowledge-based system, as a skeletal description of the design model of the knowledge-based system, and to enable flexible reasoning by selecting methods during problem solving.

PSM should enable reuse of reasoning strategies. However, only a few example of actual reuse of implemented PSM can be found. Mainly two reasons can be identified for this shortcoming. A pragmatical problem is that implemented PSMs make strong assumptions on software and hardware environments that limit reuse in other environments. Another problem at a more theoretical level is the fact that it is usually very difficult to relate the competence and knowledge requirements of a PSM with the actual problem (i.e., task) and the available domain knowledge. This is also discussed as the *indexing* problem. That is, how to annotate PSMs that they can easily be selected and adapted to given task-specific and domain-specific circumstances.

Our work provides a contribution to overcome these two bottlenecks that hampers the successful reuse of PSMs. First, we discuss the idea of using ontologies to represent task-, domain-, and PSM-specific terminologies and formulating the indexing and mapping problem as ontological reasoning. Second,

we present the architecture of an intelligent broker that realises these ontological reasoning processes and that provides the possibilities to integrate PSMs in the WWW.[1] Therefore, hardware and software limitations can be bypassed if an application has appropriate access to the WWW. Adapters (cf. [FeG97]) realize the translation process between different terminologies by realising the knowledge flows and dataflows between applications and PSMs.

We discuss the *selection* and *adaptation* of PSMs from the WWW. That is, we do not discuss the case of distributed problem solving or automatic re-configuration of problem solvers during the problem solving process. Instead, after selecting and adapting one PSM it is assumed that the method solves the given application problems. Therefore, assumptions over the task and requirements on domain knowledge have to be evaluated carefully. Finally, we do not regard down loading of PSMs to the clients side. For pragmatical reasons, we assume instead the method running on a central PSMs-server.

In section 2 we present the general architecture of an intelligent broker providing PSMs to WWW-clients. Its core elements are described respectively in section 3 and section 4. It consists of an ontological reasoner and of adapters. The *ontologist* finds PSMs based on task-specific and PSM-specific ontologies. The *adapter* maps these different terminologies to enable knowledge and data exchange between PSMs and domain knowledge. Related work is discussed in section 5. Finally, we provide conclusions in section 6.

2 The Intelligent Broker

Providing different libraries of problem-solving methods via the net requires an *intelligent broker* (IB) that mediates between customers and providers of PSMs. The general layered architecture of such a broker is depicted in Figure 1. The IB has to communicate with client via *clients systems*. A client is somebody who has a complex problem but can provide domain knowledge that describes this problem and that supports problem-solving. Because we assume the WWW as medium of interactions, we want

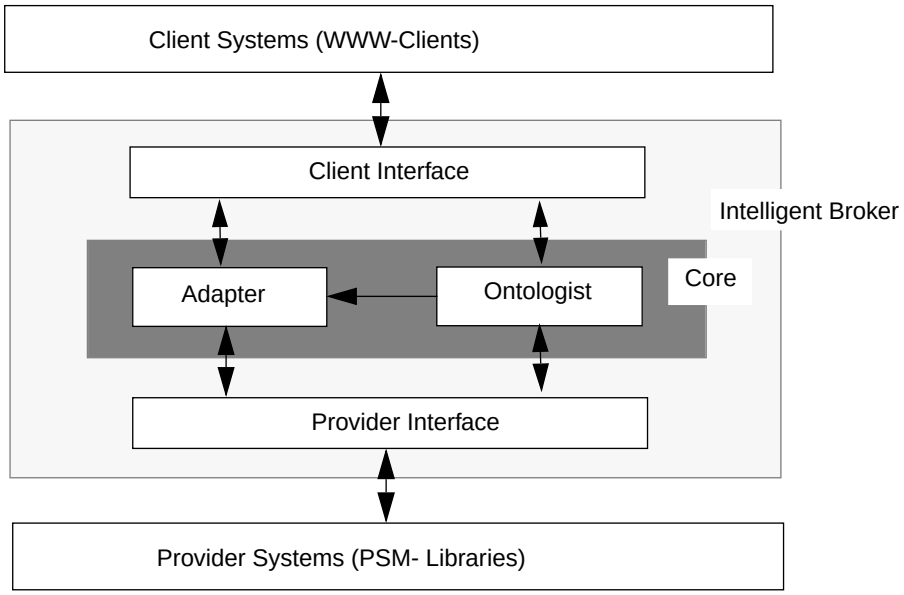

Fig. 1 The layered architecture of the intelligent broker.

1. Whereas the development of such an intelligent broker (intelligent because it is based on ontological reasoning) can be realised by one research group it requires a joint activity of the knowledge acquisition community to present their PSMs in a way that they can communicate with the broker.

to provide support for WWW-*clients* as client systems (e.g., Netscape). The providers are developer teams for PSMs. Their *provider systems* are annotated libraries of PSMs. The PSMs are implementations that solve complex tasks by using domain knowledge for defining and solving the problem. Their annotations are necessary to support their selection process and the communication process with the methods.

The three layers of the functionality of an IB are *client interface*, *provider interface*, and *core* (cf. Figure 1). We will discuss them during the following.

2.1 Client Interface

The *client interface* of the IB shall be used for two types of communications with the client. First, it has to communicate with the client system of the customer for mediating the translation and negotiation process that transforms the requirements into features that can be used to guide the selection process of a PSM. Second, it has to receive the domain knowledge and case data from the client and it has to provide the solutions of the PSM to the client system.

In this scenario, the PSM is offered as a *service* to web-clients. They can also be offered as a *product*. Instead of executing the PSM by the provider it would in principle also be possible to down load the PSM by the client. However, this requires platform independent implementation of PSMs and the client also had to download the adapter element of the IB (see section 4) that manages the terminological mapping between PSM, task, and domain. Therefore, we exclude this more complex possibility at the moment.

2.2 Provider Interface

The IB deals with provider systems via the *provider interface*. A provider system is a library of PSMs. A provider interface asks for two main elements.

First, a provider interface asks for descriptions of the components of PSMs that allow to select and to work with these components. We require three types of descriptions that have to be provided by a library of PSM: information that is relevant for selecting components; information about the syntactical structure of input and output of a components when executing them; and information about its communication style. A provider that wants to be regarded by the IB has to provide each information in a standardised way. We currently examine formal languages like CARIN [LeR96], Frame-logic [KLW95] (i.e, FLORID), LOOM [Mac90], Ontolingua[2] [FFR96], and SHOE [LSR96] as *languages* to express the selection criteria. These approaches allow the representation of terminological knowledge and inferences to derive additional information on these terminologies. In addition, we have to develop standard *ontologies* for describing PSMs and tasks (cf. [Tei97], [DFM??]).

Second, a provider interface asks for the service of the components that realize problem solving for complex tasks requiring domain knowledge and case data as input and providing a solution to the interface.

2.3 The Core of an IB for PSMs

The core of an IB (for PSMs) consists of two main elements: an *ontologist* and an *adapter*. An *ontologist* must support the selection and adaptation process of PSMs for a given application. Basically it has to provide support in building or reusing a domain ontology and in relating this ontology with a task ontology that describe generic classes of application problems. This task ontology has to be linked

2. Ontolingua provides excellent tool support for constructing standardized ontologies. However its support in ontological inferences is very limited.

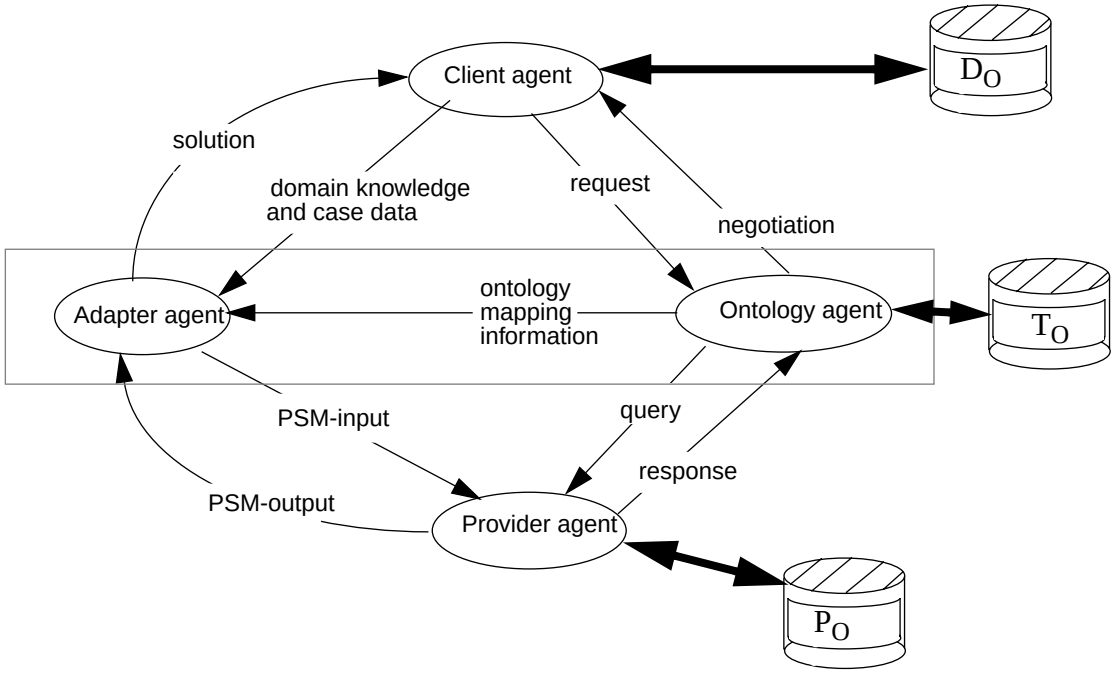

Fig. 2 Internal dataflows, agents, and libraries of the intelligent broker.

with PSM-specific ontologies that allow the selection of a method if a task model is constructed. We will provide examples for these ontologies and their connections in the following sections. Besides modelling and connecting different ontologies we have to provide the actual mapping of these terminologies via an *adapter* [FeG97] for executing a PSM. In principle this mapping could be achieved by always repeating the ontological reasoning that was necessary to establish the link between domain, task and PSM. However, this looks not very promising from a performance point of view. Therefore, we have to provide (semi-)automatic support in constructing an adapter using the outcome of the ontological reasoner as input. Adapters (also called *mediators*) that relate heterogeneous information are also investigated in the database area, cf. [Wie92].

2.4 Components and Dataflows of the IB

The layer specification of the functionality of the IB shall be realized by four software components and three shared ontologies (see Figure 2). Each software component is realized by an agent type: a *client agent*, a *provider agent*, an *ontology agent* and an *adapter agent*. This multi-agent-architecture has several advantages. The four different agents types can be developed and realized by different groups and running on different servers. Several client or provider agent (i.e., instances of the agent type) can be run in parallel (one for each current client and/or provider). Clients and Provider can develop their own agents if they fit into the conventions of the core components. Finally, also different core components can be developed by other groups.

The components of the IB use three different ontologies: domain ontologies (D_O), task ontologies (T_O), and PSM ontologies (P_O). These ontologies provide the contents of their communication processes. The communication between the agent should be realised by message passing. Because this communication includes the exchange of knowledge, it will be investigated whether KQML [FMF+94] can be used as communication means. In general, two general types of information flows exist. One is concerned with the process of finding appropriate PSMs and the other is concerned with actually solving the problem via a selected PSM. Each can be further distinguished by the contributing agents and the direction of the communication. In total, eight different communications can be distinguished

(see Table 1 for a survey). One additional communication type is an internal communication of the core

Table 1. Eight different information flows

Finding a PSM	Using a PSM
Sending a request from the client agent to the ontology agent. Terms of the domain and task ontologies are the content of the message. The client agent uses a *domain ontology* to guide the interaction process with the client and it sends selected expressions to the ontology agent.	Sending domain knowledge and case data from the client agent to the adapter agent.
Sending a negotiation from the ontology agent to the client agent. The ontology agent need further clarification from the client before it can finish the selection process of PSMs. This clarification may ask for more precise definitions of terms and their relationships necessary to derive from an element of the domain ontology an element of the task ontology.	Sending a solution from the adapter agent to the client agent.
Sending a query from the ontology agent to a provider agent. After having translated the domain-specific terminology into a task-specific terminology the ontology agent has to derive an expression in a PSM-specific ontology. Then, this expression is passed as a query to the provider agent.	Sending an input from the adapter agent to a provider agent.
Sending a response from a provider agent to the ontology agent. The response of a provider agent may have two forms: providing a simple yes that the required service can be provided or the wish to introduce further *assumptions* on the task that make the problem tractable and/ or the introduction of *requirements* on domain knowledge that has to be provided by the client (cf. [BFS96]).	Sending an output from a provider agent to the adapter agent.

that uses the ontological information derived by the ontology agent to construct or select an adapter that can provide the runtime mapping of terms as required by client and provider agents. The *ontology mapping information* (cf. Fig. 2) enables an adapter to realize the communication between clients and providers.

Gist of the matter of our IB are the ontology and adapter agents and the ontology libraries which will be discussed subsequently during the following.

3 The Ontology Agent and Its Ontologies

The ontology agent must provide service to construct and to connect different ontologies. First, it requires a *task ontology* that describes classes of generic application problems in domain-independent terms. This ontology provides definitions of classification problems, allocation problems, diagnostic problems, planning problems etc. (cf. [BV94]). This task ontology is used to negotiate with the client agent for building an application-specific domain ontology *and* with the provider agents for selecting a PSM via its PSM-ontology. Both aspects are discussed subsequently during the following:

- **Linking domain and task:** Support must be provided for the client to build a new or to reuse an existing *domain ontology* describing his problem (i.e., his requirements on a solution) and the available domain knowledge and to link this domain ontology with the task ontology of the ontology agent. Our approach does not make any assumptions about whether this application-specific domain knowledge is built from scratch or whether it is gained by adapting an already given domain ontology. Existing environments as Ontolingua [FFR96] can support these processes.
- **Linking task and PSM:** Besides linking domain and task, the ontology agent must link task and PSMs. He needs knowledge of how terms describing the competence of methods are linked with terms of the task ontology. These ontological connections have to be provided by the provider. They have to build their *PSM ontologies* and have to establish the links with the task ontology. Notice, that PSMs shall be described independent from a specific task they can be applied to. Therefore, they have to be linked explicitly to all tasks they can solve (cf. [BBH96]).

3.1 Agreement on Domain and Task Ontologies

In the following, we provide an illustration using a simplified version of the Sisyphus room-assignment problem [Lin94]. It describes a problem in which employees are assigned to office places. A GMD department is moving into a new building and the new working places should be distributed to the employees. We will provide different task ontologies to illustrate their scope and the process of connecting them.

In the first step, the client has to select the appropriate task from the set of all tasks. The task ontology (cf. [BV94]) describes generic problem types from which an appropriate one has to be chosen. Imagine that our task ontology provides besides others a task *assignment* (cf. Figure 3) and that the client was able to select this task definition.[3] He can now use the task schema as a skeleton to model his

TASK assignment[1]
CONSISTS OF

CONSUMER ≤	THING
RESOURCE ≤	THING
ASSIGNMENT ≤	RESOURCE x CONSUMER
ASSIGNMENTS ≤	**SET OF** ASSIGNMENT
CONSTRAINT ≤	**SET OF** ASSIGNMENTS
PREFERENCE ≤	ASSIGNMENTS x ASSIGNMENTS
COMPLETE ≤	ASSIGNMENTS
CORRECT ≤	ASSIGNMENTS
c ∈	CONSUMER
r ∈	RESOURCE
A, **Goal** ∈	ASSIGNMENTS

each resource can only be consumed once
$(c,r) \in A \wedge (c',r') \in A \wedge c \neq c' \rightarrow r \neq r'$
a complete assigment
$(c \in \text{CONSUMER} \rightarrow \exists\, r\,.\,(c,r) \in A) \rightarrow \text{COMPLETE}(A)$
a correct assigment
$\neg\, \exists\, C(C \in \text{CONSTRAINT} \wedge A \in C) \rightarrow \text{CORRECT}(A)$
the goal is complete and correct
$\text{COMPLETE}(\textbf{Goal}) \wedge \text{CORRECT}(\textbf{Goal})$
the goal is prefered
$\text{CORRECT}(A) \wedge \text{COMPLETE}(A) \wedge A \neq \textbf{Goal} \rightarrow \neg\, (\textbf{Goal},A) \in \text{PREFERENCE}$

ENDTASK assigment

1. The logical language we use is S-logic (i.e., Slogan level logic).

Fig. 3. The task ontology for simple assignments with one consumer type and one provider type.

Application ontology
ENRICH domain ontology, assignment
CONSUMER = EMPLOYEE
RESOURCE = WORKING PLACE
$employee_1$, $employee_2 \in$ CONSUMER
$place_1$, $place_2 \in$ RESOURCE
$(employee_1,place_1) \in A \wedge (employee_2,place_2) \in A \wedge$ same-room$(place_1,place_2) \wedge$
$\neg$ fit-together$(employee_1,employee_2)$
$\rightarrow A \in$ CONSTRAINT
smoker$(employee_1) \wedge \neg$ smoker$(employee_2) \vee \neg$ smoker$(employee_1) \wedge$ smoker$(employee_2)$
$\rightarrow \neg$ fit-together$(employee_1,employee_2)$
project$(employee_1)$ = project$(employee_2) \rightarrow \neg$ fit-together$(employee_1,employee_2)$
$(boss,place_1) \in A_1 \wedge (boss,place_2) \in A_2 \wedge$ centrality$(place_1) <$ centrality$(place_2) \rightarrow$ PREFERENCE(A_1,A_2)
END application ontology

Fig. 4. The connection of domain and task ontologies by an application ontology.

application-specific domain model. He links employees with CONSUMER and working places in office rooms with RESOURCE. Then, he has to formulate constraints on desired assignments. In his domain, a critical decision is to put two persons into the same office. Attributes that are critical are whether both employees smoke tobacco or not, and the aspect that both should work in different projects to support cross-fertilization. Based on these domain-specific circumstances, he defines his constraints. Finally it remains to define some preferences to distinguish two legal assignments. We use only one criterion of [Lin94] concerning the centrality of the working place of the boss of the group. The entire ontology mapping of task and domain is provided in Figure 4. Notice that these ontological negotiation process were done completely independent from any PSM-specific aspects. This aspect will discussed in the following.

3.2 Transforming the Task Ontology

We will now discuss the case where we cannot find directly a method for the selected task. However, we assume the existence of a method for the closely related tasks of parametric design (cf. Fig. 5.). Parametric design defines the task of designing an artefact by assigning values to a given set of parameters. Such an assignment must be complete (i.e., each value must have a parameter), correct (i.e., no constraint may be violated), and preferred (i.e., the solution should be optimal). This task can be transformed to express the assignment task of Fig. 3. We have to establish the links as defined in Figure 6. In general there are three possibilities to achieve such a transformation:

- The broker has such a strong ontological inference engine that it can derive the ontological mapping necessary to translate one task ontology into another.[4]
- The broker has the knowledge that one task ontology can be translated into the other and how this transformation can be achieved.
- The provider of PSMs annotate their PSMs with different types of tasks a method can be applied for.

Each solution poses the problem to a different group of tool provider (i.e., developers of inference engines for description languages, builders of the brokers, and the providers of PSMs).

3.3 Agreement on Task and PSM Ontologies

After establishing a link between domain and task one has to establish a link between task and PSM.

3. Clearly this process needs to be further supported by methods and techniques of the requirements engineering community.
4. Actually we are very pessimistic about this alternative as the expressive power of all existing systems is already smaller then what we used to define the tasks.

TASK parametric design
CONSISTS OF

PARAMETER ≤	THING
VALUE RANGE ≤	**SET OF** THING
VALUE RESTRICTION ≤	PARAMETER → VALUE RANGE
ASSIGNMENT ≤	PARAMETER → Y, Y ∈ VALUE RESTRICTION(PARAMETER)
DESIGN MODEL ≤	**SET OF** ASSIGNMENT
CONSTRAINT ≤	**SET OF** DESIGN MODEL
PREFERENCE ≤	DESIGN MODEL x DESIGN MODEL
COMPLETE ≤	DESIGN MODEL
CORRECT ≤	DESIGN MODEL
p ∈	PARAMETER
v ∈	Y, Y ∈ VALUE RESTRICTION(PARAMETER)
D, **Goal** ∈	DESIGN MODEL

a complete assigment
(p ∈ PARAMETER → ∃ v . (p,v) ∈ D) → COMPLETE(D)
a correct assigment
¬ ∃ C(C ∈ CONSTRAINT ∧ D ∈ C) → CORRECT(D)
# the goal is complete and correct
COMPLETE(**Goal**) ∧ CORRECT(**Goal**)
# the goal is prefered
CORRECT(D) ∧ COMPLETE(D) ∧ D ≠ **Goal** → ¬ (**Goal**,D) ∈ PREFERENCE
ENDTASK parametric design task

Fig. 5. The task ontology for parametric design.

Figure 7 provides the definition of the *propose & revise* method. It delivers a state that is complete, correct, and optimal. Finding such a state is usually an intractable problem (cf. [Neb96], [FeS96]). It is therefore necessary to negotiate additional requirements on domain knowledge and assumptions that limit the general scope of the task. Propose & revise can only solve such tasks with reasonable computational effort by introducing assumptions that restrict the complexity of the problem or by introducing requirements on domain knowledge [BFS96]. Two types of domain knowledge are required by the method (cf. Fig. 7.):

- PROPOSE-Knowledge that allows successive completion of design models.
- FIX-Knowledge that allows repair of incorrect design models.

The method can only be applied if these two types of knowledge are provided by a domain expert. Both knowledge types are further characterized by axioms that are necessary to guarantee that the competence of the method implies the task definition. How such axioms can be derived and proven is described in [FeS??a], [FeS??b]. These axioms have to be negotiated with the client and either stated as requirements on domain knowledge provided by the client or as weakening of the task. That is, the method does not guarantee to find a optimal solution or it does not guarantee to always find a solution even for cases where such a solution exist.

The link of the PSM propose & revise with the task parametric design that leads to the PSM propose & revise for parametric design is provided in Figure 8. Mainly two things have to be done: defining a

TASK MAPPING assignent -> parametric design
ENRICH task assignment, task parametric design
x ∈ VALUE RANGE
CONSUMERS = PARAMETER
x ∈ VALUE RANGE ↔ x = RESOURCE
VALUE RESTRICTION(p) = RESOURCE
ENDTASK MAPPING assignent -> parametric design

Fig. 6. The task mapping of assignemt and parametric design.

PSM propose & revise

STATE ≤	**SET OF** THING
PREFERENCE ≤	STATE x STATE
COMPLETE ≤	STATE
PARTIAL-COMPLETENESS≤	STATE → THING
CORRECT ≤	STATE
STATE-TRANSITION ≤	STATE → STATE
PROPOSE ≤	STATE-TRANSITION
FIX≤	STATE-TRANSITION
S, S´, OUTPUT ∈	STATE
P ∈	PROPOSE
F ∈	FIX

COMPETENCE

the output is a complete, correct and optimal state

COMPLETE(OUTPUT) ∧ CORRECT(OUTPUT) ∧

¬∃ S . (COMPLETE(S) ∧ CORRECT(S) ∧ (OUTPUT,S) ∈ PREFERENCE))

KNOWLEDGE REQUIREMENTS

the propose knowledge never fails and monotonically extends the state

¬ COMPLETE(S) → ∃ P. S ⊂ P(S)

the application of a propose leads to an optimal state

¬∃ S´ . ((P(S),S´) ∈ PREFERENCE ∧

PARTIAL-COMPLETENESS(S´) = PARTIAL-COMPLETENESS(P(S)))

the fix knowledge never fails

¬ CORRECT(S) → ∃ F. CORRECT(F(S))

the application of a fix does not change the completeness of a state

PARTIAL-COMPLETENESS(F(S))= PARTIAL-COMPLETENESS(S)

the application of a fix leads to an optimal design model

¬ CORRECT(S) → ¬∃ S´ . ((F(S),S´) ∈ PREFERENCE ∧ CORRECT(S´) ∧

PARTIAL-COMPLETENESS(F(S)) = PARTIAL-COMPLETENESS(S´))

ENDPSM propose & revise

Fig. 7. The PSM ontology of propose & revise.

state and defining partial completeness,

In general there are two possibilities to organise the connection between PSM and tasks. First, one can define a mapping as described in Figure 8 for each task that can be solved by a PSM. Second, one can describe a mapping between the PSM and one task and derive further mappings by expressing them as mapping between the two tasks. In the former case, it is the responsibility of the PSMs provider whereas in the latter case this service must be achieved by the broker itself.

PSM propose & revise for parametric design task

ENRICH propose & revise, parametric design

STATE = DESIGN MODEL

PARTIAL-COMPLETENESS ≤ DESIGN MODEL → **SET OF** PARAMETER

PARTIAL-COMPLETENESS(D) = {p | (p,v) ∈ D}

ENDPSM propose & revise for parametric design task

Fig. 8. The PSM ontology of propose & revise for parametric design.

4 The Adapter Agent

The necessity of adapters in the context of reusable components is discussed in [FeG97]. The description of an *adapter* maps the different terminologies of task definition, PSM, and domain model. It has two main purposes:

- Linking the domain ontology with the task ontology, and
- linking the task ontology with the PSM ontology.[5]

Because it relates the three other parts together and establishes their relationship in a way that meets the specific application problem they can be described independently and selected from libraries. Their consistent combination and their adaptation to the specific aspects of the given application (because they should be reusable they need to abstract from specific aspects of application problems) must be provided by the adapter. Adapter hardwire the results of the ontological reasoning that was necessary to establish the appropriate links between domain knowledge and PSM via a task ontology. In our example, we have the following mapping:

COMPONENTS = EMPLOYEES
$x \in$ VALUE RANGE $\leftrightarrow$ x = PLACES

Based on this we can derive an adapter that manages the mapping of domain knowledge on terms expected by the PSM. Propose knowledge and fix knowledge can now be provided as mappings between assignments of employees and places.

In addition to semantic information, an adapter needs information about the syntactical structure of the input and output that is used by the PSM. This syntactical information must be used to derive front-ends of the client agent that are used to communicate with the client. In the case, that large domain knowledge already exists and should be reused, the adapter must realize the mapping between the different syntactical conventions of domain and PSM.

5 Related Work

The general architecture of our broker system was taken from the MeDoc [BDG97], a German project on developing a broker for heterogeneous, distributed information sources in the internet. Our approach differs in regard to the core of the broker that is realized by ontological reasoning. The idea of using ontologies to annotate information in the WWW is part of the SHOE-approach [LSR96], [LSR+97]. HTML pages are annotated via ontologies to support information retrieval based on semantic information. We extend their idea for annotating software components (i.e., PSMs) that provide not only information but inference power.

Our adapters deal with two types of mappings: mapping domain knowledge on generic task definitions and mapping task definitions on PSM-specific terms. The importance of these mappings lead us to introduce this new modelling construct in addition to domain, tasks, and PSMs. Most other approaches deal with these mappings only as a side-aspect and focus on connecting domain and PSM directly (cf. [FvH94], [GTR+94]). Exceptions are [BBH96] and [SEG+96]: The former introduce the idea of de-coupling PSMs and task and the latter discuss a coupling mechanism between subtasks and PSMs. Adapters generalizes these approaches.

Mapping input and output of a PSM via a task on domain-specific data and knowledge can be interpreted as a mapping between different data schemas. The necessity of adapters or mediators

5. Therefore, an adapter allows domain and task-independent characterization and implementation of PSMs (cf. [BBH96]).

[Wie92] necessary to combine heterogeneous information is well-know in the database area. E.g. in the project TSIMMUS [CGH+94] *mediators* are provided to integrate heterogeneous information sources. An implementation of such a mediator can automatically be derived from its specification in the *Mediator Specification Language*. We will try to use this and similar ideas to automatically derive adapters from ontological information.

CORBA [OHE96] is a standard specification that describes how software components can interact across networks, languages and platforms, [GSM96] investigate the usefulness of applying CORBA to knowledge-based systems. CORBA can be used to define a standard of the syntactical form in which components can interact. Therefore, it covers one aspect that is handled by our adapters. However, it does not provide means to specify the semantics of building blocks and its data exchange nor does it help to find and adapt components.

Finally, the task ontologies we use to intermediate between domain knowledge and PSMs describe generic classes of problems (types) as assignment, parametric design, skeletal planning, model-based diagnosis etc. Work in requirements engineering also aims for such generic problem classes used to capture functional and non-functional requirements on a software artefact. [SuM94], [Mai96] distinguish 13 top-level object systems models: resource returning, resource supplying, item composition, etc. This top-level models become hierarchical refined through five levels each of them adding specific aspects to such a model. It looks quite promising for future work to compare and combine both lines of research.

6 Conclusions

We present the architecture of an intelligent broker that should bring PSM-reuse to work. The broker allows the application of PSM via the WWW. Ontological reasoning is used to select an appropriate PSM for a given problem and to derive the necessary mappings between domain-specific and PSM-specific terms.

Currently, we investigate several approaches to ontological modelling and reasoning (i.e., CARIN [LeR96], FLORID [KLW95], LOOM [Mac90], Ontolingua [FFR96], SHOE [LSR96]). However, our ontologies are not simple taxonomies but means to specify the competence and the assumptions of PSMs. Besides proving language support, developing ontologies of PSM and task-ontologies that are used to negotiate between customers and PSMs are the goals of our current research. Interesting approaches in this area is the task ontology of diagnostic task in [Tei97] and the PSM-ontology of parametric design in [MoZ96].

Acknowledgement. I would like to thank Richard Benjamins, Michael Erdman, Rudi Studer and Steve Fickas for helpful and inspiring discussions.

7 References

[AFS96] 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.

[BBH96] 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.

[Ben95] 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.

[BFS96] 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.

[BDG97] D. Boles, M. Dreger, and K. Großjohann: Konzeption eines Informationsvermittlungssystems für heterogene, verteilte Informationsquellen im Internet, EMISA, no 1, 1997. http://medoc.informatik.tu-muenchen.de/deutsch/projdescr/medocinfo.html

[BV94] J. Breuker and W. Van de Velde (eds.) (1994): *The CommonKADS Library for Expertise Modelling*, IOS Press, Amsterdam, The Netherlands.

[CGH+94] S. Chawathe, H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. Ullman, and J. Widom: The TSIMMIS Project: Integration of Heterogeneous Information Sources". In Proceedings of IPSJ Conference, pp. 7-18, Tokyo, Japan, October 1994. http://www-db.stanford.edu/tsimmis.

[CJS92] B. Chandrasekaran, T.R. Johnson, and J. W. Smith: Task Structure Analysis for Knowledge Modeling, *Communications of the ACM*, 35(9): 124—137, 1992.

[DFM??] S. Decker, D. Fensel, E. Motta, and Z. Zrahal: A Formalized Task Ontology for Parametric Design, in preparation.

[EST+95] 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.

[FeG97] 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.

[FeS96] D. Fensel und R. Straatman: The Essence of Problem-Solving Methods: Making Assumptions for Efficiency Reasons. In N. Shadbolt et al. (eds.), *Advances in Knowledge Acquisiiton, Lecture Notes in Artificial Intelligence (LNAI)*, no 1076, Springer-Verlag, Berlin, 1996.

[FeS??a] D. Fensel und A. Schoenegge: Assumption Hunting as Developing Method for Problem-Solving Methods, submitted.

[FeS??b] D. Fensel und A. Schoenegge: Specifying and Verifying Knowledge-Based Systems with KIV, submitted.

[FvH94] D. Fensel and F. van Harmelen: A Comparison of Languages which Operationalize and Formalize KADS Models of Expertise, *The Knowledge Engineering Review*, 9(2), 1994.

[FFR96] A. Farquhar, R. Fickas, and J. Rice: The Ontolingua Server: a Tool for Collaborative Ontology Construction, *Proceedings of the 10th Banff Knowledge Acquisition for Knowledge-Based System Workshop (KAW´95)*, Banff, Canada, November 9th - November 14th, 1996. http://ontolingua.stanford.edu/.

[FMF+94] T. Finin, D. McKay, R. Fritzson, and R. McEntire: KQML: An Information and Knowledge Exchange Protocol. In Kazuhiro Fuchi and Toshio Yokoi (Ed.), Knowledge Building and Knowledge Sharing , Ohmsha and IOS Press, 1994. http://www.cs.umbc.edu/kqml.

[GSM96] J. H. Gennari, A. R. Stein, and M. A. Musen: Reuse For Knowledge-Based Systems and COBRA Components. In *Proceedings of the 10th Banff Knowledge Acquisition for Knowledge-Based System Workshop (KAW´95)*, Banff, Canada, November 9th - November 14th, 1996.

[GTR+94] J. H. Genanari, S. W. Tu, T. E. Rothenfluh, and M. A. Musen: Mapping Domains to Methods in Support Reuse, *International Journal on Human-Computer Studies (IJHCS)*, 41:399-424, 1994.

[KLW95] M. Kifer, G. Lausen, and J. Wu: Logical Foundations of Object-Oriented and Frame-Based Languages, *Journal of the ACM*, vol 42, 1995. http://www.informatik.uni-freiburg.de:80/~dbis/flsys/.

[LeR96] A. Y. Levy and M.-C. Rousset : CARIN: A Representation Language Combining Horn Rules and Description Logics. In *Proceedings of the 12th European Conference on AI (ECAI-96)*, Budapest, Hungary, August 11-16, 1996.

[Lin94] M. Linster (ed.): Sisyphus ´91/92: Models of Problem Solving, *International Journal of Human Computer Studies*, 40(3), 1994.

[LSR96] S. Luke, L. Spector, and D. Rager: Ontology-Based Knowledge Discovery on the World-Wide Web. Proceedings of the *Workshop on Internet-based Information Systems*, AAAI-96 (Portland, Oregon), 1996.http://www.cs.umd.edu/projects/plus/SHOE/.

[LSR+97] S. Luke, L. Spector, D. Rager, and J. Hendler: Ontology-based Web Agents. In P*roceedings of First International Conference on Autonomous Agents 1997*, AA-97. http://www.cs.umd.edu/projects/plus/SHOE/.

[Mac90] MacGregor: *LOOM Users Manual*, ISI/WP-22, USC/Information Sciences Institute, 1990. http://www.isi.edu:80/isd/LOOM/.

[Mai96] N. A. M. Maiden: Acquiring Requirements: A Domain-specific Approach. In A. Sutcliffe et al. (eds.), *Domain Knowledge for Interactive System Design*, Chapman & Hall, 1996.

[Mar88] S. Marcus (ed.). *Automating Knowledge Acquisition for Experts Systems*, Kluwer Academic Publisher, Boston, 1988.

[MoZ96] E. Motta and Z. Zdrahal: Parametric Design Problem Solving. In *Proceedings of the 10th Banff Knowledge Acquisition for Knowledge-Based System Workshop (KAW´95)*, Banff, Canada, November 9th - November 14th, 1996.

[Mus89] M. A. Musen: *Automated Generation of Model-Based Knowledge-Acquisition Tools*, Morgan Kaufmann Publisher, 1989.

[Neb96] B. Nebel: Artificial intelligence: A Computational Perspective. In G. Brewka (ed.), *Essentials in Knowledge Representation*, Springer-Verlag, 1996.

[OHE96] R. Orfali, D. Harkey, and J. Edwars: *The Essential Distributed Objects Survival Guide*, John Wiley & Sons, New York, 1996. http://www.acl.lanl.gov/CORBA/.

[Pup93] F. Puppe: *Systematic Introduction to Expert Systems: Knowledge Representation and Problem-Solving Methods*, Springer-Verlag, Berlin, 1993.

[SEG+96] R. Studer, H. Eriksson, J. Gennari, S. Tu, D. Fensel, and M. Musen: Ontologies and the Configuration of Problem-Solving Methods. In *Proceedings of the 10th Banff Knowledge Acquisition for Knowledge-Based System Workshop (KAW´96)*, Banff, Canada, November 9-14, 1996.

[Ste90] L. Steels: Components of Expertise, *AI Magazine*, 11(2), 1990.

[SWA+94] 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.

[SuM94] A. G. Sutcliffe and N. A. M. Maiden: Domain Modelling for Resue. In P*roceedings of the 3rd International Conference on Software Reuse*, Rio de Janeiro, Brazil, November 1-4, 1994.

[Tei97] A. ten Teije: *Automated Configuration of Problem Solving Methods in Diagnosis*, PhD thesis, University of Amsterdam, 1997.

[THW+93] P. Terpstra, G. van Heijst, B. Wielinga, and N. Shadtbolt: Knowledge Acquisition Support Through Generalised Directive Models. In M. David et al. (eds.): *Second Generation Expert Systems*, Springer-Verlag, 1993.

[Wie92] G Wiederhold: *Mediators in the Architecture of Future Information Systems*, IEEE Computer, 25(3):38-49, 1992.