

# Integrating multiple problem solvers in knowledge-based systems

E. CHRISTODOULOU and E. T. KERAVALNOU

*Department of Computer Science, University of Cyprus, P.O. Box 537, CY-1678 Nicosia, Cyprus, (email: {cseleni,elpida}@turing.cs.ucy.ac.cy)*

## Abstract

Research in knowledge-based systems has shown that the use of multiple knowledge representation formalisms and reasoning mechanisms for achieving a specific task in complex domains could result in efficient and effective problem solving. This has led to the development of a number of general architectures and application-specific systems integrating multiple problem solvers. A problem solver is defined to be an association between a knowledge intensive (sub)task, an inference mechanism and a knowledge representation formalism on which the inference mechanism is working to achieve the (sub)task. A knowledge-based system making use of different problem solvers should address a number of critical aspects of integration of the solvers, like interaction, invocation, reactivity, learning and expandability. The aim of this paper is to distinguish and discuss essential integration aspects, and to review a number of proposed general hybrid architectures and application specific hybrid systems on the basis of these aspects. The review shows that none of these general architectures or application specific systems directly addresses all the identified integration aspects. In general, the limitations exhibited by these systems are due to the naive form of interaction and invocation of the integrated solvers. We give a high level specification for a competent hybrid knowledge-based architecture that supports the identified integration aspects.

## 1 Introduction

Decomposition in problem solving is a basic weapon against complexity. Task decomposition in human problem solving often provides a great deal of parallelism, which is conceptually and operationally important for increased efficiency. No one disputes the significance of task decomposition for any type of problem solving situation (Chandrasekaran, 1982, 1983, 1988; Chandrasekaran & Johnson, 1993; Keravnou & Johnson, 1986; Keravnou & Washbrook, 1989; Mittal et al., 1979; Punch, 1989; Punch & Chandasekaran, 1993). In algorithmic problem solving the principle of task decomposition guarantees a modular solution. However, in algorithmic problem solving the achievement of every (sub)task is rigidly performed in a deterministic way based on the functions/procedures, and their interactions, implementing it. This is not to say that choices are not incorporated; only that every choice point is unambiguously and categorically decided and no backtracking is necessary. Hence it can be said that in algorithmic problem solving every (sub)task embodies a single, well-defined reasoning mechanism for achieving it which applies to all situations where the services of the particular (sub)task are invoked.

In heuristic, knowledge-intensive problem solving, multiplicity of reasoning mechanisms is as essential as task decomposition if the behaviour of the knowledge-based system is to be characterized as competent, flexible and comparable to that of the corresponding human problem solvers. Determinism has never been a characteristic of knowledge-based systems, due to the fact that knowledge for most realistic domains is incomplete and uncertain, and similarly, the data of

relevant problem cases is incomplete and uncertain. Choice points encountered by a knowledge-based problem solver during its processing are rarely resolved in an unambiguous and categorical fashion. Instead the seemingly “best” choice at the time is made, but an alternative choice (for the same decision) is possible at a later point in time.

Hence, for effective knowledge-based problem solving, a (sub)task of the overall reasoning activity is associated with multiple, alternative solving approaches for achieving it. This collection of solving approaches constitutes a choice point in the context of achieving the relevant (sub)task. Such choice points are in part unknown, and similarly, the selection criteria for resolving them are rarely accurate and complete. Problem solving experience, if effectively channelled, could result in refining and extending the solving approaches for some (sub)task, thus resulting in mechanizing the important type of learning, *strategic learning*.

The solving approaches for some (sub)task are essentially competitors, but they can be used in a collaborative way as well; one solving approach takes over from where another solving approach has left. To signify the collaborative aspect (over the competitive aspect) we do not talk about a failed instantiation of some approach, but instead we talk about a suspended (or interrupted) instantiation which may well resume execution if the reason for its suspension no longer holds, because another solving approach managed to tackle the particular issue. We stress the difference between solving approaches that can both compete and collaborate, and solving approaches that can only collaborate. The reason we stress the difference is that true choice only exists under the former set up and not under the latter. If the collaborators involved in a knowledge-based system are deterministically determined then no particular intelligence is required in dynamically integrating the components of such a system for achieving some goal. True intelligence is only required when the collaborators in some venue need to be dynamically selected and integrated. Dynamic selection implies that there are alternative solving approaches for achieving some aspect of the overall goal. Architectures for knowledge-based systems based on collaborative task decomposition, and more specifically the assembling of such architectures by statically selecting a single appropriate solving approach for some aspect of the problem out of a library of alternative approaches, have been the topic of much discussion in the past, and now constitute a reasonably well understood technology. Meta-control units employed by such architectures, if any, have a different and simpler role to the one played by a metareasoner in a knowledge-based architecture employing solving approaches that can be used both competitively and collaboratively. Such architectures are not well understood, and a strong motivation for surveying this area is to clarify the particular state of the art. For lack of a better name we use the term *hybrid* architectures (used by many other researchers) to define this class of “competitive-collaborative” architectures of knowledge-based systems. However, we appreciate that many systems described as hybrid do not fall into our definition, and *vice versa*.

The move towards using multiple solving approaches for a knowledge-intensive task, like medical diagnosis, is supported by the findings of cognitive scientists analysing the decision making behaviour of human domain experts. The main characteristic of a human expert is his ability to perform tasks at an exceptional level of performance, in a specialized domain. Research with experts from a wide range of domains has replicated the finding that the superiority of an expert’s performance in problem solving is due to his appropriate decomposition of a task into subtasks, the ways his knowledge is organized, and the number of reasoning mechanisms that he flexibly applies in the context of every subtask, for achieving efficient solutions (Chi et al., 1982; Patel & Groen, 1991). The knowledge is organized in different structures of varying degrees of specificity. For example, a human expert can easily abstract the most important characteristics of his domain, can reason with finely detailed and functional descriptions of the domain entities, and can also reason at the level of actual, past, problem-cases. Human domain experts are highly skilled at quickly identifying which structures of their knowledge to use for a specific problem case, and also at changing their reasoning when the currently pursued line of reasoning ceases to be the best, thus going out of an unpromising route. Associated with every knowledge structure are mechanisms for manipulating this knowledge. A knowledge structure with a reasoning mechanism can be viewed as

a problem solving unit representing a specific way for achieving either a complete solution, or part of a solution, to some problem. Thus, there are different ways for solving a problem, each representing a different view of the relevant knowledge. The idea is to have available a number of different problem solving units which can act more or less independently. An important aspect of the human memory is the existence of a central memory executive (Logie, 1993). One function attributed to the central executive system is the coordination of the activities of the problem solving units. It is unlikely that such coordination can be provided by the problem solving units themselves. The central memory executive acts as a general attentional resource, coordinating the operation of the different problem solving units. It is involved in reasoning, decision making, comprehension and long term learning. It is considered to be more complex and less modality specific than the individual problem solving units. Freud's theories of human memory (Litcher, 1992) were based on the view of the mind as an interacting society of many solving units and a top-level control. When top-level control in a society of mind solving units is weak or nonexistent, then these units may speak up in different, possibly conflicting ways. Examples of the nonexistence of a central executive are Alzheimer patients (Baddeley et al., 1991).

The premise of this survey paper is that only hybrid knowledge-based systems, as defined above, have multiple problem solvers<sup>1</sup>, where a problem solver is defined to be an association between a knowledge intensive (sub)task, an inference mechanism and a knowledge representation formalism on which the reasoning mechanism is working to achieve the (sub)task. The multiplicity of solving approaches exists not only for the primitive tasks, but also for the higher level tasks. In fact multiplicity of problem solving approaches with respect to the topmost task is a necessary requirement. It is these solving approaches that are managed by the global metareasoner, while multiplicity of solving approaches at a sublevel is managed locally, quite possibly in the same way, through a local metareasoner.

The purpose of this paper is to distinguish and discuss a number of critical aspects of integration of the different problem solvers in hybrid knowledge-based systems (section 2), to review a representative sample of general hybrid architectures and application specific hybrid systems on the basis of these integration aspects (section 3), and to give a high level specification for a hybrid knowledge-based architecture that aims to address all the integration aspects (section 4).

## 2 Aspects of integration

Hybrid knowledge-based systems need to adequately address critical aspects of integration of the different problem solvers. The ultimate objective is to integrate different problem solvers in such a way that the multiplicity of problem solvers is not apparent to the user of the system.

In this section we list and discuss six aspects of integration which we consider essential, and therefore propose that they should be adequately addressed in any hybrid knowledge-based system. These aspects, which we use as our evaluation criteria for the review of the selected hybrid architectures and systems presented in the following section, are:

- The interaction aspect, i.e., how the different problem solvers communicate and are controlled.
- The invocation aspect, i.e., which problem solvers should be used in an initial solving plan for a specific problem case.

<sup>1</sup>We distinguish between a system making use of multiple problem solvers and one making use of multiple agents. In a multi-agent system every agent is associated with a *different* task to be solved and it communicates with other agents for achieving the goals of the system as a whole. Different agents are relatively autonomous in deciding what actions to perform for achieving their associated tasks. It is for example possible that an agent responsible for solving a knowledge-intensive task uses a number of different solving approaches (different problem solvers) for achieving this specific task. In a system making use of multiple problem solvers, more than one problem solver is associated with the same (sub)task and each one of them represents a different way of achieving the same goal. These problem solvers can be used both competitively and collaboratively in the context of the given (sub)task.

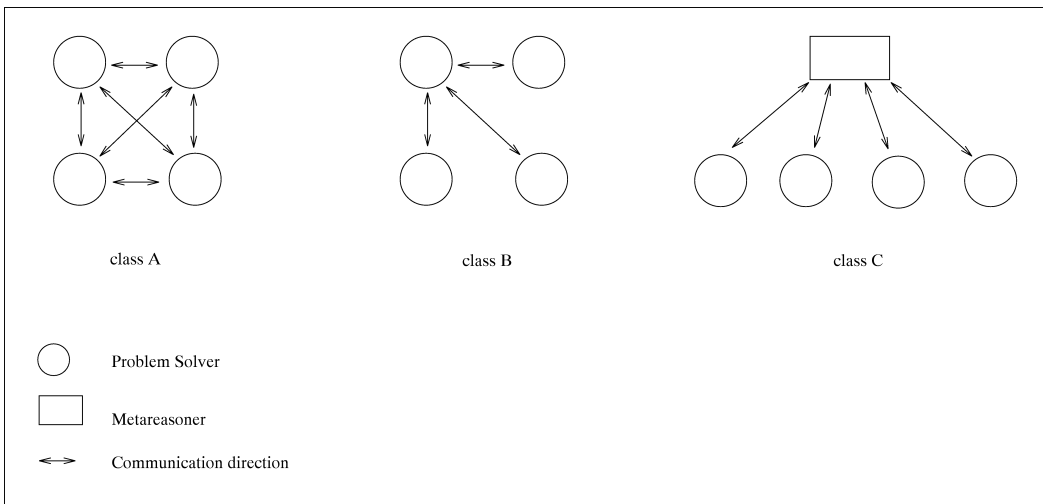


**Figure 1** Different interaction mechanisms

- The reactivity aspect, i.e., what should be done if the initial plan for solving a problem case proves to be inappropriate for the specific case.
- The types of the problem solvers involved. The type of a problem solver reflects its inference mechanism and level of detail of its domain view.
- The expandability aspect (addition or deletion of problem solvers).
- The learning aspect, i.e., the ability of a hybrid knowledge-based system to evolve on the basis of its experiences in problem solving.

### 2.1 The interaction aspect

The use of multiple problem solvers obviously leads to an interaction problem: How the different problem solvers communicate and are controlled in order to solve a specific problem. Three different interaction mechanisms can be used which are illustrated in Figure 1. Under the first mechanism, each problem solver communicates directly with the others (Figure 1 class A). This increases the complexity and influences negatively the expandability aspect of a system. Class A interaction is described as either tightly or loosely coupled (Hunt & Miles, 1994). Tightly coupled interaction means that the lines of communication and focus of control are statically determined, and because of their static nature they can be described algorithmically. Loosely coupled interaction means that the various elements of the system are not closely tied together and, as a consequence, control may be determined dynamically. Under the second interaction mechanism one problem solver is dynamically decided to be responsible for coordinating all the others (Figure 1 class B). This means that the coordinating solver should control the whole solving process, and should perhaps interrupt its own problem solving to support other problem solvers in their decision making. The performance of this solver may therefore be negatively affected, because of its additional responsibilities. Each solver communicates directly only with the coordinating solver. The fact that all the solvers are at the same level creates difficulties in assigning meta-level activities to the chosen coordinating solver, like defining and learning problem solving strategies.

A more flexible way of controlling the use of, and the communication between, the different problem solvers is by permanently utilizing a meta-level control unit, a metareasoner (Figure 1 class C). This means that all the problem solvers *per se* are defined at the same level while the metareasoner is at higher level, globally overseeing the operation of the problem solvers. Each solver communicates directly with the metareasoner and there are no intercommunications between the solvers. The metareasoner has meta-level knowledge of the different problem solvers, which includes descriptions of the different problem solvers, relations between problem solvers (for

example, which of them are both collaborators and competitors) and any constraints governing such relations. It is responsible for choosing the most appropriate combinations of problem solvers for solving specific problem cases, for activating the relevant solvers, for keeping control of the behaviour of the overall problem solving process (for example detecting in a timely fashion the inappropriateness of a selected solver and deactivating it), and also for providing constraint mechanisms for checking the dependencies between the different solvers. In general, the metareasoner is responsible for the overall control and communication of the different solvers.

*Interaction aspect: Specifies how the different problem solvers communicate and are controlled.*

## 2.2 The invocation aspect

The invocation aspect deals with the question of how to select among the different problem solvers the best combination to start with for a specific problem. So far only very simple forms of invocation have been exploited. One approach requires each solver to specify criteria for determining the problem cases, for which it can, and cannot be used. This approach is probably infeasible except in trivial situations (DeJongh & Smith, 1989). It can be improved by using data-driven heuristics to rate the plausibility of each problem solver at the outset of the problem solving process. For example in a diagnostic context, the data may be characteristic of the presence of a single point of failure, suggesting the use of a problem solver that makes a single point of failure assumption. The existence of such heuristics depends upon both the domain and the data collection process.

Another approach involves hierarchical invocation, proceeding from the solver using the most abstract view of the domain knowledge down through solvers using domain models of increasing levels of detail. This means that the solver with the most abstract view of domain knowledge is initially selected for each problem case. Different operators are then applied by the current solver, unless an expectation failure occurs, in order to select a solver using a more detailed view of the domain knowledge. Such a hierarchical approach reduces the flexibility of a system as the possible problem solving paths are predefined.

A more competent approach is to choose appropriate solvers dynamically depending on global problem solving parameters and the characteristics of the specific problem case. Problem solving parameters could include the desired level of detail of the computed solutions, the maximum length of time for solving the problem and the desired level of security of the solutions. This implies that before a specific problem solver should be activated, a global evaluation of the overall problem solving situation should be carried out. Based on this evaluation, a problem solving strategy should be planned, giving the problem solvers to be used and their relative activation patterns. For example, if the time aspect is given as a global parameter for a specific problem case, then a problem solving strategy should aim to utilize those problem solvers that could come up with some solution faster than the others. Normally, a combination of problem solvers would be used for achieving the desired results. The important thing is that both successful and unsuccessful strategies should be learned, so that successful strategies are repeated and past failures are avoided, rather than devising new strategies all the time. The meta-task of planning and learning problem solving strategies is a complicated one and it fits best under the realms of a metalevel control unit. Such a unit should be responsible for providing mechanisms for evaluating the overall problem solving situation regarding a specific case and also for planning, applying and learning problem solving strategies. Research in using case-based methods (Kolodner, 1993) for capturing strategic knowledge is very relevant to the strategic invocation of problem solvers (Aamodt, 1993, 1994a,b, 1995; Aamodt & Nygaard, 1995).

*Invocation aspect: Specifies the dynamic planning of the most appropriate initial problem solving strategy for a specific problem case; this should be incrementally enhanced through the accumulation of expertise.*

### 2.3 The reactivity aspect

The choice of a problem solving strategy, based on a first analysis of a problem case, could prove inefficient or ineffective during the actual problem solving process. The reactivity aspect ensures that the execution of a problem solving strategy should be dynamically monitored and reviewed throughout so that necessary adaptations, or replacement of strategy, are done in a timely and constructive way to minimize wastage. In general, a problem solving system can be described as reactive if it is able to continue the processing in situations where a chosen strategy proves to be inappropriate, or in situations where the expected solution is not achieved. Most of the existing hybrid systems follow a predefined invocation sequence of the different problem solvers and no fallback procedures are available for situations where the predefined strategy fails, thus lacking reactivity altogether. A truly reactive system should be able to adjust the initial plan for solving a case according to new events occurring dynamically during the solving process. For this purpose, dynamically detected events should be interpreted, to see if new actions should be undertaken. New actions could, for example, refer to the activation of other problem solvers, or the checking of constraints regarding the possibility of the existence of complex solutions.

*Reactivity aspect: Specifies how to reactively adjust the initial plan for solving a problem case in order to take into consideration new events occurring dynamically during the problem solving process.*

### 2.4 Types of problem solvers involved

A task can have different types of solution, like detailed, approximate, based on experience, etc. For each solution type different problem solvers can be defined. For example, an approximate solution can be achieved by making use of two problem solvers; one applying a rule-based reasoning mechanism on an abstract domain view, and another applying a case-based reasoning approach on a number of already solved cases. Similarly, a detailed solution can be achieved again by making use of a rule-based reasoning mechanism, but which operates on a detailed view of the domain knowledge. Thus, in the example given above there are three types of problem solvers (abstract rule-based, abstract case-based, detailed rule-based) where the type of a problem solver reflects its inference mechanism and level of detail of its domain view. A rich hybrid architecture should support different types of problem solvers from the perspective of a meta-level classification of possible solutions.

*Types of problem solvers involved: Specifies whether a meta-level classification of problem solvers for the same (sub)task is supported, reflecting a meta-level classification of possible solutions.*

### 2.5 The expandability aspect

System expandability refers to the ability of a knowledge-based system to facilitate subsequent additions and/or deletions of problem solvers by a knowledge engineer or domain expert. This aspect is essential in a hybrid system that provides different problem solving approaches for capturing the required multiplicity in solution types for a specific task. The types of solutions for some task could be changed dynamically, depending on present and future requirements for the task. Such changes entail the inclusion or exclusion of problem solvers. This should be done without influencing the remaining problem solvers. A good way of achieving this is by using a meta-control unit that maintains knowledge on the different solvers and the dependencies between them. The functions used by the global meta-control unit for manipulating the problem solvers should be independent of the number and the types of the solvers involved at any time.

*Expandability aspect: Determines whether subsequent additions and/or deletions of problem solvers by a knowledge engineer, in order to adapt to new demands regarding present and future requirements for a specific task, are facilitated.*

## 2.6 The learning aspect

Learning refers to the ability of a system to evolve on the basis of its experiences in problem solving. It constitutes a critical aspect of a hybrid knowledge-based architecture. Learning should occur at different levels, learning at a strategic level or learning at the level of the specific problem solvers. The latter refers to the ability of the solvers to improve their reasoning mechanisms and to enhance their knowledge to increase their time efficiency and the correctness of their solutions. Different solvers may use different learning mechanisms depending on the knowledge representation formalisms and the inference mechanisms that they implement. Each solver could evolve on the basis of its accumulated experiences independently of the other solvers.

Integrating different problem solvers implies that strategies should be used for invoking appropriate combinations of solvers for a specific problem case and chosen strategies should be dynamically adapted depending on new events occurring dynamically during problem solving. Learning at strategic level aims to improve problem solving efficiency by enhancing the possibility of choosing from the beginning the right problem solving strategy, i.e., to enhance the systems's ability in planning the initial problem solving strategies, and also in enhancing the system's reactive adaptation of chosen strategies. For this purpose both successful and unsuccessful solving strategies as well as adaptations of strategies should be memorized for future use.

In general, knowledge level learning aims to enhance the soundness, completeness and overall quality of the domain views (knowledge bases) of the different problem solvers, and hence it is performed at a local level. In contrast, strategic learning needs to be performed at a global level. Thus, because of the globality of strategic knowledge, strategic learning is again better placed in the realms of a meta-controller capable of providing different learning mechanisms. Recent research in problem solving learning using meta-knowledge and multiple learning mechanisms is very relevant to the learning aspect of hybrid knowledge-based systems (Cox & Ram, 1992; Plaza et al., 1993; Plaza & Arcos, 1993; Ram et al., 1992; Ram & Cox, 1994).

*Learning aspect: Specifies how to improve the problem solving performance on the basis of accumulated experience both at strategic and domain knowledge levels.*

## 3 General hybrid architectures and application specific systems

Different research groups are presently working towards the development of knowledge-based systems exploiting multiple problem solvers. Some of these efforts concentrate on the definition of general hybrid architectures, but most of the effort is in developing application specific hybrid systems. Most of the hybrid architectures make use of the term method instead of problem solver. A method defines a reasoning mechanism which is applicable across domains (it is domain independent). A specific task can be achieved by making use of different methods. For example the diagnostic task can be achieved by making use of the abductive assembly method or the heuristic classification method. A problem solver in our definition represents an association between a knowledge intensive (sub)task, an inference mechanism and a knowledge representation formalism modelling the domain from a specific view. For the purpose of this paper every problem solver adopts a generic problem solving method (abductive assembly, heuristic classification, etc.).

In addition most existing hybrid architectures are reflective systems. In recent artificial intelligence research, the term "reflective system" is used to identify a class of systems which in some way or other have knowledge about themselves and which are able to reason about and act upon themselves. So far, much theoretical work has been carried out regarding the definition of reflective systems (Maes & Nardi, 1988; van Harmelen, 1992). In this paper we do not deal with the theoretical work done in this area, but we do refer to architectures defining multiple methods or problem solvers that use notions of reflective systems.

In the following two subsections a representative sample of general, hybrid knowledge-based architectures and application specific hybrid systems are reviewed on the basis of the integration aspects discussed in section 2. Regarding the architectures, the order of presentation is to a large

extent random, since we have deliberately selected architectures which are fairly distinct in some sense from each other. However the application specific systems are grouped into those that are of a class A tightly coupled approach, those that are of a class A loosely coupled approach and those that are of a class C approach. Within each group the order of presentation is again fairly random.

### 3.1 General hybrid knowledge-based architectures

#### 3.1.1 Ten Teije and van Harmelen architecture

Ten Teije and van Harmelen (1995) propose a meta-architecture that supports a dynamic selection and configuration of problem solving methods for a particular problem type, depending on the actual problem and the problem solving goal. They argue that for a particular problem type a number of problem solving methods can be employed to solve an actual problem of the given type. For example, diagnosis problems can be solved by such diverse methods as consistency-based diagnosis, hierarchical diagnosis, abduction, etc. (Console et al., 1992).

**Description** The development of this architecture is based on the idea that there is not a single optimal problem solving method, and the choice of an appropriate problem solving method depends on the goal of problem solving and the characteristics of the specific problem case. Three types of knowledge are distinguished for performing flexible problem solving: knowledge that describes the applications (application description), knowledge that describes the problem solving methods (methods description) and knowledge that describes how to use these methods for flexible problem solving. Examples of the third type of knowledge are: under which conditions particular methods are selected, relations between different methods, and knowledge of how to use these relations between methods for configuring and approximating problem solving methods. These three different knowledge types are represented explicitly and separately. The architecture is defined to fulfill the following requirements. Firstly, a knowledge type must be able to express relations among objects of another knowledge type, secondly a knowledge type must be able to abstract from the instances of another knowledge type, and thirdly, a knowledge type must be able to express the roles that are played by the objects of another knowledge type. The content of the different knowledge types and their relations are expressed in first order logic. The meta-architecture discussed in ten Teije and van Harmelen (1995) has interesting aspects regarding the definition of three theories (object-theory, meta-theory and meta-meta-theory) each of which depends on one of the knowledge types mentioned above. The connection between the object-theory and the meta-theory is realized by reflection rules and naming relations. A reflection rule is a rule that performs a transformation of a formula that holds in the object-theory to a formula that holds in the meta-theory, or *vice versa*. A naming relation associates elements of the object-theory with their names in the meta-theory, thereby allowing the meta-theory to refer to, and express properties of the elements of the object layer (van Harmelen, 1992).

**Analysis** This architecture could be classified as a hybrid architecture because of the definition of methods (at the second knowledge level) providing different ways of achieving solutions to a problem type. A problem solving method is defined as a scheme consisting of components, and instantiations of this scheme correspond to specific methods. One of the components of a method contains a reflection to the application knowledge level. Thus an instantiation of a method's scheme together with its reflection to the domain knowledge could be seen as a problem solver in our definition. Different instantiations provide a multiplicity of differently typed problem solvers. In general, the work presented in ten Teije and van Harmelen (1995) provides interesting issues on the identification and representation of different knowledge levels and their relations; especially the identification of a meta-meta knowledge layer responsible for controlling the use of the methods defined at the meta-level. Because of the identification of a meta-meta knowledge layer the proposed architecture can be classified as a class C interaction approach (see Figure 1). The description of the proposed architecture, however, is very theoretical and does not provide concrete answers to the interaction, invocation, reactivity and learning aspects.

### 3.1.2 DeJongh's problem solving framework

DeJongh (DeJongh & Smith, 1989, 1991; DeJongh, 1991) describes a perspective for the analysis of knowledge-based systems which use representations of domain knowledge at multiple levels of detail.

**Description** Problem solving in this work refers to the identification of (sub)goals to be achieved and the identification of bodies of knowledge which can be applied to achieve them. Using the description of generic tasks (Chandrasekaran, 1986; Chandrasekaran & Johnson, 1993) the identified (sub)goals are associated with reasoning mechanisms for achieving them. A framework for integrating the use of multiple models of a domain is defined and applied in the domain of blood typing. A model consists of a set of knowledge objects expressed at a specific degree of detail. It is associated with a reasoning mechanism and a domain (sub)goal. Thus, a model could be seen as a problem solver in our definition.

**Analysis** The main aspect of this work is that different knowledge models of various degrees of detail are defined and a hierarchical invocation of these models is made, starting from the most abstract model. The different models are invoked by detecting assumption violations that occur when a model is inadequate for the case in hand. When an expectation failure occurs, the assumption violation responsible for the expectation failure is analysed so that a more precise model can be selected. The invocation of the models is predefined, so that results of more abstract models can be used by more detailed models. The different models are loosely coupled according to the class A interaction approach (see Figure 1). The hierarchical involvement of different models in the framework depends on domain and generic goals that should be achieved. This means that theoretically the expandability aspect is facilitated. The framework uses a hierarchical invocation of the different models, does not provide any planning of problem solving strategies, and does not exhibit any reactive behaviour. The different models in some sequence of invocation are just collaborators since the detailed models are used to support the more abstract models, for solving a specific problem case. Normally, only one model is associated with a specific level of detail and thus there is no competition of different models. Finally, no learning is supported by the proposed framework.

### 3.1.3 Van Heijst's architecture

Van Heijst et al. (1994) argue that solving real world problems often involves different types of knowledge and reasoning techniques. A combination of a knowledge representation formalism and a reasoning technique represents a problem solver in our definition. Thus, multiplicity of problem solvers is necessary for solving real world problems. They emphasize that the main issue in integrating different problem solvers is how their knowledge representation formalisms can be mapped into one another and they propose an approach to the integration of different knowledge representation formalisms which is based on the contents of the knowledge.

**Description** An architecture is proposed that mainly addresses the issue of how different knowledge representation formalisms can be integrated, or more specifically: how expressions in one formalism map into expressions in another formalism. The architecture presents a so called *knowledge based integration*, in which different formalisms are integrated based on the content of the knowledge and not on the syntactical form of the knowledge. A knowledge based integration implies the existence of a formal knowledge model that has unlimited expressiveness. The main purpose of this model is to make the underlying structure of the domain knowledge explicit. Every problem solver maps the concepts in the formal knowledge model into its own knowledge representation formalism.

**Analysis** Even though the proposed architecture stresses the necessity of using multiple problem solvers, it does not address the important aspects of control, invocation, reactivity and learning. The designers argue that the problem of integrating different representation formalisms can be resolved independently of all the other issues.

### 3.1.4 *Voß's architecture*

In Voß et al. (1992), it is argued that a knowledge-based system is enhanced by a suitable configuration of competence specialists.

**Description** The competence specialists included in the architecture are mutually independent, generic modules, each realizing a particular behaviour. The specialists can run on a stand-alone basis and are invoked according to specific requirements like time to be spent and number of solutions. The proposed architecture distinguishes a meta-system and an object system. Competence specialists reason at the meta-level about the (in)competence of the object system in terms like "input problem", "potential solutions", "goals", "alternative ways to achieve a goal", etc. Each specialist contains a, maybe partial, domain knowledge view of the object system. A specialist together with its domain knowledge view of the object system could be regarded as a problem solver in our definition. One of the specialists (case-based reasoner) uses a library of successful combinations of specialists to shortcut the entire problem solving process and modifies the case library by adding new successful solving cases. An operational language, called MODEL-K, is developed to specify and operationalize the proposed architecture (Karch et al., 1991).

**Analysis** The learning aspect in this architecture refers to the saving of successful combinations of specialists. This type of learning can be classified as a simple form of strategic learning. Altogether, 13 specialists are developed and tested in four different systems (Bartsch-Spörl et al., 1991). The four different systems represent reflective systems distinguishing between a meta-level and an object system (van Harmelen et al., 1992). The specialists are loosely coupled following the class A scheme (see Figure 1). As each specialist shows a different behaviour, for example decomposing overcomplex problems, limiting the number of solutions desired or simplifying the problem to speed up the solution process, the invocation aspect refers only to their merging for achieving a required output (Schreiber et al., 1991).

### 3.1.5 *The M-KAT architecture*

The Medical Knowledge Acquisition Tool (M-KAT) (Ramoni et al., 1992; Lanzola & Stefanelli, 1992a,b, 1993a,b) is designed to help a user in building and maintaining knowledge-based systems able to perform medical reasoning.

**Description** M-KAT is based on a blackboard architecture. Such an architecture is structured on a centralized blackboard and a collection of external modules interfaced to the blackboard. The blackboard is the place where all the information known to the system is held. The external modules, called problem solvers, monitor the state of the blackboard and collaboratively add new information that is relevant to the solution of a problem. Each problem solver uses a particular knowledge representation formalism as well as a specific reasoning technique. When the triggering conditions of a problem solver are met, the solver may return a suitable value to be stored on the blackboard. A metamodule is used to coordinate the different problem solvers. In a very general scenario different solvers are available for performing the same (sub)task. Users are allowed to specify the problem solver to be used according to some criteria concerning computational complexity, explanation, representation capabilities, etc.

**Analysis** In general, the initial invocation of the problem solvers to be used for a specific problem case depends on conditions specified by the user and not on planning of problem solving strategies at a meta-level. The architecture's interaction aspect between the solvers is simple depending on metarules, manipulated by a metamodule, relating intermediate process results to actions to be undertaken. Actions refer to further invocation of solvers. In general, M-KAT does not address the learning and the reactivity aspects. Theoretically problem solvers can be added and deleted depending on domain requirements, but this is not facilitated in practice.

### 3.1.6 INRECA

INRECA (Althoff et al., 1993, 1994, 1995a,b; Auriol et al., 1995) represents a tool for building hybrid knowledge-based systems by integrating induction and case-based reasoning.

**Description** Four possible levels of integration between inductive and case-based reasoning technologies have been identified. The first level keeps both solving technologies as stand alone solvers and lets the user choose the one he is interested in. In the second level of integration, called co-operative approach, the solvers are kept separate but they collaborate: one solver uses the results of the other to improve or speed up its own results, or both technologies are used simultaneously to reinforce the results, and to extend the classification capabilities of the other. The third level defines a double switch between the inductive and case-based solver. The inductive problem solver is activated first for a new problem case and when an unknown value is encountered the case-based solver is used. This determines the most similar cases to the current situation, chooses the most probable value from this subset of cases and then switches back to the inductive solver. The last level reuses the best characteristics of each technology to build a powerful integrated mechanism that avoids the weaknesses of each separate technology and preserves their advantages (Althoff et al., 1995a). In this last level INRECA keeps the classification quality of case-based reasoning, but it can also gain speed by inductively learned retrieval algorithms.

**Analysis** The main point of INRECA is that it offers several integration possibilities of two different solvers to meet the requirements of a particular application or of a particular category of users. Although the system presents interesting integration aspects, the selection of which integration level to use depends on the user's requirements and not on global problem solving strategies. Only one of these levels could be chosen for a specific application and each level has its own invocation and interaction mechanisms. Even though the fourth level of integration (seamless level) provides powerful integration mechanisms of the inductive and the case-based solver, the two solvers are used more on a collaborative rather than on a competitive way. This means that the inductive solver provides retrieval algorithms for increasing the performance of the case-based solver. The learning aspect is addressed only in one of the integration levels, the fourth, and it refers only to algorithms for improving the case-based reasoning technology. The third and the fourth level of integration show reactivity in the sense that if the activated problem solver fails in some execution point the other solver takes over. The system does not allow for any expandability as only the two predefined solvers can be involved in problem solving.

### 3.1.7 STREAM

STREAM (Reinders & Bredeweg, 1992) represents an experiment in implementing a form of reactive planning of multiple problem solving methods by utilizing a knowledge level model of problem solving and additional knowledge in the form of strategy rules that make explicit the applicability of methods and selection between alternatives.

**Description** Multiplicity of problem solving methods is used. Each method uses different types of models, e.g., causal networks, functional models, structural and behavioural models. Thus, there are multiple methods and models. A model represents a problem solver in our definition. The basic idea of the proposed framework is a separation between an object system and a reflective system. The object system is an implementation of one or more problem solving methods. The reflective system reasons about the object system in order to assess or improve its competence. It uses reactive planning to dynamically select existing problem solving methods and resources by reasoning about an abstract model of problem solving and explicit strategy rules. An abstract model of problem solving is abstracted from implementation details. Reactive planning is the dynamic selection of single actions based on the current problem solving situation and explicit strategic knowledge. After execution of the applicable methods, this is repeated for the new situation. Selection constraints are used to restrict concurrent applicability of competing methods and requirements. Preference rules are also used to conclude a preference between two methods.

**Analysis** In general, this approach separates strategic knowledge from application specific problem solving methods. An example is presented of using two different models for abductive diagnosis: a causal network solver and a qualitative structural and behavioural solver. Through reactive planning and strategic rules, knowledge missing from the causal network is obtained from the behaviour model. Even though STREAM provides an interesting application of reactive planning, no reference is made to the learning aspect.

### 3.1.8 The ANALOG project

Within the ANALOG project (Arcos & Plaza, 1994a,b; Plaza & Arcos, 1993b,c, 1994) a knowledge modelling framework has been developed integrating different inference and learning methods. The ANALOG project hypothesis is that learning methods could be integrated if they are viewed as inference methods able to inspect the problem solving process and modify the system itself so as to improve its behaviour. The approach taken is to start with the reflective representation language NOOS that integrates case-based reasoning and learning and to extend to integrate other machine learning methods.

**Description** The elements that build up the framework are tasks, methods, theories and case models. Tasks are goals to be achieved and problem solving methods are reasoning mechanisms to achieve tasks. Case models and theories embody the domain knowledge for a problem. A case model consists of all factual knowledge on some topic and the tasks that make sense for the specific topic. A theory is defined by the methods declared usable to solve the tasks of a case model. In particular, for every task in the case model the theory holds an object called metafunction containing a set of methods declared usable for that task and a set of preferences to choose among them. For example the metafunction associated with the task of diagnosis may contain the generate-and-test and heuristic-classification methods as possible methods for solving the diagnostic task, plus a preference to try the former before trying the latter method. The framework integrates learning by using the notion of memory and the notion of a new level (so-called inference level) that work upon domain knowledge. Memory is required so as to have successful methods and failures to guide future decision making. At the inference level, inference methods are used to deal with failures of methods on domain tasks and to explicitly search for new domain knowledge and new methods that may overcome the impasses produced by failures. The knowledge modelling framework described above is implemented in the object-oriented frame-based language NOOS (Arcos & Plaza, 1995). NOOS supports the notion of impasse: every time a lack of knowledge to do something is detected, an opportunity for learning is detected and an impasse is generated. NOOS also assures that a method will be tried out to see if it solves a task, backtracking to other declared methods if it fails.

**Analysis** The philosophy of the framework described above is that different methods can be used for achieving solutions to a specific task, and methods can be decomposed into elementary subtasks that use a set of elementary methods (e.g., conditionals, set intersections, etc.). For example a case-based reasoning method can be decomposed into the elementary subtasks of retrieve, select, and reflect and a number of methods can be provided for achieving these subtasks. Once a chosen method can not provide appropriate solutions, an impasse occurs to the inference level for deciding which of the provided methods should be used next. Learning regards the memorization of successful and failed methods and of what impasses have occurred. An advantage of the framework is the definition of the inference level representing a kind of meta-level reasoner allowing preferences among methods, impasse-driven learning and inferencing upon conflicting criteria. Even though the framework described above goes a step further than the other architectures by defining multiple inference and learning methods, the invocation aspect, i.e., which of the methods and in which sequence should be used (at a global level) for a specific case is not strong enough, meaning that problem solving is not strategically planned (preferences provide local rather than global, and hence integral control). This disadvantage could be overcome by improving the meta-level reasoner to carry out strategic planning. The proposed framework supports the

**Table 1** Evaluation of general hybrid architectures (X: not addressed)

| <i>Integration aspect</i><br><br><i>Architecture</i> | <i>Types of problem solvers</i>         |                                                               |                                         |                                            |                                         |                                           |
|------------------------------------------------------|-----------------------------------------|---------------------------------------------------------------|-----------------------------------------|--------------------------------------------|-----------------------------------------|-------------------------------------------|
|                                                      | <i>Interaction</i>                      | <i>Invocation</i>                                             | <i>Expandability</i>                    | <i>Reactiveness</i>                        | <i>Learning</i>                         |                                           |
| ten Teije and van Harmelen                           | class C meta-level coordination         | depends on the goal and characteristics of a specific problem | add & remove methods                    | multiple solving methods                   | X                                       | X >                                       |
| DeJongh and Smith                                    | class A loosely coupled                 | hierarchical from abstract to detailed model                  | depends on domain goals & generic goals | multiple problem solvers (multiple models) | X                                       | X                                         |
| van Heijst et al.                                    | X                                       | X                                                             | X                                       | multiple problem solvers                   | X                                       | X                                         |
| Voß et al.                                           | class A loosely coupled                 | depending on time, relaxation, number of solutions            | depends on behaviour                    | specialists depending on behaviour         | X                                       | simple (case-based) strategic learning    |
| M-KAT                                                | class C meta-module                     | depends on conditions specified by the user                   | add & remove problem solvers            | multiple problem solvers                   | X                                       | X                                         |
| INRECA                                               | class A loosely coupled/tightly coupled | predefined depending on chosen technology                     | X                                       | inductive & case-based problem solvers     | switching between solvers               | algorithms improving case-based reasoning |
| STREAM                                               | class C meta-level coordination         | depending on strategic rules                                  | add & remove methods                    | different methods using different models   | reactive planning using strategic rules | X                                         |
| ANALOG modelling framework                           | class C meta-level inferencing          | meta-level inferencing                                        | add & remove methods                    | different methods using elementary methods | meta-level decision (impasses)          | multiple learning methods                 |

expandability and reactivity aspects. Thus new methods can be added and deleted and impasses are used for meta-level decisions regarding the continuation of a problem solving process.

### 3.1.9 Concluding remarks

Table 1 summarizes the analyses of the selected general, hybrid architectures given above.

Even though most of the architectures discussed make use of a class C meta-level interaction approach, they provide only simple forms of meta-level coordination. For example, the M-KAT architecture and the architecture proposed by ten Teije and van Harmelen stress the necessity and the benefits of using a meta-level coordination and control of multiple problem solvers, but they do not provide any concrete answers as to how such a meta-level unit should be realized. Other simple utilizations of the class C interaction approach are the use of meta-level strategic rules (STREAM) and the use of simple meta-level decisions based on factual knowledge (ANALOG).

From the architectures described only STREAM refers to a strategic invocation of different problem solvers by making use of meta-level strategic rules. A limitation is that the strategic rules used depend on factual knowledge characteristics of a problem case, and not on global problem solving parameters. Additionally, no improvement of the invocation aspect through experience is provided. Voß et al. consider the invocation of the defined specialists to depend on specific problem requirements (parameters), like time to be spent and number of solutions. DeJongh discusses a hierarchical definition and invocation of multiple models working in a collaborative fashion, and in M-KAT the invocation of the different problem solvers depends on the user. INRECA offers four

different ways of integrating induction and case-based reasoning. The user is responsible for deciding which one of the four ways it should be used for a specific application. Once the way of integration is chosen, the invocation of the two different reasoners is predefined.

Most of the general hybrid architectures described refer to the expandability aspect, but they do not discuss any ways for avoiding efficiently any side effects resulting from the addition or deletion of problem solvers. There is no doubt that system expandability is an essential aspect in a hybrid system. A way of supporting this aspect is to make the metareasoner responsible for keeping a global list of the solvers defined and the dependencies between them. Every time a new problem solver is added, its meta-level specification is appropriately incorporated in the list maintained by the metareasoner.

Furthermore most of the hybrid architectures described leave open the issue of reactively adjusting the initial plan for solving a problem case. Hybrid efforts addressing the reactivity issue provide only simple ways for taking into consideration new events occurring dynamically during a problem solving process. Such ways include the use of a predefined switching between problem solvers if an activated solver fails (INRECA) or simple strategic rules (STREAM) for changing over to another problem solver. The BOLERO system supports reactivity by taking into consideration new information obtained as a consequence of the execution of a plan and by generating at a meta-level a new plan to be followed. An interesting approach is presented by the ANALOG architecture. ANALOG makes use of impasses for signaling difficulties occurring during a problem solving process, and meta-level decisions are undertaken for resolving such situations.

In general, most of the architectures mentioned above do not address the strategic learning aspect. The only architectures addressing the strategic learning aspect, though in a rather weak way, are the ones by Voß et al., INRECA and ANALOG. For example in Voß et al.'s proposal, sequences of solver invocations are learned by one of the defined specialists (the case-based specialist), and in the fourth integration level of INRECA inductively learned retrieval algorithms are remembered to improve the use of case-based reasoning. In the knowledge modelling framework developed within the ANALOG project, learning is a kind of metalevel inferencing where successful methods and impasses arising during problem solving are remembered for future use.

### 3.2 *Application specific hybrid systems*

Application specific hybrid systems are developed for capturing the requirements of specific applications. Most of the hybrid knowledge-based systems integrate just two problem solvers which are invoked in a predetermined fashion. Hunt and Miles (1994) argue that case-based reasoning is a promising technology for building hybrid systems. In such systems a case-based reasoner co-operates with one or more reasoners which employ a different type of reasoning mechanism (e.g., qualitative simulation, constraint satisfaction systems, rule-based reasoning, etc.). These so-called "co-reasoners" can be used, for example, for augmenting the reasoning processes of case matching, case selection, case adaptation, case testing and case repair of a case-based reasoner, or for providing an alternative way to a partial or complete solution to a problem.

The majority of hybrid case-based reasoning systems make use of a class A tightly coupled approach for interaction (see Table 2). Examples of systems making use of a class A tightly coupled approach are given in subsections 3.2.1–3.2.4. Examples of systems making use of a class A loosely coupled approach are given in subsections 3.2.5–3.2.9 and of systems making use of a class C interaction approach are given in subsections 3.2.10 and 3.2.11.

#### 3.2.1 *CASEY*

The CASEY system (Koton, 1988, 1989, 1993) integrates two different problem solvers: a causal and a case-based problem solver. The system's domain is heart diseases diagnosis.

**Description** The system diagnoses a new problem case by recalling a previous, similar case and modifying that case to fit the current case. Because it is unlikely that any new problem will exactly

**Table 2** Evaluation of application specific hybrid systems

| Hybrid system   | Integration aspect                    |                                                         |               |                                          |                                       |                                          |
|-----------------|---------------------------------------|---------------------------------------------------------|---------------|------------------------------------------|---------------------------------------|------------------------------------------|
|                 | Interaction                           | Invocation                                              | Expandability | Types of problem solvers                 | Reactiveness                          | Learning                                 |
| CASEY           | class A tightly coupled               | hierarchically                                          | X             | 2 solvers causal & case-based            | X                                     | case knowledge                           |
| CABARET         | class A loosely coupled               | depending on heuristics                                 | X             | 2 solvers case-based & rule-based        | heuristical switching between solvers | case knowledge                           |
| BOLERO          | class C meta-level strategic planning | depending on strategic plans                            | X             | 2 solvers case-based & rule-based        | reactive behaviour                    | strategic & case knowledge               |
| Macchion and Vo | class A loosely coupled               | hierarchical switching between solvers                  | X             | 3 solvers model, case-based & rule-based | X                                     | case knowledge                           |
| CHROMA          | class C meta-level inferencing        | simple meta-level decision (based on factual knowledge) | X             | 2 case-based & 2 inductive methods       | meta-level decision                   | impasse-driven learning & case-knowledge |

match a previous one, the system evaluates differences between the problems by using a set of principles for reasoning about evidence in causal explanations. This means that the causal problem solver is a collaborator of the case-based reasoner regarding case adaptation.

**Analysis** CASEY makes use of a class A tightly coupled interaction approach. There are only two problem solvers involved and these are hierarchically invoked. The strategic learning and reactiveness aspects are not addressed and the solvers exhibit only collaborative behaviour. It is noted, of course, that such aspects are meaningless if there are just two tightly coupled collaborators.

### 3.2.2 Golding and Rosenbloom's system

In the system developed by Golding and Rosenbloom (1991), a rule-based and a case-based solver are used for the full-scale task of pronouncing surnames.

**Description** The central idea is to apply the rule-based solver to a target problem to get a first approximation to the answer; but if the problem is judged to be compellingly similar to a known exception of the rules in any aspect of its behaviour, then that aspect is modelled after a case-based model rather than the rules. The system performs rule-based reasoning with whatever imperfect rules are available, and supplements the rules with case-based reasoning. This enables the system to tap a knowledge source that is often more readily available than additional rules; namely, sets of examples (cases) from the domain.

**Analysis** The learning, reactiveness and expandability aspects are not addressed. The case-based solver is used to improve the performance of the rule-based solver. The solvers are collaborators. The invocation of the solvers is hierarchical having the rule-based solver starting every new problem case.

### 3.2.3 Nakatami and Israel's system

Nakatami and Israel (1993) propose an approach for combining a rule-based and a case-based

problem solver, especially in domains for which precise and exceptionless rules are known to be unavailable. The approach is implemented for the domain of cooking.

**Description** In this system a rule-based solver is tuned by a case-based solver. When the result of execution of a rule is not satisfactory, the system stores in a case base the name of the executed rule, the conditions under which the rule was executed, an overall evaluation, and a representation of what the user judges to be the factors that account for the unsatisfactory performance together with the rule element to be modified. The next time the rule is to be executed under the same conditions, the case base is searched for a relevant case. If there is no such case stored, the rule is executed without modification. If the most relevant case is decided, the relevant elements are modified by reference to the alternatives in the case. After a certain number of such experiments, the system can reconstruct the whole rule base by referring to the stored cases.

**Analysis** The two solvers involved in the system are tightly coupled in a class A interaction approach and a hierarchical invocation of the solvers is followed. The proposed approach does not address any of the other integration aspects.

### 3.2.4 *Portinale's system*

Portinale (Portinale et al., 1993; Portinale & Torasso, 1995) presents an approach to the integration of case-based reasoning with model-based reasoning. The approach is implemented in a mechanical (car troubleshooting) and a medical (hanseniasis diagnosis) domain.

**Description** The integration is exploited by defining adaptation criteria on solutions retrieved by a case-based problem solver, in order to focus the model-based problem solver in the search for the solution of the current case and to avoid, as much as possible, the computation of the solution from scratch. The proposed system is similar in some aspects to the CASEY system, however this system relies on a precise and general theory of model-based diagnosis.

**Analysis** A case-based solver is used to focus a model-based solver in an attempt to augment the basic mechanisms of the model-based solver by taking experience into account. The solvers are tightly coupled and their invocation is predetermined. None of the other integration aspects are addressed.

### 3.2.5 *CABARET*

The CABARET system (Rissland & Skalak, 1989, 1991) integrates a case-based solver with a rule-based solver by making use of a set of control heuristics. The application area of the system is legal argumentation.

**Description** The system presents a heuristically controlled approach for combining reasoning with cases and rules. The proposed approach is grounded in a set of control heuristics that determine what reasoning tasks to perform, given the states of the problem solvers. Each problem solver is capable of running in a stand-alone manner and has a dedicated reporter process that reports the end results of the solver and certain aspects of its intermediate processing. A controller module uses the observations of the reporters and its library of control heuristics (Skalak & Rissland, 1992) to decide how the system as a whole and the individual reasoning processes are to proceed.

**Analysis** The involved solvers are of a class A loosely coupled approach. The rule-based solver acts as a competitor/collaborator to the case-based solver and the invocation of the solvers depends on control heuristics provided by a controller module. The controller module is a very simple form of a meta-controller in our definition, because it does not provide any dynamic planning of problem solving strategies or any adaptation of planned strategies. Thus the reactivity, strategic learning and expandability aspects are not addressed.

### 3.2.6 ROUTER

ROUTER (Goel & Callentine, 1992; Goel et al., 1991) is a system for path planning which uses both a case-based solver and a numerical model-based solver. The application area is route planning.

**Description** Router selects heuristically the best solver to be used for every new problem case. The solvers are showing just competitive behaviour.

**Analysis** The solvers present a class A loosely coupled interaction approach. Heuristics are used for the invocation of the solvers and all the other integration aspects are ignored in the proposed system.

### 3.2.7 IKBALS III

IKBALS III (Zelevnikow et al., 1993, 1994; Zelevnikow & Hunter, 1993) integrates rule-based and case-based reasoning by using a distributed agent approach.

**Description** The system provides developers of legal knowledge-based systems with a unified approach to combining legal rule-based reasoning with case-based reasoning. The problem solvers involved in the system have been designed and implemented as separate agents. The system establishes a common communication protocol that the agents use when sending messages across the system. The knowledge structures used by both the case-based and the rule-based reasoning modules coexist in an external knowledge-based object and the agents gain access to this rich pool of knowledge by communicating via an interface.

**Analysis** The system is of a class A loosely coupled interaction approach. Agents in the IKBALS III system are autonomous and once invoked by a user event, they execute their specific task without requiring intervention. The agents show just competitive behaviour. The proposed system does not address the reactivity, expandability and learning aspects.

### 3.2.8 Chi and Kiang's system

Chi and Kiang (1993) present another system making use of a rule-based and a case-based problem solver. The system task is personnel-evaluation support.

**Description** For a new problem case the system starts with the rule-based solver and if no solution is found the case-based solver is activated. If neither of the solvers can provide a solution, a domain theory is used to generalize old cases and the problem solving continues by making use of the generalized cases.

**Analysis** The invocation of the involved solvers is hierarchical, always starting with the rule-based solver and continuing with the case-based solver only if the former fails to solve the problem case in hand. The solvers exhibit class A loosely coupled interaction working in a competitive way. Most of the integration aspects (e.g., learning, expandability, reactivity) are not addressed in this system.

### 3.2.9 Macchion and Vo's system

In the system by Macchion and Vo (1993) model-based, case-based and rule-based reasoning techniques are utilized for solving problems in fault diagnosis in a technical domain.

**Description** Within the model-based reasoner, the system to be diagnosed is described under functional, structural and causal aspects; basic principles operating on such technical aspects allow a systematic diagnostic process. Within the rule-based reasoner, the solutions of the most frequent incidents are synthesized into rules which can be either manually supplied or automatically generated.

**Analysis** The strategy used for invoking the three reasoners is fixed. Solving a target incident consists of a sequential activation of the rule-based, case-based and model-based reasoners. This

means that there is no real interaction between the different solvers. The learning, reactivity and expandability aspects are also not addressed.

### 3.2.10 BOLERO

A hybrid system exhibiting reactive behaviour and metalevel planning is the BOLERO system (Lopez, 1993; Lopez & Plaza, 1993) developed for the domain of pneumonia diagnosis.

**Description** The system integrates a case-based and a rule-based solver. The case-based solver is capable of learning and building problem solving plans. It acts as a meta-level planner. The rule-based solver provides solutions by chaining rules in order to validate its goals. The integration of the two reasoners aims to improve the performance of the rule-based solver by planning its task. The meta-level planner can modify the state of the rule-based solver through plans, and it is also in charge of starting and stopping the system execution. The case-based planner builds plans according to the state of the rule-based solver, when the latter is solving a problem, and plans are executed by the rule-based solver. When the rule-based solver obtains new information as a consequence of the execution of a plan, the case-based solver can generate a new plan, addressing the new circumstances. Thus, the validation of any goal in the rule-based solver can be interrupted, if the case-based planner needs to generate a new plan.

**Analysis** The strong aspects of BOLERO are the meta-level planning and the ability to learn strategic knowledge from its own experiences. That is, when a new problem is solved, a new case is obtained. The case consists of all the facts gathered during problem solving plus the final plan. The new case can be incorporated in the system taking into account the successes and failures of plans. This way the system learns problem solving plans (strategies) from its own experiences. Even though the system presents an interesting integration of a meta-level planner and a rule-based reasoner, it is restricted to only two problem solvers with very specific roles, one for planning tasks and the other for executing a planned (sub)task. A possible extension of the system that supports higher flexibility is to use different solvers for executing a planned task, and to make the planner responsible for deciding which solver to invoke for a specific problem case. This implies that the planner should incorporate knowledge regarding the abilities of different problem solvers and the dependencies between them. Different strategies could then be used for choosing among the solvers the best one for a particular problem case. In short, the planner is promoted to a meta-controller. Because of the existence of only a single actual problem solver the system does not really address the invocation and expandability aspects. In general, the existing planner and problem solver interact according to a class C approach.

### 3.2.11 CHROMA

CHROMA (Armengol & Plaza, 1994, 1995), being developed within the ANALOG project, represents a hybrid system integrating induction and case-based reasoning for problem solving and learning.

**Description** The main task performed by CHROMA is the purification of proteins from biological sources (animal tissues, bacteriological cultures, etc.) using chromatographic techniques. CHROMA has to determine a purification plan using either a memory of cases or domain knowledge. Four different methods can be used for obtaining a purification plan: equal-sample, classify-by-prototype, analogy-by-determination and default-plan (Armengol & Plaza, 1994). The first two represent case-based methods and the last two domain-based methods. The system is implemented in the NOOS language (Arcos & Plaza, 1995). The basis for integrating four different inference methods is the capability of NOOS for ascribing several alternative methods to any task, selecting among them using a language of preferences, and automatically backtracking to other alternative methods.

**Analysis** In general, meta-knowledge about the applicability range of the different methods is used to select the most adequate method for every case to be solved. This meta-knowledge depends

only on some factual knowledge (quantity of protein produced) and not on meta-level features of the current problem. This means that the invocation of the methods is simple and no strategic planning is provided. The reactivity aspect is addressed in the proposed system by backtracking to other alternative methods if a selected inference method fails to provide appropriate solutions.

### 3.2.12 Concluding remarks

Table 2 summarizes the evaluations of the hybrid systems discussed above. All the discussed class A tightly coupled hybrid systems show many similarities and for this reason the CASEY system is taken as a representative of them in Table 2. For the same reason the CABARET system represents class A loosely coupled hybrid systems, in this table.

In general, the application specific hybrid systems reviewed above are characterized by the limited types of problem solvers involved. The actual solvers and their integration mechanisms in the above systems are strictly predefined (hardwired), so that any knowledge or reasoning changes would be difficult to incorporate, and may well result in undesirable side effects reducing the performance and reliability of such systems. In short, all the discussed hybrid systems leave open the expandability aspect. Thus, no allowance is made for increasing or reducing the number of the involved problem solvers.

Another common aspect of all these systems is the very simple forms of interaction between the different solvers. Most of the systems are tightly or loosely coupled and make use of the class A approach, meaning that the different solvers can communicate directly with one another. BOLERO is exceptional in that it supports a class C metalevel strategic planning. In BOLERO, the strategy to be followed is planned by a metalevel case-based solver, and is then executed by a rule-based solver. A limitation of BOLERO is that only one problem solver, a rule-based solver, is responsible for executing the planned actions. As mentioned previously an improvement of the system could be to increase the number of the problem solvers responsible for executing plans and to choose among them the best one for a given situation. Another limitation of the BOLERO system is that the planning of solving strategies is based only on case factual knowledge and not on global problem solving parameters or characteristics of the specific problem case.

The invocation of the integrated solvers by class A tightly coupled systems is hierarchical. Some problem solver derives a solution and another one is, for example, justifying, adapting or focusing the solution of the first solver. Class A loosely coupled systems make also a hierarchical switching or they use heuristics for the invocation of different solvers. CHROMA integrates inductive and case-based reasoning and learning methods for problem solving. Even though a metalevel reasoner is used for the invocation of the different methods, the actual invocation is simple, based only on some factual knowledge.

The reactivity aspect is left open by most of the discussed hybrid systems. BOLERO can be distinguished from the other systems by its reactive behaviour. In this system important events occurring during the solving process are taken into consideration and new plans are constructed, thus reflecting the dynamic nature of the requirements imposed on the problem solving process.

Finally, the issue of strategic learning is also left open by most of the hybrid systems. Again only BOLERO addresses a simple form of strategic learning by memorizing successful strategic plans.

## 4 Essential components of a competent hybrid architecture

The use of multiple problem solvers has become more and more popular in the knowledge-based systems community in recent years. A number of hybrid systems have been developed in an attempt to overcome the limitations of systems making use of only one problem solver (Keravnou & Washbrook, 1989) and thus to achieve efficient and effective problem solving. Even though all of the efforts developed stress the necessity of using multiple problem solvers to improve the performance of knowledge-based systems, they do not directly address all the essential aspects that are necessary for the transparent integration of different problem solvers which can be used both competitively and collaboratively. In the hybrid systems developed, the integrated problem solvers are used either

competitively or collaboratively, but not both. All the hybrid systems reviewed are characterized by the limited types of the integrated problem solvers, the fact that no allowance is made for increasing or reducing the number of the involved solvers, and that no meta-classification of the problem solvers from the perspective of solution types is provided. The problem solvers in most of these systems are integrated in a predetermined way and their invocation is predefined and mostly hierarchical. Some of the proposals make reference to meta-level coordination and control of the multiple problem solvers, but they do not provide any concrete answers as to how this ought to be achieved. Some proposals make use of simple meta-level strategy rules or meta-level decisions based on factual knowledge for invoking the different problem solvers. The critical issue of strategic learning is either left completely open or marginally addressed.

In summary, all the general hybrid architectures and application specific hybrid systems that have been investigated appear to leave open a number of issues regarding the integration of multiple problem solvers. Such issues include a meta-level classification of problem solvers, meta-level communication and control of different problem solvers, strategic invocation of the appropriate problem solvers, dynamic adapting of strategies and overall strategic learning. Because of the complexity and globality of all these issues, they are best addressed in the context of a metareasoner responsible for controlling the whole solving process. In this section we give a high level specification for a *competent* hybrid knowledge-based architecture<sup>2</sup> by pointing out the essential components comprising it and their interrelationships. A hybrid system to be competent must provide adequate solutions to all identified integration aspects.

A competent hybrid knowledge-based architecture integrates multiple problem solvers for the same (sub)task, each of which represents a different way for executing the particular (sub)task. Each problem solver can operate on a stand alone basis, but can also collaborate with other solvers associated with the same (sub)task. This means that the different solvers, for the same task, show a “competitive-collaborative” behaviour. The decision as to which of the competitor solvers to use for a particular problem case is taken dynamically depending on specific requirements (time considerations, level of detail of the solution, etc.). If during problem solving a chosen solver is unable to deliver the required intermediate or final results, then other solvers can be used, either for replacing (competitive mode) or supporting (collaborative mode) the failing solver. Thus dynamic decisions need to be taken at different points during the problem solving activity. These decisions can be quite complex and should be taken by a meta-control unit that has a global view of the system’s behaviour. Hence at the top level, essential components of a hybrid knowledge-based system, are its problem solvers, i.e., the problem solvers associated with the overall task performed by the system, and its metareasoner performing the critical integration meta-tasks. Thus, a competent hybrid knowledge-based architecture exhibits class C interaction (see Figure 1).

#### 4.1 Problem solvers

A problem solver is a unit implementing a knowledge representation formalism combined with a reasoning method. The unit is partitioned into a *domain module* and a *control module* (Figure 2). The domain module stores information about domain objects, their properties and their relations. It defines the relevant domain ontology. The control module is partitioned into an *inference part* and a *local controller*. The inference part gives the reasoning method to be used on the knowledge in the domain module. The local controller monitors the execution of the problem solver for a particular problem case. It is used, for example, for storing intermediate and final results, for checking the satisfiability of constraints applied to the problem solver’s domain module, etc. It is also responsible

<sup>2</sup>The given high level specification is demonstrated in practice through a hybrid knowledge-based system for the domain of breast cancer histopathology (Christodoulou, 1994, 1995). The developed system uses three different problem solvers, each capable of solving the given problem on its own, a rule-based, a case-based and a deep-model solver. All three solvers can work in a competitive/collaborative way. The metareasoner of the system controls the overall system.

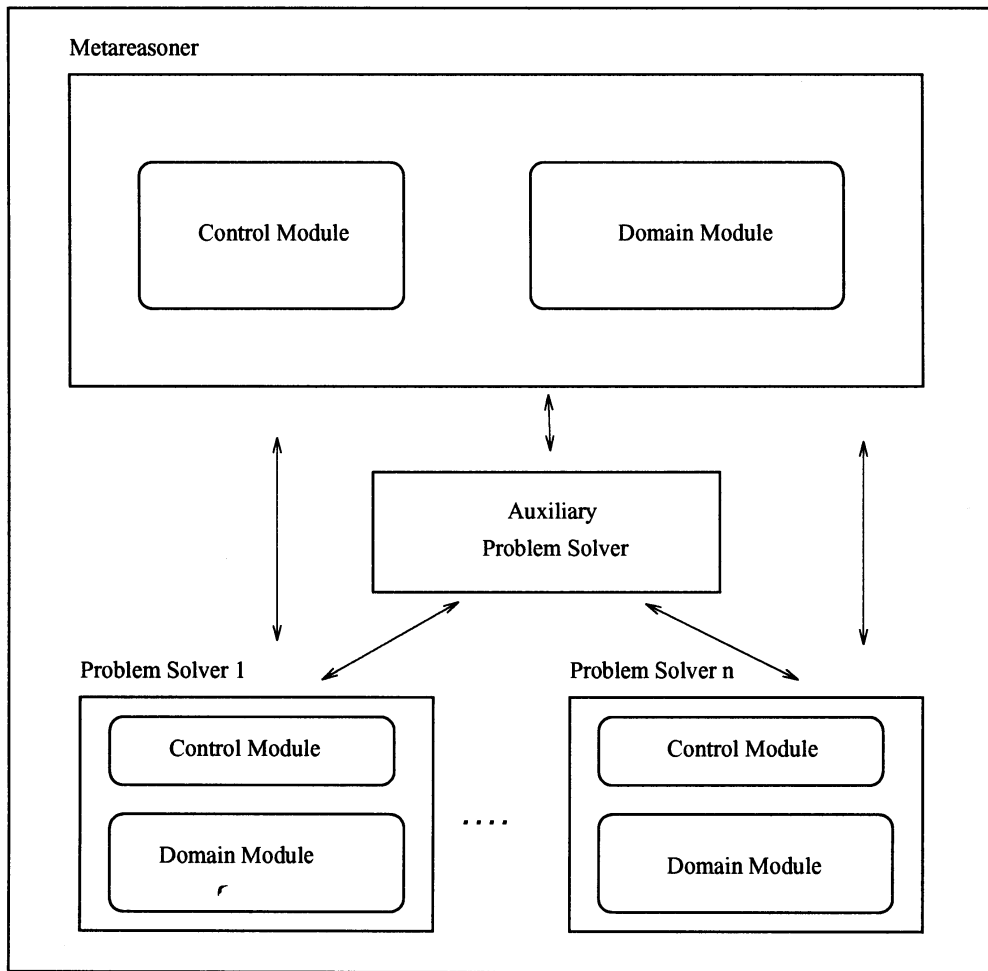


**Figure 2** A hybrid architecture

for the communication of the solver with the metareasoner. To avoid redundancy and duplication of problem specific information, an auxiliary problem solver could be used (Figure 2). The domain module of the auxiliary problem solver contains a global structure for storing information about the problem in hand. Every other problem solver has access to this information and can directly call the auxiliary problem solver. The control module of the auxiliary problem solver provides mechanisms for checking constraint violations with respect to the overall domain knowledge. Finally, the auxiliary problem solver provides a uniform way for the transfer of information between the other problem solvers.

The types of problem solvers in a hybrid architecture are determined by the different types of solutions which are generally required for a specific (sub)task. For example, the task of medical diagnosis could be achieved in different ways depending on different circumstances (ten Teije & van Harmelen, 1995). If the patient's symptoms point to an urgent case then a less detailed diagnosis is needed initially to contain the patient's situation, while a more detailed diagnosis is carried out at a later point for long term patient management. These two different but, depending on the circumstances, acceptable types of solution for the task of medical diagnosis could be realized as two different problem solvers, to be used both competitively and collaboratively.

#### 4.2 Metareasoner

A metareasoner (Figure 2) is itself a problem solver, and consists of a domain module and a control module. The metareasoner has a critical role in a competent hybrid architecture dealing with the

efficient and effective integration of the other problem solvers. To provide adequate solutions to the different integration aspects, the domain module of the metareasoner should contain knowledge about the different problem solvers and how they should be used and controlled, while its control module should provide appropriate inference mechanisms.

### **The metareasoner's domain module**

The metareasoner's domain module consists of two different types of knowledge, *descriptive* and *strategic* knowledge.

#### *Descriptive knowledge*

Descriptive knowledge consists of knowledge describing the different solvers involved in the system, the dependencies between them and constraints underlying such dependencies. Knowledge describing a problem solver can include, for example, information about the maximum length of time needed by the solver to achieve a solution, information about the complexity of the domain knowledge used by the solver, as well as dynamic information about the problem solver's status (activated, stopped) in the context of a specific problem solving activity. Knowledge describing dependencies between solvers refers to information about which solvers could support others, or are competitors, etc.

#### *Strategic knowledge*

Strategic knowledge defines strategies for problem solving. A problem solving strategy gives a combination of problem solvers able to solve a specific type of problem cases. A strategy is chosen on the basis of parameters describing a desirable solution to the problem case and the characteristics of the problem solvers. The metareasoner is responsible for matching the required parameters against the characteristics of appropriate solvers. A problem solving strategy could be constructed from scratch or it could be retrieved and adapted from a memory of already existing problem solving strategies. Such a memory is part of the metareasoner's strategic knowledge and is created gradually through problem solving activities.

### **The metareasoner's control module**

The metareasoner's control module provides inference mechanisms responsible for:

- User communication.
- Planning of problem solving strategies (supporting the invocation aspect).
- Reactively adapting chosen strategies (supporting the reactivity aspect).
- Activating/deactivating problem solvers (supporting the invocation aspect).
- Learning problem solving strategies (supporting the learning aspect).
- Updating the problem-solvers-definition-list, by adding or deleting descriptions of solvers (supporting the expandability aspect).

#### *User communication*

The metareasoner is responsible for the communication with the user of the system in the context of acquiring the meta-characteristics of a specific case, like time limits in which a solution should be achieved, or the level of detail to which a solution should be worked out. Additionally, it guides the user, so that there are no contradictions between the meta-characteristics entered and it also acquires from the user the main piece of information describing the problem in hand. This information is passed to the auxiliary problem solver (see above) for formulating a first approximation to the solution of the problem. The metareasoner is also responsible for evaluating the results achieved by the different problem solvers and for presenting these results to the user.

#### *Planning problem solving strategies*

A problem solving strategy is a plan of problem solver activations. A strategy is planned on the basis of meta-characteristics of the problem case, and problem solver characteristics. To avoid a conflict between multiple possible strategies, probably giving the same degree of success, preferences should be associated with the meta-characteristics of problem cases. Such preferences could, for example,

indicate that a time efficient solution is preferred over a detailed solution. This way, priorities are assigned to the various strategies. The strategy with the highest priority should be followed first, while strategies with lower priorities should be listed as alternatives if the chosen strategy fails to give acceptable solutions. To increase the performance of a hybrid architecture, after choosing some strategy problem solving should be focused around the first solution approximation derived by the auxiliary problem solver. In general, the use of problem solving strategies provides a powerful solution to the invocation aspect of a competent hybrid architecture.

#### *Reactively adapting chosen strategies*

A metareasoner should exhibit reactive behaviour, meaning that intermediate results should be continuously evaluated to detect whether the chosen strategy should be replaced by another or suitably adapted to address new requirements. This means that the reactivity aspect should be placed under the realms of the metareasoner. During problem solving, a *solving path* tracing the whole activity in a detailed way should be kept by the metareasoner for controlling the overall process. The results achieved by the chosen strategy should be recorded in the solving path. If such results are in conflict with the first solution approximation, the metareasoner could replace the operative strategy with another strategy, the one with the next highest priority, or the first solution approximation is reviewed to see if an adjustment is warranted. A strategy change is recorded in the solving path. This way, the metareasoner exhibits reactive behaviour. A paradigm, appropriate for strategic planning and reactive behaviour, is the case-based one. Both successful and unsuccessful solving paths are stored as strategic cases (as opposed to problem cases) for future use. A successful solving path is one that resulted in an acceptable solution. Future problem cases should be solved by retrieving and adapting existing strategic cases.

#### *Activating / deactivating problem solvers*

After the selection of some strategy, the metareasoner sets the strategy in motion by following the activation sequence of the relevant problem solvers and resolving potential choice points (as specified by the strategy). Thus, activating / deactivating problem solvers is essential for realization of the invocation aspect of a hybrid system. An activated problem solver should be given a time limit during which it should come up with a solution, and its status is set to “activated”. When the given time limit is over, if the solver is still active, it should be deactivated (solver’s status is set to “stopped”). Intermediate results obtained by the given solver are already memorized by its local controller. The metareasoner has direct access to the solver’s local controller for analysing the intermediate results and for deciding which execution steps should be undertaken next. Intermediate results are of different types. Such results could indicate that the problem solver should be supported by another problem solver to achieve final conclusions, or that the specific problem solver is unable to reach a conclusion because of missing information, and such information should be supplied by the metareasoner without switching to another problem solver. The decision about which solvers should support others, and under which actions, is based on the relevant part of the metareasoner’s descriptive knowledge.

#### *Updating the problem-solvers-definition-list*

The metareasoner is responsible for supporting the expandability potential of a hybrid architecture. Under expandability we refer to later additions or deletions of problem solvers by a systems engineer in order to adapt to new demands regarding present and future requirements for a specific task. Such changes can be supported by keeping at the metareasoner level a global list of all the existing problem solvers, together with their dependencies. Depending on the requirements of a specific application, it should, for example, be possible to include in the global list the specifications of new problem solvers without influencing the existence of other solvers.

#### *Learning*

Learning is an essential aspect of any viable knowledge-based system. It reflects the ability of a system to evolve on the basis of its accumulated experiences. Most of the existing knowledge-based systems and architectures provide learning mechanisms for remembering the details of past,

successfully solved cases. This type of learning refers to knowledge level learning. A hybrid knowledge-based architecture should additionally support strategic level learning (Christodoulou & Keravnou, 1997). Because of the globality and complexity of strategic knowledge, the metareasoner should be responsible for providing different learning mechanisms for strategic knowledge. Both successful and unsuccessful strategies should be remembered. This means that the characteristics of a planned strategy, like the meta-characteristics of a required solution, priorities set on these characteristics and the sequence of the invoked problem solvers should be saved in a specific memory of problem solving strategies. Additionally, reactive adaptations of a strategy occurring during problem solving should also be remembered. The learning of strategies resulting in no solution is also important because unsolved strategies could be used for the extraction of general failure rules for avoiding such failure situations in the future.

## 5 Conclusions

Multiple problem solvers representing different ways for achieving a specific (sub)task, could be used by a knowledge-based system in the context of the particular (sub)task. Deploying a number of solvers in a competitive and collaborative way enhances the performance of a system. A knowledge-based system integrating multiple problem solvers should take into consideration a number of critical integration aspects, such as interaction, invocation, reactivity, learning, types of problem solvers and expandability. Most of the existing general hybrid architectures and application specific hybrid systems make use of rather naïve interaction, invocation and learning mechanisms. A hybrid architecture supporting the different integration aspects, should have a metareasoner as a critical entity of the system, able to effectively and flexibly integrate the different problem solvers. For this purpose, strategic knowledge should be provided for planning appropriate problem solving strategies. An ability to change planned strategies reactively depending on intermediate results is also a must and such adaptations should also be remembered for future use.

## Acknowledgements

The research by E. Christodoulou reported in this paper was financially supported by the University of Cyprus through a research grant. The authors would like to thank the two anonymous referees for their very useful comments on a previous version of the paper.

## References

- Aamodt, A, 1993. "A case-based answer to some problems of knowledge-based systems" In: *Proc. Scandinavian Conference on Artificial Intelligence* 168–182, E Sandewall and CG Jansson (eds) IOS Press.
- Aamodt, A, 1994a. "Explanation-driven case-based reasoning" In: S Wess, K Althoff and M Richter (eds) *Topics in Case-based Reasoning* 274–288, Springer-Verlag.
- Aamodt, A, 1994b. "A knowledge representation system for integration of general and case-specific knowledge" In: *Proc. IEEE International Conference on Tools with Artificial Intelligence (TAI-94)* New Orleans, November 5–12 1994.
- Aamodt, A, 1995. "Knowledge acquisition and learning from experience – the role of case-specific knowledge" In: G Tecuci and Y Kodratoff (eds) *Machine Learning and Knowledge Acquisition: Integrated approaches* 197–245, Academic Press.
- Aamodt, A and Nygaard, M, 1995. "Different roles and mutual dependencies of data, information and knowledge – an AI perspective on their integration" *Data and Knowledge Engineering* **16** 191–222.
- Althoff, K-D, Bergmann, R, Maurer, F, Wess, S, Manago, M, Auriol, E, Conruyt, N, Traphöner, R, Bräuer, M and Dittrich, S, 1993. "Integrating inductive and case-based technologies for classification and diagnostic reasoning" In: *Proc. ECML-93 Workshop on Integrated Learning Architectures* E Plaza (ed).
- Althoff, K-D, Bergmann, R, Maurer, F, Wess, S, Manago, M, Auriol, E, Conruyt, N, Traphöner, R, Bräuer, M and Dittrich, S, 1994. "Induction and case-based reasoning for classification Tasks" In: HH Bock, W Lenski and MM Richter (eds) *Information Systems and Data Analysis, Prospects-Foundations-Applications* 3–16, Springer-Verlag.
- Althoff, K-D, Wess, S and Traphöner, R, 1995a. "INRECA – A seamless integration of induction and case-

- based reasoning for decision support tasks" In: *Proc. 8th Workshop of the German Special Interest Group on Machine Learning* K Morik and J Herrmann (eds) 116–122.
- Althoff, K-D, Auriol, E, Bergman, R, Breen, S, Dittrich, S, Johnston, R, Manago, R, Traphöner, R and Wess, S, 1995b. "Case-based reasoning for decision support and diagnostic problem solving: The INRECA approach" In: *Proc. XPS-95-Workshop: Fallbasiertes Schliessen – Grundlagen und Anwendungen* B Bartsch-Spörl, D Janetzko and S Wess (eds).
- Arcos, JL and Plaza, E, 1994a. "Integration of learning into a knowledge modelling framework" In: *Proc. 8th European Knowledge Acquisition Workshop* L Steels, AT Schreiber and W van de Velde (eds) *Lecture Notes in Artificial Intelligence* 867 355–373, Springer-Verlag.
- Arcos, JL and Plaza, E, 1994b. "A reflective architecture for integrated memory-based learning and reasoning" In: *Proc. First European Workshop on Topics in Case-Based Reasoning* S Wess, K-D Althoff and M Richter (eds) *Lecture Notes in Artificial Intelligence* 837 289–300, Springer-Verlag.
- Arcos, JL and Plaza, E, 1995. "Reflection in NOOS: An object-centered representation language for knowledge modelling." In: *Proc. IJCAI'95 Workshop on Reflection and Meta-Level Architectures in AI* 1–10, Montreal, Canada.
- Armengol, E and Plaza, E, 1994. "A knowledge level model of knowledge-based reasoning" In: *Proc. First European Workshop on Topics in Case-Based Reasoning*, S Wess, K-D Althoff and M Richter (eds) *Lecture Notes on Artificial Intelligence* 837 53–64, Springer-Verlag.
- Armengol, E and Plaza, E, 1995. "Integrating induction in a case-based reasoner" In: *Proc. Second European Workshop on Advances in Case-Based Reasoning* J-P Haton, M Keane and M Manago (eds) *Lecture Notes in Artificial Intelligence* 984 3–17, Springer-Verlag.
- Auriol, E, Manago, M, Althoff, K.-D, Wess, S and Dittrich, S, 1995. "Integrating induction and case-based reasoning: methodological approach and first evaluations" In: J-P Haton, M Keane and M Manago (eds) *Advances in Case-based Reasoning* 18–32, Springer-Verlag.
- Baddeley, AD, Bressi, S, Della Sala, S, Logie, RH and Spinnler, H, 1991. "The decline of working memory in Alzheimers disease: A longitudinal study" *Brain* **114** 2521–2542.
- Bartsch-Spörl, B, Bredeweg, B, Coulon, C-H, Drouven U, van Harmelen, F, Karbach, W, Reinders, M, Vinkhuyzen, E and Voß, A, 1991. "Studies and experiments with reflective problem solvers" *ESPRIT Basic Research Action P3178 REFLECT, Report IR.3.1,2 RFL/BSR-UvA/II.2/1* REFLECT Consortium, August.
- Chandrasekaran, B, 1982. "Decomposition of domain knowledge into knowledge sources: the MDX approach" In: *Proc. 4th National Conference of Canadian Society for Computational Studies of the Intelligence*.
- Chandrasekaran, B, 1986. "Generic tasks in knowledge-based reasoning: high level building blocks for expert design" *IEEE Expert* **1** 23–30.
- Chandrasekaran, B, 1988. "Generic tasks as building blocks of knowledge-based systems: the diagnosis and routine design examples" *The Knowledge Engineering Review* **3** (3) 183–210.
- Chandrasekaran, B and Johnson, TR, 1993. "Generic tasks and task structures: history, critique and new directions" In: JM David, JP Krivine and R Simmons (eds) *Second Generation Expert Systems* 239–280, Springer-Verlag.
- Chandrasekaran, B and Mittal, S, 1983. "Conceptual representation of medical knowledge for diagnosis by computer: The MDX and related systems" *Advances in Computers* **22** 217–293.
- Chi, MTH, Glaser, R and Rees, E, 1982. "Expertise in problem solving" In: RS Sternberg (ed) *Advances in the Psychology of Human Intelligence Vol 1* Erlbaum.
- Chi, RTH and Kiang, MY, 1993. "Reasoning by coordination: an integration of case-based and rule-based reasoning systems" *Knowledge-Based Systems* **6** (2).
- Christodoulou, E and Keravnou, E, 1997. "Meta-level learning in a hybrid knowledge-based architecture" In: *Proc. 6th Conference on Artificial Intelligence in Medicine Europe (AIME'97): Lecture Notes in Artificial Intelligence* **1211**, 467–478 Springer-Verlag.
- Christodoulou, E, 1995. "A hybrid architecture for knowledge-based systems" In: *Proc. 5th Conference on Artificial Intelligence in Medicine Europe (AIME'95)* P Barahona, M Stefanelli and J Wyatt (eds) *Lecture Notes in Artificial Intelligence* 934 421–422, Springer-Verlag.
- Christodoulou, E, 1994. "An integrated decision support system for the domain of breast cancer" In: *Proc. International Conference Neural Networks and Expert Systems* EC Ifeachor, and KG Rosen (eds) 433–441, Plymouth, UK.
- Console, L, de Kleer, JH and Hamscher, WC (eds) 1992. *Readings in Model-based Diagnosis* Morgan Kaufmann.
- Cox, M and Ram, A, 1992. "Multistrategy learning with introspective meta-explanations" In: *Proc. 9th International Conference on Machine Learning* D Sleeman and P Edwards (eds) 123–128, Aberdeen, Scotland.
- DeJongh, M and Smith, J, 1989. "Bridging the knowledge and symbol levels: Towards an integrated knowledge

- engineering environment" In: *Proc. AAAI Spring Symposium on Knowledge System Development Tools and Languages*.
- DeJongh, M and Smith, J, 1991. "Integrating models of a domain for problem solving. Expert Systems and their applications" In: *Proc. Eleventh International Conference on Expert Systems and their applications, General Conference on Second Generation Expert Systems: Avignon 91* Vol 2, 125–135.
- DeJongh, M, 1991. "Integrating the use of device models with abductive problem solving" PhD thesis, The Ohio State University.
- Goel, A and Callentine, T, 1992. "An experience-based approach to navigational path planning" In: *Proc. IEEE/RSJ Int. Conf. on Robotics and Systems* IEEE Press.
- Goel, A, Callentine, T, Shanker, M and Chandrasekaran, B, 1991. "Representation and organization of topographic models of physical spaces for route planning" In: *Proc. IEEE Conference on Artificial Intelligence Applications* IEEE Press.
- Golding, AR and Rosenbloom PS, 1991. "Improving rule-based systems through case-based reasoning" In: *Proc. 11th International Conference Expert Systems and their Applications Vol 1 (Tools, Techniques and Methods)* 759–764.
- Hunt, J and Miles, R, 1994. "Hybrid case-based reasoning" *The Knowledge Engineering Review* 9 (4) 383–397.
- Karbach, W, Voß A, Schukey, R and Drouven, U, 1991. "MODEL-K: Prototyping at the knowledge level" In: *Proc. 11th International Conference Expert Systems and their Applications Vol 1 (Tools, Techniques and Methods)* 501–511.
- Keravnou, E and Johnson, L, 1986. *Competent Expert Systems: A case study in fault diagnosis* Kogan-Page.
- Keravnou, E and Washbrook, J, 1989. "What is a deep expert system: An analysis of the architectural requirements of second-generation expert systems" *The Knowledge Engineering Review* 4 (3) 205–233.
- Kolodner, JL, 1993. *Case-based Reasoning* Morgan Kaufmann.
- Koton P, 1988. "Integrating case-based and model-based reasoning: A symbiotic system" PhD thesis, MIT Laboratory for Computer Science.
- Koton P, 1989. "Using experience in learning and problem solving" *Technical Report MIT/LCS/TR-441* MIT Laboratory of Computer Science.
- Koton P, 1993. "Combining causal-models and case-based reasoning" In: JM David, JP Krivine and R Simmons (eds) *Second Generation Expert Systems* 46–68, Springer-Verlag.
- Lanzola, G and Stefanelli, M, 1992a. "A specialized framework for medical diagnostic knowledge based systems" *Computers and Biomedical Research* 25 351–365.
- Lanzola, G and Stefanelli, M, 1992b. "Knowledge acquisition for medical diagnosis" In: *Proc. VI Mediterranean Conference on Medical and Biomedical Engineering* F Denoth and M Bracale (eds) Vol 2, 825–828.
- Lanzola, G and Stefanelli, M, 1993a. "Computational Model 3.0" *AIM Project report A2034: A General Architecture for Medical Knowledge-Based Systems (GAMES-II)* deliverable 25, December 30.
- Lanzola, G and Stefanelli, M, 1993b. "Inferential knowledge acquisition" *Artificial Intelligence in Medicine* 5 253–258.
- Litcher, P, 1992. *A Complete Interdisciplinary Science of Mind* Bradford Books.
- Logie, RH, 1993. "Working memory in everyday cognition" In: GM Davies, GM and RH Logie (eds) *Memory in Everyday Life* 173–218, Elsevier.
- Lopez, B, 1993. "Reactive planning through integration of a case-based system and a rule-based system" In: *Proc. 9th Biennial Conference on Artificial Intelligence, Society of the Study of Artificial Intelligence and the Simulation of Behaviour (AISB-93)* University of Birmingham, UK.
- Lopez, B and Plaza, E, 1993. "Case-based planning for medical diagnosis" In: *Proc. 7th International Symposium on Methodologies for Intelligent Systems (ISMIS93)* Z Ras (ed) *Lecture Notes in Artificial Intelligence* 689 96–105, Springer-Verlag.
- Macchion, DJ and Vo, D-P, 1993. "A hybrid knowledge-based system for technical diagnosis, learning and assistance" In: *Proc. First European Workshop on Topics in Case-Based Reasoning* S Wess, K-D Althoff and M Richter (eds) *Lecture Notes in Artificial Intelligence* 837 301–312, Springer-Verlag.
- Maes, P and Nardi, D, 1988. *Meta-level Architectures and Reflection* North-Holland.
- Mittal, S, Chandrasekaran, B and Smith, J, 1979. "Overview of MDX – A system for medical diagnosis" In: *Proc. Third Annual Symposium on Computer Application in Medical Care* 34–46.
- Nakatani, Y and Israel, D, 1993. "Tuning rules by cases" In: *Proc. First European Workshop, Topics in Case-Based Reasoning (EWCBR-93)* University of Kaiserslautern, Germany, 319–324.
- Patel, VL and Groen, GJ, 1991. "The general and specific nature of medical expertise: A critical look" In: KA Ericsson and J Smith (eds) *Toward a General Theory of Expertise: Prospects and limits* 93–125, Cambridge University Press.
- Plaza, E, Aamodt, E, Ram, A, Van de Velde, W and Van Someren, M, 1993. "Integrated learning architectures" In: *Proc. European Conference on Machine Learning (ECML-93)* PV Brazdil (ed) *Lecture Notes in Artificial Intelligence* 667 429–441, Springer-Verlag.
- Plaza, E and Arcos, JL, 1993. "A reflective architecture for integrated memory-based learning and reasoning"

- In: *Proc. First European Workshop on Case-Based Reasoning (EWCBR-93)* University of Kaiserslautern, Germany, 329–334.
- Plaza, E and Arcos, JL, 1993b. “Using reflection principles in the integration of learning and problem solving” In: *Proc. ECML-93 Workshop on Integrated Learning Architectures (ILA-93)* Vienna, Austria.
- Plaza, E and Arcos, JL, 1993c. “Reflection and analogy in memory-based learning” In: *Proc. Second International Workshop on Multistrategy Learning* 42–49, Harpers Ferry, USA, May.
- Plaza, E and Arcos, JL, 1994. “Flexible integration of multiple learning methods into a problem solving architecture” In: *Proc. Machine Learning (ECML-94)* F Bergadano and L De Raedt (eds) *Lecture Notes in Artificial Intelligence* 784 403–405, Springer-Verlag.
- Portinale, L, Torasso, P, Ortalda, C and Giardino, A, 1993. “Using case-based reasoning to focus model-based diagnostic problem solving” In: *Proc. First European Workshop, Topics in Case-Based Reasoning (EWCBR-93)* University of Kaiserslautern, Germany, 335–340.
- Portinale, L, Torasso, P, 1995. “ADAPtER, an integrated diagnostic system combining case-based and abductive reasoning” In: *Proc. First International Conference on Case-Based Reasoning (ICCBR-95)* 277–288, Sesimbra.
- Punch, WF, 1989. “A diagnosis system using a task integrated problem solving architecture (TIPS), including causal reasoning” PhD Thesis, The Ohio State University.
- Punch, WF and Chandrasekaran, B, 1993. “An investigation of the roles of problem-solving methods in diagnosis” In: J-M David, J-P Krivine and R Simmons (eds) *Second Generation Expert Systems* 673–688, Springer-Verlag.
- Ram, A, Cox, MT and Narayanan, S, 1992. “An architecture for integrated introspective learning” In: *Proc. 9th International Conference on Machine Learning, Workshop on Computational Architecture* Aberdeen, Scotland.
- Ram, A and Cox, MT, 1994, “Introspective reasoning using meta-explanations for multistrategy learning” In: RS Michalski and G Tecuci (eds) *Machine Learning: A Multistrategy Approach Vol IV* 349–377, Morgan Kaufmann.
- Ramoni, M, Stefanelli, M, Magnani, L and Barossi, G, 1992. “An epistemological framework for medical knowledge-based systems” *IEEE Transactions on Systems, Man, and Cybernetics* 22 (6).
- Reinders, M and Bredeweg, B, 1992. “Reflective strategic control of multiple problem solving methods” In: *Proc. 10th European Conference on Artificial Intelligence (ECAI-92)* B Neumann (ed) 577–581, John Wiley.
- Rissland, EL and Skalak, D, 1989. “Combining case-based reasoning and rule-based reasoning: A heuristic approach” In: *Proc. IJCAI-89 Vol 1* 20–25.
- Rissland, EL and Skalak, D, 1991. “CABARET: Rule interpretation in a hybrid architecture” *International Journal of Man-Machine Studies* 34 (6) 839–887.
- Schreiber, G, Bartsch-Spörl, B, Bredeweg, B, Van Harmelen, F, Karbach, W, Reinders, M, Vinkhuyzen, E and Voß, A, 1991. “Designing architectures for knowledge-level reflection” *ESPRIT Basic Research Action P3178 REFLECT, Deliverable IR.4 RFL/UvA/III.1/4, REFLECT Consortium* August 1991.
- Skalak, D and Rissland, EL, 1992. “Arguments and cases: An inevitable twining” *Artificial Intelligence and Law* 1 (1) 3–48.
- Ten Teije, A and Van Harmelen, F, 1995. “Using reflection techniques for flexible problem solving” In: *Proc. IJCAI-95 Workshop on Reflection and Metalevel Architectures and their Applications in AI* Montreal.
- Van Harmelen, F, Wieland, B, Bredeweg, B, Schreiber, G, Karbach, W, Reinders, M, Voß, A, Akkermans, H, Bartsch-Spörl, B and Vinkhuyzen, E, 1992. “Knowledge-level reflection” In: B Le Pape and L Steels (eds) *Enhancing the Knowledge Engineering Process – Contributions from ESPRIT* 175–204, Elsevier.
- Van Harmelen, F, 1992. “Definable naming relations in meta-level systems” In: *Proc. Third Workshop on Meta-programming in Logic (META-92)* A Pettorossi (ed) *Lecture Notes in Computer Science* 649 89–104, Springer-Verlag.
- Van Heijst, G, Post, W and Schreiber, AT, 1994. “Knowledge based integration of representation formalisms” In: *Proc. 11th European Conference on Artificial Intelligence (ECAI94)* A Cohn (ed) 319–329, John Wiley.
- Voß, A, Karbach, W, Coulon, C-H, Drouven, U and Bartsch-Spörl, B, 1992. “Generic specialists in competent behaviour” In: *Proc. 10th European Conference on Artificial Intelligence (ECAI92)* B Neumann (ed) John Wiley.
- Zeleznirow, J, Hunter, D and Vossos, G, 1993. “Integrating rule-based and case based reasoning with information retrieval: The IKBALS project” In: *Proc. First European Workshop, Topics in Case-Based Reasoning (EWCBR-93)* University of Kaiserslautern, Germany, 341–346.
- Zeleznirow, J, Vossos, G and Hunter, D, 1994. “The IKBALS project: Multimodel reasoning in legal knowledge based systems” *Artificial Intelligence and Law* 2 (3) 169–203.
- Zeleznirow, J and Hunter, D, 1993. “Designing intelligent litigation support tools – the IKBALS perspective” *Law, Computers and Artificial Intelligence* 2 (1) 77–93.