

An Introspective Component-Based Approach for Meta-Level Reasoning in Clinical Decision Support Systems

Tor Gunnar Houeland, Agnar Aamodt
Department of Computer and Information Science,
Norwegian University of Science and Technology,
NO-7491 Trondheim, Norway
{houeland,agnar}@idi.ntnu.no

Abstract

The paper presents core elements of a meta-level architecture for clinical decision support, in the domain of palliative care. The goal of the reported research is to develop an architecture and an integrated set of methods for an introspective meta-level reasoner. Within the architecture a system is under development that addresses the identification and utilization of clinical guidelines for the assessment and treatment of cancer pain. Case-based reasoning is a core component of the architecture, which also incorporates rule-based and probabilistic model-based methods. The paper presents the overall architectural constraints and exemplifies parts of it through structured component descriptions.

1 Introduction

A clinical decision support system that covers several clinical tasks, such as patient examination, disease hypotetization, diagnosis determination, treatment planning, and drug administration, would typically need to combine several types of knowledge and several reasoning methods to provide good advisory support. There has recently been a renewed interest in meta-level reasoning in which a computer reasons about its own reasoning processes as well as the problem at hand [1].

The idea is that this allows the system to improve its own reasoning processes, for example by determining why a problem was solved incorrectly. By identifying where the error occurred in the reasoning process, the faulty part can be amended which would let the system solve such problems correctly in the future. In order to enable the meta-level reasoner to improve its own performance, learning within the meta-level reasoner itself, i.e. introspective learning, is also called for.

The focus of the research presented here is on meta-level reasoning for clinical decision support. A component-based architecture and an integrated set of methods for a meta-level reasoner to improve its reasoning and learning abilities are currently being developed. The architecture allows for the integration of all three reasoning paradigms in symbolic AI, i.e. rule-based, case-based, and (deeper) model-based reasoning.

This paper was presented at the NIK-2009 conference; see <http://www.nik.no/>.

We are addressing this problem in the domain of clinical decision support for palliative care. In a cooperation with the Palliative Medicine Unit, Cancer Department, St. Olavs University Hospital in Trondheim, we are studying the potential for proactive, advice-giving systems for the improved treatment of pain in cancer patients. In our current project, short-named TLCPC, which has national funding, the focus is on lung cancer. However, this research is tightly linked to a larger EU project called EPCRC [2], which covers all forms of cancer pain as well as other problems related to palliative care for long-term cancer patients.

A motivation for that project is also to standardize procedures and to unify clinical practice in palliative care [3], a goal for which computerized decision support systems have a strong potential. The system under development will in particular address the identification and utilization of clinical guidelines [4].

In the next section we review some of the related research in metareasoning with case-based components, and CBR used for guideline supported clinical decision-making. In section 3 our metareasoning approach and introspective architecture are introduced. Section 4 illustrates important semantic types used to describe components in the architecture, which is discussed in section 5. The status of the clinical guideline application is presented section 6. Concluding remarks end the paper.

2 Related Research

There is a significant amount of research that has addressed metareasoning in relation to case-based reasoning systems. In Meta-AQUA [5, 6] introspection is used in the retain phase to learn from mistakes. What is referred to as *introspectivemeta* – *XPs* are used to represent failures encountered while the system operates. The system constructs learning plans that consist of calling various learning algorithms.

Early accounts of meta-level architectures involving CBR also include BOLERO [7] and ANALOG [8]. In BOLERO, a case-based meta-level planner controls the execution of a rule-based reasoner designed to accomplish different medical diagnosis tasks. New plans are constructed during the problem solving process when needed, and captured as new cases by the meta-level learner. ANALOG provides a knowledge modeling framework and a system architecture that links various types of tasks, methods, and domain knowledge types. A selected method runs until an impasse situation is encountered, at which time a new method selection process is run. The meta-level learner remembers failed and successful instantiations of methods.

Christodoulou and Keravnou describe a meta-level architecture [9] in the domain of breast cancer histopathology. Each problem solver is associated with a task, an inference mechanism, and domain knowledge constraints. A set of meta-parameters is used by the metareasoner to characterize knowledge types (e.g. experience-based, causal), and desired solution properties (e.g. level of detail, accuracy, efficiency). The metareasoner is a case-based reasoner that captures and stores problem-solving paths as strategy cases incrementally.

In the ADAPtER system [10], an architecture that combines model-based and case-based reasoning for medical diagnosis, the CBR component is the primary object level reasoner. The model-based method is triggered either when the retrieval method fails to find a matching case, or the case adaptation module fails. Cases are captured as a compilation of the model-based process.

ROBBIE [11] is a case-based planning system in which an introspective model-based reasoner provides learning goals for the system when it fails to meet reasoning

performance expectations. This way of generating learning objectives based on reasoning failures is similar to Meta-AQUA, although it is performed in different ways.

jCOLIBRI [12] is a Java framework for building CBR systems with metareasoning capabilities. It uses a two-layer architecture that separates the user interface and core classes, and uses advanced features of the Java language and libraries to simplify development. jCOLIBRI uses a task structure with semantics defined in CBRonto, an ontology created for representing the terms and concepts that are important for CBR development.

The principles of evidence-based medicine, in which systematic research evaluation regimes are used to assess and justify results from medical research, form a well-established basis for medical practice [13]. One manifestation of this is the specification of clinical guidelines, i.e. operational procedures for how to conduct a particular type of patient examination, make a diagnosis, administer a type of drug, or perform other types of treatment.

Another source of information used by clinicians in their daily practice is, of course, the set of experiences a clinician has from earlier patients. While guidelines are important in unifying high-quality practice at a general level, past patient cases provide another level of specificity, closer to concrete actions. Hence, a combination of general guidelines with past experience cases is potentially a strong combination. Identifying the strengths and weaknesses of a case-based method vs. a generalization-based one can be done analytically or experimentally.

Marling et. al. [14] reports on an experiment in the domain of nutritional menu planning, in which a hybrid system was developed by combining the strengths of separate rule-based and case-based reasoners. The hybrid system outperformed both separate systems.

In the CARE-PARTNER system [15] medical guidelines are realized in the form of problem solving pathways, implemented as a rule-based system combined with information retrieval, and with CBR as the main problem solving method. GLARE [16] is a general decision-support system for managing and utilizing clinical guidelines, in which guidelines are represented as a hierarchical structure of decision paths. Based on that system a CBR system was incorporated as additional knowledge to handle situations that are not covered by the guidelines, so-called non-compliances [17].

Even if several medical guideline systems combine different reasoning paradigms, there has been very little work on moving the combined reasoning up to the metareasoning level.

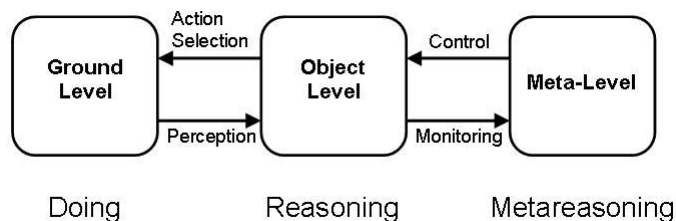


Figure 1: Duality in reasoning and acting (from Cox and Raja 2008 [1])

3 Metareasoning Approach

Recently, Cox and Raja [1] presented a general high-level framework of metareasoning, shown in figure 1. It relates three levels, i.e the ground level (physical perception and action), the object level (reasoning about action), and the meta-level (reasoning about reasoning).

Within this high-level perspective, Leake and Wilson [18] address the learning part of metareasoning, and present a set of challenging issues, such as learning for self-understanding and self-explanation. Many meta-level reasoning systems start by building a partial solution and then invoking meta-level reasoning functionality to determine whether the object-level reasoning processes are working satisfactorily or not. Such systems then either proceed as normal, or classify the reasoning process as a failure and create a new metareasoning goal to learn from this failure.

This is in contrast to the meta-level control agent in our approach, which operates on the object-level reasoning components directly without a specific reasoning failure to address. This allows the system to have a clearer broad focus on performing changes that affect the entire reasoning system instead of correcting single failures, and this broadening is also identified by Leake and Wilson [18] as an important opportunity for a more flexible learning focus.

Future planned meta-level components in our approach such as a competence-evaluator for problem-solving methods can also assist in providing the system with a level of self-understanding, which is another identified opportunity for introspective learning systems.

A system using our introspective architecture with CBR reasoning methods which follows this general framework is shown in figure 2. The ground level is represented as I/O for the system, which is not covered in our architecture but must necessarily exist in some form for a complete CBR system. In the shown system reasoning tasks are explicitly represented, with a “CBR-based problem solving task” at the object level and several metareasoning tasks.

In our architecture there is no fundamental difference in how reasoning tasks for the two levels must be represented, but they are shown as separate boxes in the figure to ease understanding and fit the metareasoning framework of Cox and Raja. The importance of our introspective architecture lies in the components, each of which implements a specific functionality, and is modified and assigned to a particular reasoning task by meta-level reasoning components.

Richter [19] introduced the knowledge container model, where pieces of knowledge can be categorized into four categories: case vocabulary, cases, similarity assessment and adaptation knowledge. Given that a set of precise and consistent terms is important for reasoning at the meta-level, we explicitly include the CBR system vocabulary as part of

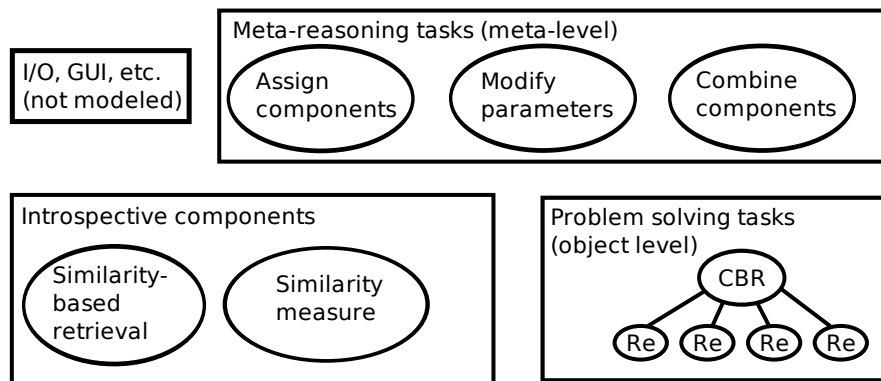


Figure 2: A CBR system based on the introspective architecture

the case vocabulary category.

To make sure that the vocabulary contains semantics that are useful for human designers, we suggest that the vocabulary should be created and updated manually, unlike the other knowledge sources which will be changed and influenced by the meta-level reasoner.

In our architecture the case knowledge is stored in cases, similarity assessment as part of the components providing methods for object-level retrieval and the adaptation knowledge is shared between object level reuse components and a more profound system-level adaptation in meta-level components.

For the meta-level control agent to be able to examine and calibrate the system's components, it's important that the settings available can be interpreted by the control program, and that the components expose interfaces at an appropriate level of abstraction. To facilitate this, we are developing a vocabulary of CBR-related terms and the intended semantics as part of our approach.

An important aspect of the vocabulary is that it describes what the terms mean at a semantic level without relying on the specific realizations in any particular CBR system implementation. An example of this is a case base, which is simply defined as a set of cases. While in practice many CBR systems store the cases as a form of ordered lists, their reasoning processes typically do not rely on the particular order the cases are listed in within computer memory or on disk.

The purpose of the vocabulary is to abstract away the particular specifics in implementations and generalize the terms to a semantic level where it describes the actual requirements for a term without imposing a needlessly specific design.

Our vocabulary is being developed to be conceptually compatible with CBR_{Onto} [20], a modeling framework that has already examined the meanings and relations between terms from an ontological perspective. As an added benefit the CBR_{Onto} ontology is compatible with the "4 REs" CBR process cycle [21] which is widely quoted and referred to in the CBR literature.

4 Vocabulary Examples

Of particular importance for our semantic task description is the type system for inputs and outputs. These types are meant to represent at the knowledge level the essential meaningful content that is to be processed. For system implementations the exact structures used to realize these knowledge types may differ, and often the same programming language structures may be used for several different knowledge types. E.g.

for a case-based reasoning system based on feature vectors, the stored cases, problem specifications, retrieval queries and produced solutions will typically all be implemented using the same type of programming language data structure, but they are still different on a semantic level.

Important semantic input and output base types.

Case a record of a problem solving experience, consisting of the encountered problem and the solution details

AttributeCase a subtype of Case where the problem descriptions are represented as a set of features

Feature a particular aspect of an AttributeCase, where each separate case can either lack the feature or have a corresponding value selected from the feature's set of possible values

ProblemInput A problem specification for a problem-solving method

ProblemSolution A solution to a problem

SolutionMethod A method used to create a problem solution

SolutionEvaluation An evaluation of a solution applied to a problem

Similarity a float used specifically to represent similarity

Derived semantic types.

Set $\langle T \rangle$ a collection of objects of type T

List $\langle T \rangle$ an ordered collection of objects of type T

(a, b, c, \dots) an n -tuple such as **(Case, Similarity)** consisting of n values where the first is of types a , the second of type b , etc.

$A \rightarrow B$ a function such as **Case** \rightarrow **Similarity** mapping inputs of type A to output of type B

Other terms used in the example figures.

SizeOf returns the number of objects in a *Set* or *List*

PC an identifier used in the examples to specify an unrestricted type for problem characterizations

CB an identifier used in the examples to specify a type adhering to the **Case** semantics for cases in a case base

Task	Input	Output
Case-based reasoning	ProblemInput	ProblemSolution
- Retrieve	ProblemInput	Set<CB>
- Problem characterization	ProblemInput	<i>PC</i>
- Case retrieval	<i>PC</i>	Set<CB>
- Focus	Set<CB>	Set<CB>
- Reuse	<i>(PC, Set<CB>)</i>	ProblemSolution
- Adapt solution method	<i>(PC, Set<CB>)</i>	SolutionMethod
- Adapt solution	<i>(PC, SolutionMethod, Set<CB>)</i>	ProblemSolution
- Revise	ProblemSolution	(PS, SE)
- Evaluate solution	ProblemSolution	SolutionEvaluation
- Repair solution	(PS, SE)	ProblemSolution
- Retain	(PS, SE)	
- Update general knowledge	(PS, SE)	
- Add to case base	(PS, SE)	

(PS, SE) is an abbreviation for **(ProblemSolution, SolutionEvaluation)** due to space concerns.

Figure 3: Inputs and output for the CBR tasks

5 Introspective Architecture

In our conceptual framework we consider the traditional core case-based reasoning process as one possible problem solving method at the object level. We consider this to be one limited part of the combined reasoning system, with specific responsibilities, and there can be potentially many other problem solving methods implemented in the same architecture.

A high-level task decomposition of the CBR process for our architecture is shown in figure 3, where the task “Case-based reasoning” is split into the 4 RE subtasks, and each of these are further split into smaller subtasks. Any components that are assigned to parts of this process must match the semantic input and output types for the corresponding tasks, which are also included in the figure.

To elaborate on the case-based retrieval task, it starts from a **ProblemInput** and retrieves a set of cases **Set<CB>** where *CB* is a type adhering to the **Case** semantics, and is further subdivided into subtasks that further specify how this is performed. It starts with a subtask identifying the important aspects of the problem and characterizing it as a query, transforming the **ProblemInput** into an intermediate form *PC*.

This intermediate form is not restricted by the architecture, and the only requirement is that the methods performing the subsequent CBR subtasks can accept the characterization form *PC* as input. This characterization is used to retrieve a number of previous cases from the case base, and then this is further narrowed to just focus on the most relevant information by e.g. filtering out a subset of cases or generalizing cases. The other subtasks are formalized in a similar way and correspond to the well-known steps that are often used to describe CBR systems.

In our approach, the metareasoning components exist separately from the object-level CBR reasoning components, and in fact influence and controls how the CBR problem solving method is performed, which corresponds to the aforementioned metareasoning cycle [1]. By evaluating how the CBR method performs while solving actual new problem

instances, the meta-level control agent can identify the strengths and weaknesses of the current system and attempt to use this to improve the system's competence or use alternative reasoning methods. For our meta-level component-combiner this is achieved by attempting to re-solve problems using different assignments of methods for each of the tasks and subtasks in the architecture. Whether the newly combined reasoning process is an improvement is then evaluated based on whether the solutions produced for individual problem queries are correct for more problem instances than before.

Architectural components

One of the most important features of this architecture is that each component contains extra structures that semantically describe the component semantically using our CBR vocabulary. It is on the basis of this added information that the meta-level control component can automatically assign components to perform the system's reasoning tasks. Figure 4 shows such a self-describing semantic structure for an example object-level retrieval component.

Each component contains a list of types that apply for the component, which is listed first in the figures. This can either just be a single *name* to indicate that any type is supported, or a statement of the form "*name isa supertype*", which indicates that the *name* type must support the same operations as *supertype*. This is useful when a component performs a generic operation that can apply to many different types of input, and e.g. only requires that the input and output types are the same or that two inputs are comparable. This section is empty for simpler components that only refer directly to specific types in the vocabulary, such as the illustrated example retrieval component.

After that the input and output variables are listed. Each variable consists of a line of the format "*name: type*", where *name* is an identifier that is used to refer to the input or output throughout the component specification and *type* is the semantic type, which can either be a specific from the vocabulary or one of the types specified in the component's Types section. An example of this is the line "*query: AttributeCase*" from the example component, which means that the component receives an input of type **AttributeCase** which is referred to as *query* in the component description.

The following section lists Conditions that have to be fulfilled for the component to produce the expected results. As long as the input variables conform to the specified conditions, the output variables are guaranteed to follow the specifications in the Guarantees section. While the type specifications are semantic restrictions on what kind of operation the component can perform, the conditions specify for which values of inputs the component will actually behave as intended, and these conditions are not necessarily checked by the component.

Because of this the component's operation can usually be performed for non-conforming inputs, but this can produce undefined results and should be avoided. By listing the conditions in the self-describing structure, the metareasoning component can make sure that only compatible components are combined, or that the conditions are checked on-demand before the operation is performed where this cannot be guaranteed.

The final section is a short semantic description of the core functionality performed by the component. This has the format "*x based on y*", where *x* and *y* are statements composed using the input and output variables as well as a set of pre-specified terms representing important concepts related to the reasoning system and the application domain.

Based on these two examples, the meta-level control agent can create a new

Types:
Input: <i>query</i> : AttributeCase <i>casebase</i> : Set<AttributeCase> <i>similarityfunction</i> : (AttributeCase source, AttributeCase target) → Similarity
Output: <i>ordered</i> : List<AttributeCase>
Conditions: $5 \leq \mathbf{SizeOf}(casebase) < 100000$ $0 \leq similarityfunction(x, y) \leq 1$
Guarantees: SizeOf(ordered) = 5
Approach: <i>order(casebase)</i> based on <i>query</i>

A component that assigns an order to the *casebase* based on similarity to the *query* using the specified *similarityfunction*.

Types: <i>cCase</i> isa AttributeCase
Input: <i>source</i> : <i>cCase</i> <i>target</i> : <i>cCase</i>
Output: <i>similarity</i> : Similarity
Conditions:
Guarantee: $0 < similarity \leq 1$
Approach: compute similarity <i>similarity</i> based on <i>source</i> features, <i>target</i> features

A component for computing similarity based on local feature similarities between *source* and *target*.

Types:
Input: <i>query</i> : AttributeCase <i>casebase</i> : Set< AttributeCase >
Output: <i>ordered</i> : List< AttributeCase >
Conditions: $5 \leq \mathbf{SizeOf}(casebase) < 100000$
Guarantee: SizeOf(ordered) = 5
Approach: <i>order casebase</i> based on <i>query</i> similarity

A combined component that assigns an order to the *casebase* based on feature weighted similarity to the *query*.

Figure 4: Example components for object-level reasoning tasks.

component that accepts a query and case casebase of type **AttributeCases** and a feature weight map and returns a ranked list of matching cases. This is done by substituting **AttributeCase** for *cbType* and *cCase* (which fulfills the listed type requirements) and using the feature similarity measure as the *similarityfunction* (by identifying that the output guarantee $0 < similarity \leq 1$ fulfills the $0 \leq similarityfunction(x, y) \leq 1$ condition).

The meta-level learning component can also further refine this into a component that learns the feature weight map automatically while it is being used. This can be done by matching it with an appropriate learning method that e.g. takes a casebase (**Set**< **AttributeCase** >) as input and produces a similarity importance value for each feature among cases in the casebase.

6 Palliative Care Application

Although there has been a lot of focus on developing clinical guidelines within the medical community, and several guideline systems have been developed by medical professional organizations, there is not a consensus as to what is a good guideline system. Further, the active use of guidelines in a clinical setting is far from the level desired, both from the perspective of quality of treatment and the perspective of unified treatment across hospitals and countries.

In our partner project EPCRC, being a collaboration involving many highly influential medical groups in across Europe, the aim is to reach a consensus on a set of high-quality and operational guidelines that will be used in practice. While awaiting the results from EPCRC, we currently work with an existing set of guidelines defined by NCCN (National Comprehensive Cancer Network) in the US. An ontology is currently being built based on a combination of generic UMLS terms combined with terms from the SNOMED and NCI (National Cancer Institute in the US) ontologies.

The top-down design process is combined with bottom-up experimental system building, starting from simple system components that will be combined. Currently a simple a rule-based reasoner is being implemented at the object level. Example guidelines link patient data related to pain level, pain history, and history of treatment, to the next treatment. Type of treatment considered is the administration of different types of analgesics, with opioids as the largest subclass. Based on the results from the initial model, the system will be extended with case-based and model-based components. The model-based component will a be Bayesian network for reasoning about causality under uncertainty.

Acquiring the necessary medical knowledge needed for our experiments is a continuing process. To advance the method development we are developing a toy example system in the domain of advice-giving for film selection, in parallel to the more complex clinical guidelines system. In that system CBR methods are currently focused on both the meta and object levels. Cases, problems and solutions are currently represented as feature vectors, and there are no explicitly represented general knowledge structures outside of the CBR reasoning components. To predict a movie rating for a user, a retrieval component retrieves a number of similar other users that have seen the movie, where similarity is determined as the average difference in ratings for movies both users have rated. A simple reuse component then copies the majority rating among the retrieved cases. The system has a meta-level control agent that adheres to the principles of the metareasoning approach described earlier. Component combinations are tried out based on matching the input and output type descriptions. Although simple and different

from the clinical application, the system assists in the bottom-up specification of the introspective architecture by providing a test-bed for experiments.

In a clinical guideline support system past cases may be utilized in several ways. The role we have intended for the cases in our guideline system is two-fold. Having arrived at a leaf node in the guideline structure, CBR will be used to continue from there by providing a more specific and detailed advice, based on adapting a past result. On the other hand, if the guideline system cannot provide reasonable advice, CBR is triggered as a complementary method. The latter approach is similar to the non-compliance method [17] referred to earlier.

7 Conclusions

In this paper we have presented an introspective approach to meta-level learning and outlined a component-based architecture for designing reasoning systems which supports our introspective methods. The main contribution of our approach is the way our architecture allows for gradual additions of metareasoning methods that focus on broad, system-wide improvements.

We are working on further developing this architecture, and adding new components directed towards both object-level and meta-level reasoning methods for a clinical decision support system based on medical treatment guidelines.

Acknowledgements

This research is supported by Norwegian Research Council (NFR) grant, Contract no 183362, Translational Research in Lung Cancer and Palliative Care (TLCPC), together with funds from the Norwegian University of Science and Technology (NTNU). Thanks to Tore Bruland and Helge Langseth who contributed on the AI method side through a series of discussions, Sunil Raja who is our main contact on the medical side and has provided insight into clinical pain assessment and treatment, and Stein Kaasa who is the initiator and project leader of both the TLCPC and EPCRC projects.

References

- [1] Cox, M.T., Raja, A.: Metareasoning: A manifesto. Technical report, BBN TM-2028, BBN Technologies (2007)
- [2] Haugen, D.F., Kaasa, S.: The EPCRC. *Eur J Palliat Care* (2007; 14(3):130)
- [3] Kaasa, S.: Palliative care research – time to intensify international collaboration. *Palliat Med* (2008; 22:301-2.)
- [4] Quaseem, A.e.: Evidence-Based Interventions to Improve the Palliative Care of Pain, Dyspnea, and Depression at the End of Life: A Clinical Practice Guideline from the American College of Physicians. *Annals of. Internal Medicine*, vol. 148, no. 2 (2008:141-146)
- [5] Cox, M.T., Eiselt, K., Kolodner, J., Nersessian, N., Recker, M., Simon, T.: Introspective multistrategy learning: On the construction of learning strategies. *Artificial Intelligence* **112** (1999) 1–55

- [6] Cox, M.T.: Multistrategy learning with introspective meta-explanations. In: *Machine Learning: Proceedings of the Ninth International Conference*, Morgan Kaufmann (1992) 123–128
- [7] Lopez, B., Plaza, E.: Case-based planning for medical diagnosis. In Ras., Z. (ed.), *Proceedings of ISMIS-93*. LNAI 837, Springer, 1993: 96-105
- [8] Arcos, J., Plaza, E.: A reflective architecture for integrated memory-based learning and reasoning. In Weiss, S. et. al. (eds.): LNAI 873 , Springer, 1994: 289-300
- [9] Christodoulou, E., Keravnou, E.: Metareasoning and meta-level learning in a hybrid knowledge-based architecture. *Artificial Intelligence in Medicine* 14 (1998): 53-81
- [10] Torasso, P.: Multiple representations and multi-modal reasoning in medical diagnostic systems. *Artificial Intelligence in Medicine*, 23 (2001): 49-69.
- [11] Fox, S., Leake, D.B.: Combining case-based planning and introspective reasoning. In: *Proc. of the 6th Midwest Artificial Intelligence and Cognitive Science Society Conference*. (1995) 95–03
- [12] Recio-García, J.A.e.a.: Ontology based CBR with jCOLIBRI. In Ellis, R., Allen, T., Tuson, A., eds.: *Applications and Innovations in Intelligent Systems XIV*. *Proceedings of AI-2006*, Springer (December 2006) 149–162
- [13] Friedland, D.: *Evidence-based medicine: A framework for clinical practice*. (1998)
- [14] Marling, C., Petot, G., Sterling, L.: Integrating case-based and rule-based reasoning to meet multiple design constraints. *Comp. Intell.*, 15 (3), 1999, 308-332
- [15] Bichindaritz, I., Kansu, E., Sullivan, K.M.: *Case-Based Reasoning in CARE-PARTNER: Gathering Evidence for Evidence-Based Medical Practice*. EWCBR'98, LNAI 1488, pp. 334-345 (1998)
- [16] Terenziani, P., Molino, G., Torchio, M.: A modular approach for representing and executing clinical guidelines. *Artificial Intelligence in Medicine*, 23(3):249–276 (2001)
- [17] S. Montani, S.: Case-based reasoning for managing non-compliance with clinical guidelines. ICCBR 2007, 5th Workshop on CBR in the Health Sciences (2007)
- [18] Leake, D., Wilson, M.: Extending introspective learning from self-models. *Proceedings of the AAAI 2008 Workshop on Metareasoning: Thinking About Thinking* (2008)
- [19] Richter, M.M.: The knowledge contained in similarity measures. Invited Talk at ICCBR-95 (1995)
- [20] Díaz-Agudo, B., González-Calero, P.A.: CBROnto: A Task/Method Ontology for CBR. In: *Procs. of the 15th International FLAIRS'02 Conference*, AAAI Press (2002) 101–105
- [21] Aamodt, A., Plaza, E.: Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI Communications* 7(1) (March 1994) 39–59