A Conceptual Modelling Framework for Knowledge-level Reflection

Martin Reinders[1], Erik Vinkhuyzen[2], Angi Voß[3], Hans Akkermans[2], John Balder[1], Brigitte Bartsch-Spörl[4], Bert Bredeweg[2], Uwe Drouven[3], Frank van Harmelen[2], Werner Karbach[3], Zeger Karssen[2], Guus Schreiber[2], Bob Wielinga[2]

1: Netherlands Energy Research Foundation ECN; 2: SWI, University of Amsterdam; 3: National German Research Centre for Computer Science GMD; 4: BSR Consulting. Conctact: Martin Reinders, ECN, P.O. Box 1, 1755 ZG Petten, The Netherlands. reinders@ecn.nl

In this paper we develop a conceptual modelling framework for knowledge-level reflection (KLR), i.e, the modelling of tasks that require a self-representation of a knowledge system's own object-level problem solving tasks. This framework builds upon the KADS methodology for knowledge acquisition and design of knowledge systems [Hayward *et al.*, 1987, Wielinga *et al.*, 1991].

We argue for the separation of object and reflective problem solving levels and a self-representation that is distinct from the object-level because it is selective, specialised and knowledge oriented, i.e., it is a knowledge-level model congruent with the KADS conceptual model of the object system. As an example we describe a conceptual model for competence assessment and improvement in Office Plan, a configuration system for office space allocation. A broad comparison with notions of reflection in logic and computational reflection clarifies the distinctiveness of our notion of knowledge-level reflection and investigates some of the architectural options that are open for its realisation in knowledge systems.

Introduction

The overall goal of the ESPRIT basic research action REFLECT is to lay foundations for the construction of a class of second-generation expert systems that are knowledgeable of the limits of their competence and are more flexible in the ways in which they can employ their knowledge. The fundamental premise that underlies this project is that such more advanced systems can be realised by creating *reflective systems*. In this project a novel approach is taken to reflective systems, which takes as its basis the KADS methodology for knowledge modelling [Hayward *et al.*, 1987; Wielinga *et al.*, 1991].

Reflective systems incorporate reflective tasks, tasks that require an abstract model of other problem solving tasks. In this paper we give an example of a reflective task for competence assessment and competence improvement. Other examples of reflective tasks are for instance task-oriented explanation,

The research reported here was carried out in the course of the REFLECT project. This project is partially funded by the Esprit Basic Research Programme of the Commission of the European Communities as project number 3178. The four partners in this project are mentioned in the title above.

model-based knowledge acquisition and dynamic task configuration.

The first objective of REFLECT is to develop a conceptual framework for the *modelling* of reflective reasoning. This in contrast with frameworks for reflective *architectures* or *implementations*. This framework should constitute an important extension of the present KADS methodology for knowledge modelling. As a second objective, aided by this framework, specification concepts and a ske leton architecture for a class of reflective systems are to be developed. This paper focuses on the first of these objectives, the development of a conceptual structure for modelling reflective systems.

Overview of this Paper

In the next paragraph we briefly describe some of the standard terminology from computational reflection that we will adopt as a basis for our framework. Then we introduce the notion of knowledge level reflection and we focus on how we view the structure of a conceptual model for reflective systems. In particular, we describe the ingredients of such a structure, and how these ingredients are put together. In the paragraph "Example, An Assignment Application", we present a conceptual model of a reflective task in a configuration domain. Finally, we relate the notion of knowledge-level reflection to notions of reflection in cognition, logic and computational reflection.

Terminology in Reflection

In this section we briefly introduce some terms we adopt from the literature on computational reflection, in particular [Maes, 1987; vanHarmelen, 1989].

Reflection in computational systems refers to the act of switching from reasoning about an application domain to reasoning about the current reasoning of a system. A *reflective system* performs that activity. In order to represent the system *itself*, a reflective system has a *meta-level architecture*, i.e., the part of the system that is reasoned about, the object-level, is represented or modelled in a causally connected manner at the meta-level (the self-representation). Note that *meta* is used in the meaning of "about" something: the object. Hence a *meta-level* is a level of a system that is about another level of the same system, the *object-level.*

A *causal connection* is informally described in [Maes, 1987] as a link between a representation and a system that it represents, such that if one of these changes, the other changes accordingly (the representation is still truthful). In this context [Smith, 1986] introduces the notions *introspective integrity* and *introspective force*. Introspective integrity involves the question whether any significant property of the representation at the meta-level is in accordance with its content (the represented object-level). Introspective force involves the realisation of a system's reflective goals in the object-level through the causal connection downwards.

Knowledge-Level Reflection

Knowledge-level models provide an abstract, implementation-independent description of a system. They describe the behaviour of a system in terms of the goals, actions, and mediating knowledge of the system [Newell, 1982]. Our hypothesis is that theories about this abstract model of a system constitute a particularly interesting form of reflection. For example, it requires knowledge *about* the limits of one's knowledge to decide whether one can solve a problem. Knowledge engineers typically possess this kind of meta-knowledge about the systems they build. In REFLECT our aim is to make this knowledge explicit, so that the systems are themselves capable of reasoning about the limits of their competence, thereby considerably enhancing the flexibility of their problem solving behaviour. We shall call this type of reflection, in which the meta-system reasons about an abstract model of the object system, *knowledge-level* reflection.

Figure 1 gives a diagrammatic representation of

knowledge-level reflection. The key point is that the self-representation of the object system is an *abstract* model of the object-system. This is in contrast to computational reflection, where the self-representation concerns operational bookkeeping data and implementation details, and with logical reflection, where the self-representation is a purely syntactic mapping of sentences onto names (see paragraph "Comparison", p 82).

Based on this characterisation of a reflective system as a system that realises reflection through employing an abstract model as its self-representation, we will explore the structuring principles that are applicable to the conceptual modelling of such systems.

The Structure of a Reflective Model

The two basic tenets of our approach to knowledge-level modelling are: *abstraction* and *differentiation.* A knowledge model abstracts from implementational and operational matters and focusses instead upon the reasoning goals, tasks and mediating knowledge of an intelligent agent. These are conceptually described in a differentiated fashion, by distinguishing within the model a variety of knowledge types and components that have their own inherent structure and specific role in the reasoning process. Common to all knowledge-level approaches (e.g. [McDermott, 1989], [Chandrasekaran, 1987], [Musen, 1989]) is the intuition that it is possible to identify recurrent and relatively stable types and components of knowledge that are generic for a wide class of domains and/or tasks. Elsewhere we have worked out this viewpoint on the knowledge level and its implications in some more detail [Schreiber *et al.* , 1990].

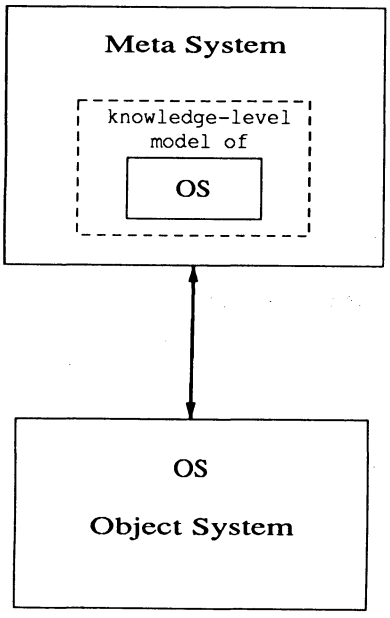

Figure 1. Knowledge-level reflection: a meta system uses a knowledge level model to reason about an object problem solver

Re-use in Reflection of Standard Conceptual Models

A conventional knowledge-based system embodies a theory about some real-world domain that it utilises for problem solving. In addition to this real-world theory, a reflective system embodies a theory that has the conventional problem solver as its object or domain. Apart from having such a special domain, a reflective theory is a theory like any other. This is a central observation since it implies that the structuring principles used in standard knowledge modelling also apply to reflective reasoning. This idea seems attractive and helpful, because it approaches reflective reasoning simply as yet another problem-solving task (just as diagnosis or design tasks are) and so reduces the exotic and magical flavour that surrounds the concept of reflection. In particular, we suggest that a KADS-like approach can also be employed for knowledge-level reflection. KADS provides a general modelling language for the conceptualisation and description of knowledge-based systems. Since our framework relies strongly on t his language, we will briefly describe its main ingredients.

The KADS Four Layer Model

The basic framework of the KADS methodology for knowledge-based system developme nt distinguishes four layers of knowledge types:

- a *domain layer* containing domain concepts and relations; These concepts and relations are represented declaratively, independently from

2: The term meta-class is an inheritance from the KADS project, and does not cary any "meta-" connotations in sense of object-meta system, reflection, etc.

the use that will be made of them in the infer ence process;

- an *inference layer*consisting of basic inference steps (or knowledge sources) and meta-classes[2]. The knowledge sources describe what the basic inference steps are that can be taken in the infer ence process, but no order is yet implied among these steps. The meta-classes describe the role that various domain expressions will play during the inference process;
- a *task layer*containing a task-decomposition, where each task employs methods for the control and application of knowledge sources. Thus, it is only at this layer that control is en forced on the execution of the basic inference steps.
- a *strategy* layer that decides on the application and order of tasks. In traditional KADS this often meant a fixed task structure. In REFLECT the strategy layer is seen as part of a meta-level problem solver, which is however not restricted to pure strategic reasoning.

Separation of reflective and object-components

At the highest level of aggregation, a first and very natural distinction to be made within a conceptual model for knowledge-level reflection is the differentiation into an object-part (the conventional problem-solving theory that the system has about the real-world domain) and a meta-level part (the reflective theory proper). Thus, the object and reflective parts are seen here as two distinct systems, each of which corresponds to a conventional conceptual model. This is in contrast to some approaches in logic and computational reflection that aim at a full description within a single context and language (see paragraph "Comparison", p. 82).

The nature of the self-representation

By defining knowledge-level reflection as a form of reflection where the meta-system has an abstract (knowledge-level) model of the object system, the self-representation is necessarily separate from the object system. However, a distinction must be made between the conceptual model of the object system as it occurs in the reflective framework, where it is an abstract model, and the symbol-level implementation of the modelled object system. With respect to the conceptual model of the object system, the self-representation is:

- *selective* : In order to reduce complexity, the re flective components should begiven partial knowledge of and access to only those parts of the object-system that are really necessary for performing their reflective task.
- *specialised towards the reflective task* : The complexity of reflective reasoning can be further reduced by making the self-description of the system better tailored to the reflective task. In other words, the nature of the self-representation is *relative to the reflective theory* that uses it. This implies that it is possible for a reflective system to have *different* meta-models of the

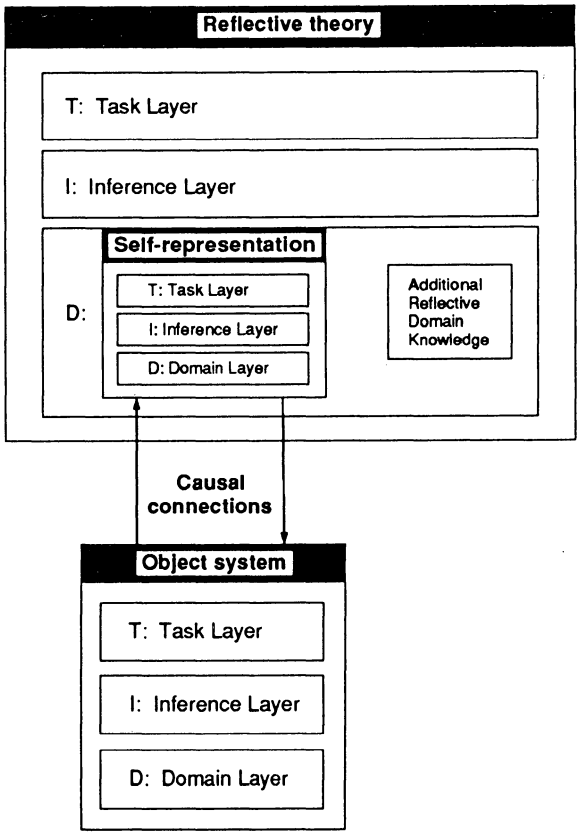

Figure 2. The structure of a conceptual model for knowledge-level reflection

same object-level system, being associated to different reflective tasks.

With respect to the symbol-level object system, the self-representation is also:

- *knowledge-oriented:* To make the selfdescrip tion more useful to the reflective theory it is 'annotated' with indicators concerning the knowl edge roles fulfilled by the object-system. This concept is formally represented as "meaningful naming" in [Akkermans *et al.* , 1990]. In this way, the self-representation provides explicit conceptual handles for use in the reasoning by the reflective components, in contrast to the usual approach, where object-knowledge is described in purely syntactic terms (see paragraph "Comparison", p. 82).

The self-representation is a partial knowledge-level model

The above constitutes our central argument why it is useful for the self-representation of a reflective system to be a *partial* knowledge-level model. The distinction between the object-system and the meta-model of the object-system proposed here is in our opinion important because it enables this model to be different from and more meaningful than just a simple and complete image of the system it describes, by representing only those aspects of the object system that are relevant for the particular reflective reasoning task. In our framework, the self-representation is congruent with the KADS conceptual model of the object system, i.e., it differentiates tasks, inference structures, knowledge sources and metaclasses as indicated in Figure 2.

Knowledge differentiation in reflection

We have stated earlier that a reflective theory is a theory like any other, that is, a reflective component is seen as a normal problem solver for a certain kind of reasoning task. This suggests that reflective components can be conceptually modelled on the same basis as is the case for, say, diagnosis or configuration problem solvers. Accordingly, we suppose that the reflective component can also be modelled by means of a KADS conceptual model. The self-representation of the reflective system would then be part of the domain layer of the reflective component. This is in line with the intuitive idea that the reflective component operates 'on top of' the self-representation the system has.

At the reflective domain layer, we distinguish several other kinds of knowledge that are relevant for reflective tasks in addition to the self-representation:

- Knowledge about problem solving tasks in general;
- Knowledge about the object-system's tasks, e.g., whether a task is complete or not or its estimated costs;
- Additional domain knowledge. This concerns additional knowledge that a reflective task requires about the application domain, for example the importance of particular constraints for relaxation of a constraint problem.

In this section we have indicated how each of the three major components of a reflective model — the reflective component, the object-system and the self-representation — can be differentiated. One important observation is that the conceptual model of the reflective system and the conceptual model of the object system are two abstract models that constitute outside descriptions of the system, but that the self-representation is a model ascribed to the system itself, which is one of the reasons that we strive for a structure-preserving implementation (see below).

The 'structural correspondence' principle

A conceptual model need not be recognisable as such in the implementation

It is important to realise that the various parts of the conceptual model of reflection, as outlined in this section, are not necessarily mirrored in the physical realisation or implementation as separate, recognisable data structures. The fact that any reflective system can be thought of as having a self-representation, should not be taken as implying that a particular system actually has to be built in that way. In fact, the architectures of many reflective systems that

have been built in the past do not conform to this description. All we need for the applicability of our conceptual model is the fact that reflective systems *can be usefully described* in terms of the present structured conceptual framework for knowledge-level reflection.

Preserving the structure of the conceptual model in the implementation

On the other hand, it is possible to make the choice that the system architecture of a reflective system preserves the structure of its corresponding conceptual model. This we will call the *structural correspondence principle.* For the purposes of the REFLECT project we have decided to adopt and investigate this principle, because having a structural correspondence between the conceptual model and the reflective architecture seems to be of interest for various reasons:

- The well-known problems associated with the explainability, construction and maintenance of knowledge-based software are alleviated if the implementation preserves the structure of the conceptual model. Since this is already the case for standard knowledge-based software, it will even more strongly apply to reflective systems.
- The structured approach we have developed helps to introduce role differentiations and access limitations of knowledge also in the case of reflective reasoning. As argued in paragraph "Comparison", p.82, this is seen as a possible way to make reflective reasoning computationally more tractable.
- The causal connections between the self-representation and the object system require that the object system is implemented in a structure-preserving manner. If not, it could be virtually impossible to attach the self-representation and the object system.

Finally, it is to be stressed that the present conceptual framework is still tentative. To investigate it further, experimental work with reflective modules will be carried out. It will be easier to have useful feedback from these experiments for the purpose of improving our conceptual model for reflection, if the reflective system mirrors the conceptual structure.

Example: An assignment application

In this section we discuss an example that uses the structure sketched above to specify a reflective system in a configuration domain. For this application, OFFICE-PLAN, an office room allocation problem, we specified several reflective modules for competence assessment and improvement. Here we will describe the common pattern they obey.

The object system OFFICE-PLAN

Planning the arrangement of employees on a floor is a time-consuming process, especially if personnel movement is high as in research institutes. To reach a satisfactory solution all criteria for a fertile working climate should be considered, e.g. dense communications between projects, proximity of central services, equipment requirements and personal characteristics like smoker aversion ([Karbach *et al.* , 1989, Karbach *et al.* , 1990]).

Our object system OFFICE-PLAN treats the problem as an assignment problem where a set of objects, the employees, have to be assigned to another set, the rooms, so as to satisfy the given requirements. The system matches requirements with employees obtaining a constraint network with employees as variables and possible rooms as their values. Global propagation followed by a filtering yields all solutions, possibly none in case the problem is overspecified.

OFFICE-PLAN does not behave very competently. This incompetence can manifest itself in a number of ways: OFFICE-PLAN takes too much time to solve complex problems and it finds no solution for overspecified ones. In case of missing knowledge, OFFICE-PLAN always assumes the best case, and it does not detect inconsistent nor redundant requirements ([Voß *et al.*, 1990]), we have developed reflective modules to deal with these types of incompetence. These modules analyze and cope with overcomplex and overspecified, inconsistent and redundant problems. Subsequently, we will sketch the

conceptual model we used to specify these modules. Special details will be given for a module tackling contradictory constraints.

The reflective task knowledge

Every module tackling some type xy of incompetence, essentially performs a combined analyze and repair task:

```
task tackle-xy
  subtasks:
    OS-start-xy
    analyze-xy
    if malfunction = xy then

propose-repair-xy
    apply-repair-xy
    OS-finish-xy
```

The exact moment of when to analyze and repair depends on the type xy of malfunction being treated. For instance, we have a module recognizing inconsistent requirements in the original problem statement, and another one recognizing the finer grained inconsistencies in the internal constraint network. The first one is activated after the problem statement has been read in, while the latter interferes only after the problem has been transformed into a constraint network.

The reflective inference knowledge

All our reflective modules obey a common inference structure, which is shown in Figure 3. Usually, not the entire conceptual model of the object system is required, and often an abstraction into a more handy data structure may be convenient, to be represented in metaclass *abstraction of object system.* It is *analyzed* to produce certain *findings* like a comparison of required and available resources, a quantitative assessment of the complexity of the problem, or the number of similar cases. The findings may be *interpreted* as *malfunctions* like "overcomplex problem" or "inconsistent problem". Malfunctions are usually referenced on the task layer to determine whether a repair is needed (c.f. `malfunctions = xy` above). Based on the findings, some *repair*actions may be *proposed.* — We separated the proposal of repairs from their actual *application* , because in the integrated system o be built, different reflective modules may propose conflicting repairs such that not all of them can be applied. — The metaclass *experience* allows the module to *update* its knowledge and use it for analyzing and proposing repairs. For instance, we have a reflective module that decomposes a problem into a sequence of constraint networks. This module accumulates the solutions of already solved subproblems in metaclass *experience.* For yet another module, metaclass *experience* represents a complete case library, used to retrieve solutions of similar, previous cases to simplify the case at hand.

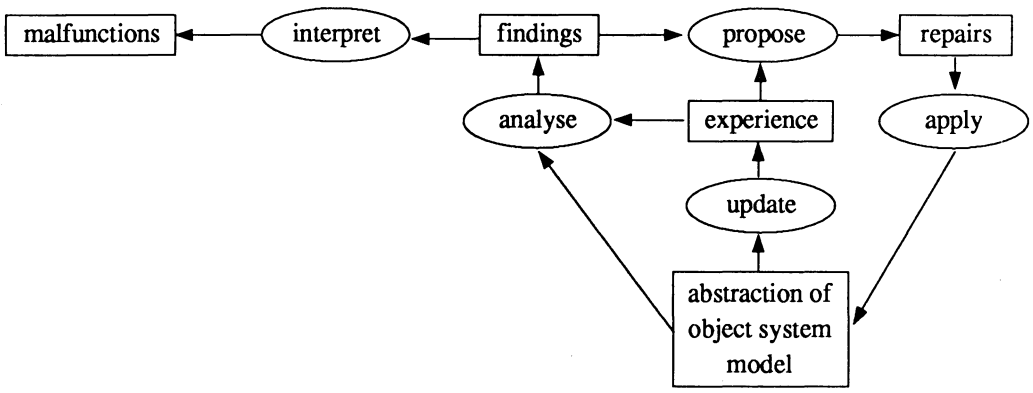

figure 3. An inference structure for reflective competence assesment and improvement

Figure 4 shows the inference structure for the module tackling contradictory constraints. It simplifies the general scheme as there is no experience being accumulated. Besides, the model of the object system is reduced to an *internal-problem-description* to represent the constraint network. Knowledge source *analyze-contra-c* determines all pairs of inconsistent constraints, and *propose-contra-c* removes one of each. *Apply-contra-c* removes the proposed constraints from the network. *Interpret-contra-c* signals an overspecified problem if any pairs of contradictions have been found. In order to determine the pairs of contradictory constraints, *analyze-contra-c* uses a binary formal relation *opposite.*

The reflective domain knowledge

The reflective domain layer contains the conceptual model of the problem solver, i.e. OFFICE-PLAN, and the additional knowledge referenced by the reflective knowledge sources. In the example of tackling contradictory constraints, this is a relation *opposite-constraints* , the concrete counterpart for the formal relation *opposite.*

Opposite-constraints can be defined extensionally, by listing all pairs of contradictory constraints, or intensionally, by using two additional relations and inspecting the definitions of the constraints. One of the additional relations determines which relations OFFICE-PLAN uses as constraints, and the other determines the arity of a relation. To connect the domain layer to the generic layers above, we have to connect the formal task-names *OS-start-contra-c* and *OS-finish-contra-c* with the corresponding knowledge sources from OFFICE-PLAN, metaclass *internal-problem-description* with the OFFICE-PLAN metaclass containing the constraint network, and the formal relation *opposite* with *opposite-constraints.*

Evaluation

Up to now we have developed ten individual reflective modules on top of OFFICE-PLAN. All of these have been implemented along the same scheme as the module for contradictory constraint discussed above.

These modules are able to detect unsolvable problems, assess the complexity of problems, remove redundancies, cope with overcomplex problems and handle time-limitations for the object system by assigning dynamically time slots for substasks. OFFICE-PLAN improves significantly when supported by reflective modules. It does not start to solve in principle unsolvable problems, in some experiments the size of the constraint network was reduced up to 30% by removing redundancies and the time spend to solve problems was reduced up to factor 10 applying decomposition strategies and employing explicit time management of subtasks. The reflective architecture provided a modular way to enhance the competence of OFFICE-PLAN. Apart from that,

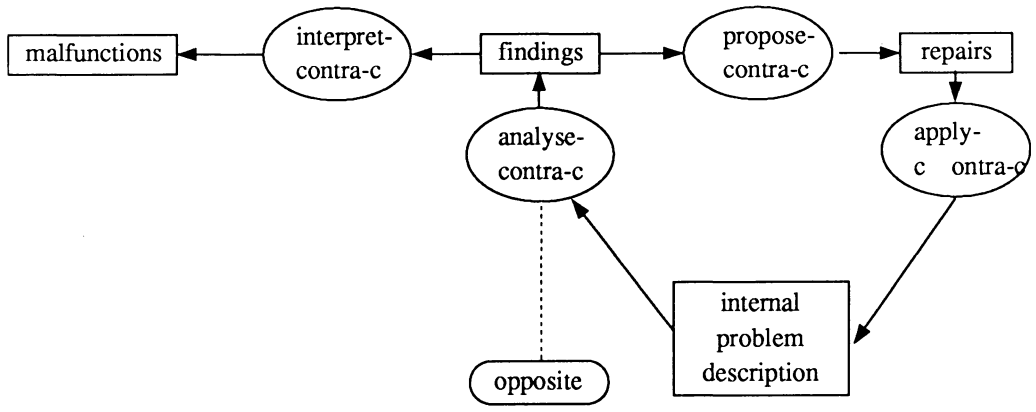

Figure 4: An inference structure for tackling contradictory constraints

much of the knowledge conserved in the reflective modules can be reused to reason not only about a particular problem solver but also about a class of systems performing assignment tasks.

Comparison with Other Notions of Reflection

In the previous sections we explicitly presented the concept of knowledge-level reflection as distinct from other notions of reflection, based on its usage in conceptual modelling rather than formalisation or implementation. In this section we will discuss the similarities and dissimilarities between the various notions of reflection as occurring in cognitive science, logic and computer science and we interpret some selected architectures in our conceptual framework. This broad survey will show that there is indeed sufficient reason for our newly introduced concept of knowledge-level reflection. The section will conclude by summarising the distinctive features of knowledge-level reflection.

Comparison to cognitive notions

When discussing knowledge-level reflection in relation to cognition, a crucial observation to be made is that the present work has no cognitive-science ambitions whatsoever. It does not intend to make any claim with regard to cognitive modelling. Instead, our aims are directed towards engineering goals: to develop a conceptual basis that enables the construction of more powerful and flexible intelligent systems. Notwithstanding this, however, it is interesting to notice that there appear to be similarities with ideas concerning meta-cognition as developed in cognitive science, in particular the information-processing theory of human problem solving.

This compatibility is manifested particularly in the following two aspects:

- The idea that complex high-level reasoning can be laid out in terms of different components and meta-components that are distinguishable on the basis of the function they perform in the overall information-processing system.
- The importance that is attached to the notion of access to the various components for the overall performance of the system.

Especially the work of Sternberg [Sternberg, 1982b] emphasises the multi-component perspective and employs the distinction between on the one hand components for performance, acquisition, transfer and retention and on the other hand meta-components. At a more detailed level however, there seem to be quite some differences with our framework. Sternberg's performance components, that are used in the execution of a reasoning or problem solving strategy, may probably be equated to what we call the object-level problem solvers. In our framework components equivalent to his acquisition and transfer components are absent. This is only natural for the acquisition (*i.e.* , learning) components, since learning is beyond the present scope of our reflective framework. Transfer components, that are used in generalisation, i.e., transfer of task knowledge, are also absent in our case, but their role is taken over in a completely different fashion by our emphasis on and use of the *generic* character of knowledge types and components. This is a pervasive notion in knowledge-level modelling in general, and it is made applicable to reflection in the present work.

Sternberg's meta-components are restricted to the selection and maintenance of a control strategy for problem solving. These strategic control issues are only a subclass of the functions that the reflective level is thought to perform in our framework. His retention components, processes used in retrieving previously stored knowledge, are a combination of what we would classify as a certain type of object problem solvers (in particular knowledge organised in case libraries), with types of meta-knowledge (such as knowledge about problem solving) that we would rather situate as being part of the reflective level. Thus, the demarcation between object-level and meta-level components is different from our framework. Most importantly, the nature of the self-representation is not given any prominent place, in contrast to the situation in knowledge-level reflection. In sum, although there is a global similarity, the structural decomposition of the model is quite different.

The notion of access is developed and applied in [Pylyshyn, 1978] and [Campione *et al.*, 1982]. They introduce a distinction between *multiple access,* the ability of flexible use of knowledge, and *reflective access,* the ability to mention as well as use components. Thus, both forms of access imply the capacity to use knowledge flexibly, but in multiple access the associated knowledge remains implicit and is thus not describable, whereas in reflective access it can be stated explicitly. The notions of multiple and reflective access can be applied to give an interesting insight into the KADS strategy layer, and into the nature of strategic knowledge in general. There are two different ways to conceptualise the strategic level:

1. The strategic level can be conceptualised as be ing of the multiple-access type. This means that it essentially consists of knowledge (for instance concerning external circumstances, internal in put-output conditions, repair measures) neces sary for the planning and control of the execu tion of a relatively wide collection of task struc tures. This relatively wide applicability ensures the flexible use of individual tasks. This seems to be a standard view, and it is also the tradition al KADS interpretation.
2. An alternative interpretation, however, is to consider the strategic level as a form of reflec tive access. In this view, the strategic level does not directly operate on (or in) a subsystem that takes care of a certain task, but instead on a cau sally connected metamodel of such a system. Hence, it acts here as a reflective component that has statable knowledge of the domain-, infer ence- and task-content of the underlying system. In this second interpretation the overview role that is intuitively ascribed to the strategic level is made much more explicit.

A tentative idea we want to bring forward here — also being implied or suggested in some cognitive work — is that the second approach of having reflective access makes it possible for strategic knowledge to become less specific or 'welded' to the tasks it happens to be applied to, because the relevant knowledge has been made explicit and statable. This describability makes knowledge more easily transferable and modifiable. Hence, a reflective approach might lead to greater flexibility and adaptivity of systems.

In conclusion, there is a resemblance between our framework and notions from meta-cognition, particularly with respect to the idea of separating components and metacomponents. The concepts of reflective and multiple access can be fruitfully applied to knowledge-level reflection. At a more detailed level there are many differences related to the distinctions between and the organisation of components.

Comparison to notions from logic and computation

With respect to the logical and computational work on reflection, our work also has a different goal. It is our aim to develop a practical framework for certain common sense-like problem-solving modes that focus on problem solving competence, rather than entertaining reflective computation *per se* or developing new logics of (self-)knowledge. This difference in perspective gives (although it is pre-theoretic) rise to specific technical choices and differences with notions from logic and computer science.

Reflection in logic is concerned with the representation of truth, provability and belief. These logics employ a single language and do not possess an internal structure. The self-representation is complete and direct, realised in first order logic by the naming of formulae by ground terms. The investigation of the consistency of self-referential statements is a prominent issue and concentrates on avoiding the fundamental negative results of [Gödel , 1931; Tarski, 1936] (incompleteness, inconsistency) by introducing variants of the truth predicate ([Kripke, 1975; Perlis, 1985]) or iterated extensions of the axiom set via reflection principles [Turing, 1939; Feferman, 1962].

We have opted for a *multilingual* logical framework that *separates* languages and theories instead of amalgamating them. In particular, we separate the object- and meta-levels, thus avoiding self-referential problems altogether. This choice of logical

framework can be traced back in AI to the multi-level and multi-context framework of FOL [Weyhrauch, 1980]. In our view, such a framework is more suitable for expressing conceptual structures within knowledge, also with respect to reflection. Such an approach also leads to the consideration of the use of reflection principles for inter-theory inference. Here, the interest is in so-called enlarged reflection principles (a term adopted from [Giunchiglia & Smaill, 1989]) meaning that the requirement of truthfulness is dropped, for modelling the inferencing between meta- and object-theories.

A second major departure lies in the nature of the self-representation and of the naming relation. For reasons extensively discussed in paragraph "The structure of a Reflective Model" (page 76), the self-representation we intend to use is selective, specialised and knowledge-oriented: it is a partial and abstract model of the object-system. This is in contrast to the commonly encountered complete and direct self-representation. As a consequence, also the employed naming relation has a different character. Instead of using for example quotation names or structural-descriptive names, we have introduced a so-called *meaningful naming relation* [Akkermans *et al.*, 1990] that is able to express knowledge types and roles of object-level sentences and theories.

These logic-oriented differences also have implications with respect to frameworks for reflective computation:

- Due to the multilingual approach, our interest is not so much attracted towards a so called meta-circular interpreter, i.e., an explicit representation of the interpreter in the language itself, which is also actually used to run the language. Applying a meta-circular interpreter implies that the language at each level of the tower must have the same structure. Rather than this meta-circular approach, our multilingual approach is directed to a low number of semantically rich levels (essentially what we call 'first-order reflection', i.e., the object-system is not a meta-system of another object-system) and to 'localised' types of computation.
- In the same vein, our framework concentrates upon forms of declarative instead of procedural reflection [Maes, 1987], whereby the causal connection is given by what Maes calls a specification relation rather than an implementation relation via a meta-circular interpreter.

Notwithstanding the differences with knowledge-level reflection as pointed out above, it should be stressed that many individual ideas from logic and computer science remain important for our purposes and are taken over in our framework. Examples are the notions of bidirectional causal connections, reflection principles for intertheory inference, a naming relation between object-level structures and their representation at the meta-level, *et cetera.*

In conclusion, in developing the concept of knowledge-level reflection we have made some very specific choices with respect to the broad range of reflective frameworks that has been made available as a result of work in logic and computation. Our conceptual idea of separating multiple components and adhering to the 'structural correspondence principle' leads in formal terms to a multilingual and multilevel system, in contradistinction to the mainstream of logical and computational work on reflection. Another basic difference is the abstract (knowledge-level) character of the self-representation and the associated notion of meaningful naming.

Conceptual Interpretation of Reflective Tasks in Existing Architectures

It is interesting that many existing architectures for knowledge-based systems can be conceptually modelled in the current framework, i.e., they can be ascribed a (mixed) knowledge level model of problem solving tasks and reflective tasks operating on that model, although this is not necessarily made explicit in the architecture. For example NEOMYCIN/HERACLES [Clancey, 1985] is a rule-based expert system for heuristic classification that distinguishes abstract tasks and inference procedure rules (somewhat confusingly called metarules) from domain knowledge, in a way similar to the KADS inference and task layers. The explanation facility in HERACLES can be seen as a reflective task that reasons about tasks, meta-rules and additional explanation heuris-

3: This interpretation of SOAR's capability to deal with impasses as a reflective task should be distinguished from the interpretation of universal subgoaling as a form of computational reflection in [Rosenbloom *et al.*, 1988].

tics in order to provide procedural, focused and layered explanations, in contrast to merely printing the applied domain rules as in MYCIN. Also it enables strategic modelling of a student's behaviour. The implicit model that is used consists primarily of the metarules. The crucial difference between the use of metarules in the abstract tasks and as a model is that between actually interpreting them in order to perform diagnosis tasks and reasoning about them in order to explain that tasks.

Architectures for model-based knowledge acquisition, for example MOLE [Eshelman, 1989] and ASK [Gruber, 1989] use a knowledge-level model of object problem solving tasks and the role of knowledge therein in order to guide and constrain knowledge acquisition. In particular MOLE dynamically acquires missing knowledge based on an analysis of the role of knowledge in the cover-and-differentiate method for diagnosis. In a similar fashion ASK acquires refining strategic knowledge (about when to apply actions such as clinical tests) by considering the role of choices and justifications in reactive planning. Thus model-based knowledge acquisition can be seen as a reflective task operating on a model of object problem solving and distinguishing knowledge roles and knowledge deficiencies.

As a last example, SOAR [Laird *et al.*, 1987] employs a fixed method for impasse repair by subgoal creation, called universal subgoaling, that can be ascribed a model of tasks and that makes explicit a fixed set of impasses resulting from incomplete or inconsistent information. This conceptual interpretation of impasse repair as a reflective task comes close to the example for competence assessment and improvement in the paragraph "An Assignment Application" (page 79)[3]

However there are striking differences. In our framework we consider several categories of malfunctions that are different from or more specific than the fixed set of impasses in SOAR. The knowledge deficiencies resulting in these malfunctions are specific for certain object problem solving tasks. As the example in the above mentioned paragraph indicates, methods for analysis and repair also depend on the specific object tasks. Thus the reflective tasks depend on a specific type of domain (read: the object problem solving methods) as much as standard problem solving tasks depend on a particular type of domain theory. In contrast the impasse-repair method in SOAR is general and fixed, which is possible because of its dedication to the problem-space paradigm.

What this shows is that the conceptual framework is sufficiently general to capture different forms of reflective behaviour that can be desired in knowledge systems, such as explanation, knowledge acquisition and impasse repair. That several existing architectures can be described in this manner implies that there is not necessarily a single architecture that can be derived from this conceptual framework. The restrictions imposed above on the limited number of levels having separate languages and the requirement of structural correspondence exclude certain architectural options, but leave open such design decisions as the synchronisation of model and object system via a causal connection, the switching paradigm (what triggers a shift in levels) and the scope of the self-representation. Investigation of guidelines for navigating through this design space is a current task in REFLECT.

Conclusions

In this article we introduced the concept of *knowledge level reflection* (KLR). In brief, a KLR system has been defined as a system that *realises forms of reflection on object-level problem solving tasks by employing an abstract (knowledge-level) model as its self-representation.*

It focusses around the notion that there are tasks that require a abstract model of problem solving carried out in the object system, thus reason about that problem solving in the same manner as object tasks reason about an external domain.

It differs from other approaches dealing with reflection essentially with respect to the following three points:

- KLR is a knowledge-level approach in the tradi tion of Newell's knowledge-level hypothesis and of the KADS framework. It focusses on using an

epistemologically adequate language for the conceptual modelling activities, and it preserves the distinguished structures as much as possible throughout the rest of the system development process.

- It is a structured and multi-level approach, which allows to give to both the conceptual model and the running system a modular struc ture that mirrors explicitly the different compo nents, roles and behaviours that constitute the building blocks of KLR systems.
- It uses a special type of self-representation which focusses only on those aspects of the ob ject system which are necessary for achieving specified reflective behaviours. Thus, the self-description of a KLR system is (i) distinct from the object part of the system; (ii) partial and tail ored to the reflective task; (iii) abstract. This is realised by employing a knowledge-level model as the system's self-representation.

The distinctiveness of the whole approach is due to the *combination* of the three points mentioned above. Taken individually, none of these three points is completely new, with the exception perhaps of the third point. The first point is becoming widely accepted as an important prerequisite for the development and maintenance of more ambitious and re-usable models and their exploitation by advanced knowledge-based systems. The second point has a long tradition in all disciplines which build artifacts of a critical size and complexity. The novelty of our approach lies in applying these ideas concerning knowledge-level structuring to reflection. The third point is a natural consequence of this line of thinking.

Another very important characteristic of our approach is the explicit commitment to the preservation of the structures used for conceptual modelling both for the formalisation and for the operationalisation of KLR-systems. It is this structural *correspondence principle* that will allow us to solve the software construction, explainability and maintenance problems associated with the development of powerfull reflective systems.

References

[Akkermans *et al.*, 1990] J.M. Akkermans, F. van Harmelen, A.Th. Schreiber, and B.J. Wielinga. A formalisation of knowledge-level models for knowledge acquisition. Technical report, Netherlands Energy Research Foundation ECN, Petten (NH), The Netherlands, 1990. To appear in the International Journal of Intelligent Systems.

[Bartsch-Spörl *et al.*, 1990] B. Bartsch-Spörl, M. Reinders, H. Akkermans, B. Bredeweg, T. Christaller, U. Drouven, F. van Harmelen, W. Karbach, G. Schreiber, A. Voß, and B. Wielinga. A tentative framework for knowledge-level reflection. ESPRIT Basic Research Action P3178 REFLECT, Deliverable IR.2 RFL/BSR-ECN/I.3/1, BSR Consulting and Netherlands Energy Research Foundation ECN, August 1990.

[Campione *et al.*, 1982] J.C. Campione, A.L. Brown, and R.A. Ferrara. Mental retardation and intelligence. In Sternberg [Sternberg, 1982a].

[Chandrasekaran, 1987] B. Chandrasekaran. Towards a functional architecture for intelligence based on generic information processing tasks. In *Proceedings of the 10th IJCAI*, pages 1183--1192, Milano, 1987.

[Clancey, 1985] W.J. Clancey. Representing control knowledge as abstract tasks and metarules. In M.J. Coombs and L. Bloc, editors, *Computer Expert Systems*. Springer, Berlin, 1985.

[Eshelman, 1989] L. Eshelman. MOLE: A knowledge-acquisition tool for cover-and-differentiate systems. In S. Marcus, editor, *Automating Knowledge Acquisition for Expert Systems*, pages 37--80. Kluwer Academic Publishers, Boston, MA, 1989.

[Feferman, 1962] S. Feferman. Transfinite recursive progressions of axiomatic theories. *Journal of Symbolic Logic*, 27:259--316, 1962.

[Giunchiglia & Smaill, 1989] F. Giunchiglia and A. Smaill. Reflection in constructive and non-constructive automated reasoning. In H. Abramson and M.H. Rogers, editors, *Meta-Programming in Logic Programming (Proceedings META-88 Workshop, Bristol)*, pages 123--140, Cambridge, MA, 1989. MIT Press.

[Gödel, 1931] K. Gödel. Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I. *Monatsh. Math. Phys.*, 38:173--98, 1931. English translation in {*From Frege to Gödel: a source book in Mathematical Logic*, 1879-1931, J. van Heijenoort (ed.), Harvard University Press, 1967, Cambridge, Mass.

[Gruber, 1989] T.R. Gruber. *The Acquisition of Strategic Knowledge*. Academic Press, 1989.

[Hayward et al., 1987] S.A. Hayward, B.J. Wielinga, and J.A. Breuker. Structured analysis of knowledge. *International Journal of Man-Machine Studies*, 26:487--498, 1987.

[Karbach et al., 1989] W. Karbach, M. Linster, and A. Voß. Office-plan: Tackling the synthesis frontier. In D. Metzing, editor, *GWAI-89, 13th German Workshop on Artificial Intelligence*, pages 379--387, Berlin, 1989. Informatik Fachberichte 216, Springer-Verlag.

[Karbach *et al.*, 1990] W. Karbach, A. Voß, U. Drouven, and R. Schuckey. Model-k: Prototyping at the knowledge level. In *Proceedings of the 11th International Workshop on Expert Systems \& their Apllications, Avignon 91*, 1990.

[Kripke, 1975] S. Kripke. Outline of a theory of truth. *Journal of Philosophy*, 13(72), 1975.

[Laird et al., 1987] J.E. Laird, A. Newell, and P.S. Rosenbloom. SOAR: an architecture for general intelligence. *Artificial Intelligence*, 33:1--64, 1987.

[McDermott, 1989] J. McDermott. Preliminary steps towards a taxonomy of problem-solving methods. In S. Marcus, editor, *Automating Knowledge Acquisition for Expert Systems*, pages 225--255. Kluwer Academic Publishers, The Netherlands, 1989.

[Maes, 1987] P. Maes. Computational reflection. Technical Report 87-2, AI-Laboratory, Vrije Universiteit Brussel, 1987.

[Musen, 1989] M.A. Musen. *Automated Generation of Model-Based Knowledge-Acquisition Tools*. Pitman, London, 1989. Research Notes in Artificial Intelligence.

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

[Perlis, 1985] D. Perlis. Languages with self-reference {I}: Foundations. *Artificial Intelligence*, 25:301--322, 1985.

[Pylyshyn, 1978] Z.W. Pylyshyn. When is attribution of beliefs justified? *Behavioral and Brain Sciences*, 1:592--593, 1978.

[Rosenbloom et al., 1988] P. Rosenbloom, J. Laird, and A. Newell. Meta-levels in soar. In P. Maes and D. Nardi, editors, *Meta-Level Architectures and Reflection*, pages 227--240. North-Holland, Amsterdam, 1988.

[Schreiber *et al.*, 1990] A.Th Schreiber, J.M. Akkermans, and B.J. Wielinga. On problems with the knowledge level perspective. Technical report, University of Amsterdam/Netherlands Energy Research Foundation ECN, Amsterdam/Petten, 1990. Banff-90 International Knowledge Acquisition Workshop, to appear.

[Smith, 1986] B.G. Smith. Varieties of self-reference. In J.Y. Halpern, editor, *Theoretical Aspects of Reasoning about Knowledge*, Proceedings of the 1986 Workshop at Monterey, CA, pages 19--43, Los Altos, CA, 1986. Morgan Kaufmann. .

[Sternberg, 1982a] J.S. Sternberg, editor. *Handbook of Human Intelligence*. Cambridge University Press, Cambridge, 1982.

[Sternberg, 1982b] J.S. Sternberg. Reasoning, problem solving and intelligence. In *Handbook of Human Intelligence* [Sternberg 1982a}, pages 225--307.

[Tarski, 1936] A. Tarski. Der Wahrheitsbegriff in den formalisierten Sprachen. *Studia Philosophica*, 1:261--405, 1936. English translation in *Logic, Semantics, Metamathematics*, A. Tarski, Oxford University Press, 1956.

[Turing, 1939] A.M Turing. Systems of logic based on ordinals. *Proceedings of the London Mathematical Society*, 45:161--228, 1939.

[vanHarmelen, 1989] F. van Harmelen. A classification of meta-level architectures. In H. Reichgelt P. Jackson and F. van Harmelen, editors, *Logic-Based Knowledge Representation*, chapter 2. MIT Press, Cambridge, MA, 1989.

[Voß *et al.*, 1990] A. Voß, W. Karbach, U. Drouven, and D. Lorek. Competence assessment in configuration tasks. In *Proceedings ECAI'90*, pages 676--681, London, 1990. Pitman.

[Weyhrauch, 1980] R. Weyhrauch. Prolegomena to a theory of mechanized formal reasoning. *Artificial Intelligence*, 13:133--170, 1980. Also in: *Readings in Artificial Intelligence*, Webber, B.L. and Nilsson, N.J. (eds.), Tioga publishing, Palo Alto, CA, 1981, pp. 173-191. Also in: *Readings in Knowledge Representation*, Brachman, R.J. and Levesque, H.J. (eds.), Morgan Kaufman, California, 1985, pp. 309-328.

[Wielinga *et al.*, 1991] B.J. Wielinga, A.Th. Schreiber, and J.A. Breuker. KADS: A modelling approach to knowledge engineering. ESPRIT Project P5248 KADS-II/T1.1/PP/UvA/008/1.0, University of Amsterdam, 1991. Submitted to *Knowledge Acquisition*.