Explaining Problem Solver Answers

Vladia Pinheiro[1] Vasco Furtado[1] Paulo Pinheiro da Silva[2] Deborah L. McGuinness[2]
[1] Universidade de Fortaleza, Fortaleza, CE, Brazil
vladiacelia@terra.com.br, vasco@unifor.br
[2] Knowledge Systems Laboratory, Stanford University, Stanford, CA, USA
{pp,dlm}@ksl.stanford.edu

Abstract

Knowledge based systems (KBSs) should explain their answers if their users are expected to understand and thus trust the answers. Problem solvers, KBSs implementing problem solving methods (PSMs), should also explain their answers. Few KBS systems, however, are effective at explaining their answers either because they cannot systematically generate explanations or, when they can, their explanation components cannot easily be extended to new kinds of tasks. In this paper we present an approach enabling problem solvers to explain their answers in a systematic way. To generate proofs for their answers, the approach relies on the fact that problem solvers can retrieve and reuse their PSMs. To generate explanations automatically the approach relies on the Inference Web infrastructure. The approach is implemented for a deployed problem solver tool using explanations to train police teams to perform resource allocation for public safety.

1 Introduction

Problem-solving methods (PSM) are patterns describing how to use reasoning steps to solve knowledge-intensive tasks such as planning, scheduling, diagnosis, design and assessment. PSMs are expected to contain reasoning step descriptions generic enough to solve tasks independently of application domains (Fensel & Benjamins, 1998). In the context of this paper, problem solver systems are knowledge-based systems (KBSs) using PSMs to solve tasks. Thus, if problem solver users, software agents and humans, need to understand solver answers then users may need to understand the PSM reasoning processes to understand and thus believe the answers. Some existing KBSs are capable of explaining their answers (e.g., Buchaman & Shortliffe, 1984; Clancey, 1986; Swartout, 1983; Chandrasekaran, 1986; Richards, 2000; Shankar & Musen, 1998). Few systems, however, are able to provide explanations for answers from their PSM implementations. Some KBSs can generate explanations based on generic tasks or meta-rules but we are not aware of any that provide a systematic way to generate explanations of PSM implementations.

In the work described in this paper, we are using PSMs to support the development of KBSs by using the Unified Problem-Solving Method Description Language (UPML) (Fensel *et al.*, 2003). UPML-based KBS development makes KBS reasoning processes explicit by implementing PSMs as part of the applications. Thus, the access to PSMs at explanation-time can be used by UPML-based KBSs to explain their answers. In fact, by using UPML patterns, one can enhance the quality of explanations abstracting away task-specific reasoning steps from proofs and keeping relevant information for understanding answers.

The definition of an automatic approach for explaining KBS executions both at the strategic level and at the domain level is a major goal in this work. The explanation strategy described in this paper is based on Inference Web (IW), an infrastructure for Web explanations enabling applications to generate portable and distributed explanations for any of their answers (McGuinness & Pinheiro da Silva, 2004). This paper introduces some extensions to the Inference Web infrastructure enabling IW tools to explain answers from UPML-based KBSs. The new UPML explanation component can generate proofs for KBS answers in the Proof Markup Language (PML) format (Pinheiro da Silva, McGuinness & Fikes, 2004). Thus, proof fragments in PML can be shared with other applications, besides using the IW's infrastructure to abstract proofs into explanations and to present proofs and explanations to users.

The rest of the paper is organized as follows. In Section 2, we discuss various approaches for KBS explanations. In Section 3, we revisit the UPML framework and introduce the UPML explanation component for the Inference Web responsible for generating PML documents. In Section 4 we describe how IW uses PML documents to explain PSM answers. In Section 5, we describe how our approach was used to explain a public safety problem solver decisions. Finally, in Section 6, we summarize our current contributions and discuss future work.

2 Explaining Knowledge Based System Answers

Moore (1995) defines some criteria to be observed in explanation systems including *extensiveness*, which indicates that explanation facilities must be easily extendable in order to cope with new demands and information needs. To meet the

extensiveness criterion, we claim that a KBS explanation framework must be able to explain most kinds of knowledge-intensive tasks whether that involves the KBS use of many PSMs or a single PSM.

Explanations for KBS have appeared as a significant and independent topic of study since MYCIN (Buchaman & Shortliffe, 1984) was developed. For explaining a diagnosis, MYCIN follows the trace of rules answering "why" and "how" questions. In the project GUIDON (Clancey, 1986) identified that most of the knowledge necessary to explain the behavior of the system, as for example, the reasoning strategy for diagnosis, was not explicitly represented in the knowledge base. NEOMYCIN (Clancey, 1986) contributed to explanation research in KBSs by using an explicit representation for problem resolution strategies and by using meta-rules in explanation planning. By using meta-rules, NEOMYCIN separated the domain ontology from MYCIN's rules. This separation allowed NEOMYCIN to be more usable as an explanation infrastructure; however it was specific to MYCIN in terms of problem solving and domain representation.

XPLAIN (Swartout, 1983) represented general problem-solving knowledge using explicit models to produce a general description of the system's reasoning. Also, Chandrasekaran (1986) proposed KBS explanations through the use of Generic Tasks. There, each subtask contains knowledge for explaining its execution that contributes to the entire system explanation. In both XPLAIN and Chandrasekaran's approaches for explanation, the extensiveness criterion is satisfied although the knowledge for explaining a reasoning strategy is separated from the KBS knowledge base.

The use of Ripple Down Rules (RDRs) (Richards,2000) presents a new paradigm for KBS explanation providing users with enough information about the circumstances in which a rule was applied and the relationship of the rule with others. RDRs provide a structure that allows explanations of why an answer was obtained and which values could be modified to reach another answer. RDR is a technique for acquisition and representation of knowledge that uses a case-based reasoning approach, providing contextualized knowledge. RDR provides only cases as representation for explanation which is a domain-specific representation. RDR's tools, like browsers, line diagrams, concept matrix are specific for this knowledge representation technique and this approach was applied only in classification tasks. Therefore, it is not trivial to extend the RDR approach to other knowledge-intensive tasks such as design.

WOZ (Shankar & Musen, 1998) is a framework for explaining component-based decision-support systems. The framework is composed of: (i) a functional component responsible for the reasoning process (problem-solving) and data recovery; (ii) cooperative visualization agents associated with each functional component and responsible for explanation presentation and user interaction through graphical user interfaces (GUI). Agents interact with other agents and components to provide adequate explanations according to the user characteristics gathered by the agents; (iii) a system program called Director who intermediates interactions among agents in accordance with the Explanation Strategy; (iv) application domain models such as user model, agent model and explanation strategy. WOZ incorporates some of the major trends in software engineering including explicit models, multi-agent architectures, and visualizations. Although WOZ meets the extensiveness criterion its explanation strategy may not be scalable since a new situation-action mapping specification must be developed for each application, and building and maintenance can be laborious.

3 The UPML Explanation Component

Our approach to explain KBS answers integrates UPML and Inference Web. The approach benefits from UPML advances in knowledge modeling as well as the semantic web's OWL language (W3C, 2004). Our use of the Proof Markup Language for dumping proofs supports the interoperability of distributed applications sharing proofs and provides explanations about the reasoning process embedded in PSMs.

Our approach uses but also favors reuse of components because we are proposing generic components for explanation, which are reused by UPML PSMs. Thus, UPML-based KBSs can deploy proofs in PML as justifications for their answers.

3.1 The UPML framework

The UPML framework supports KBS modeling from reusable components, adapters, development guidelines, a description language and tools. The UPML architecture describes the different software components of a KBS:

- a *task* defining the problem that should be solved by the KBS;
- a *problem-solving method* defining KBS reasoning processes;
- a *domain model* describing the KBS domain knowledge;
- an *ontology* providing the terminology used in the other components;
- a *bridge* for each distinct pair of KBS components modeling relationships between components;
- a *refiner* specializing a KBS component.

KBS development effort is reduced through reuse. Moreover, the growing library of generic UPML components eases the development of KBSs for other tasks. For example, Pinheiro, Furtado & Furtado (2004) describe a set of UPML generic components implemented in Java including the *abstract-and-match* PSM used for assessment tasks as described in (Schreiber et al., 2000). Basically, the *AbstractMatch* java class implements the PSM reasoning process through a control structure that is responsible for sequencing the subtasks. The class extends the *PSMComponent* class containing generic methods for executing the

Domain-PSM and Task-PSM mapping. These mappings are responsible for the knowledge role communication among UPML components, and also for the execution of subtasks related to the PSM. The knowledge roles defined for the abstract-and-match PSM are the following: *case description*, *criteria to be evaluated*, *abstraction rules*, *evaluation rules* and *decision rules*. The KBS developer needs to define these domain-specific knowledge roles and to use the API generic method to define the mappings. The development of a KBS for assessment tasks using the UPML components in any domain can thus reuse the PSM implementation leaving developers with the task of implementing domain-specific classes.

The ExpertCop System (Furtado & Vasconcelos, 2004) is a UPML-based KBS example performing an assessment task implemented by the abstract-and-match PSM. In ExpertCop's decision-making process, a criminal agent must evaluate data gathered from a geo-simulated environment using a set of criteria to decide whether it should commit a crime. This simulation process is used to train police about when and where crimes are likely to be committed. The task reasoning process defines a control structure executing the following subtasks sequentially: (i) *abstract* that simplifies the case data; (ii) *specify* that finds criteria relevant to the case data; (iii) *select* that selects one criterion for evaluation; (iv) *evaluate* that evaluates the select criterion with respect to the case data; (v) *match* that checks whether the criteria that were evaluated lead to a decision. The select, evaluate and match subtasks are interactively executed for each criterion until a decision can be found or the criteria set is exhausted.

Risk level, *opportunity* and *crime level* are examples of criteria evaluated by criminal agents. For example, the risk level criterion can be evaluated against *police officer proximity*, *population density*, *place safety rate*, and *public illumination* data. In turn, the police officer proximity w.r.t. criminals is encoded using the following discrete values: "close", "medium" and "far". Discrete values are used since this matches criminal evaluations more closely than continuous values. The abstract subtask uses the following heuristic: "the proximity of the police officer is 'close' if the value is below 300 meters". For a 280 meters value, the abstract subtask simplifies the police officer proximity to "close". The specify subtask finds the following criteria for the case: risk level, opportunity, crime level. The select subtask selects criteria for the evaluate subtask, one criterion a time. After each evaluation, the match subtask checks if the criteria value leads to a decision.

3.2 A UPML Component for the Inference Web

In this section we introduce an UPML Explanation component responsible for the generation of PSM proofs in PML. With the help of IW tools and PML documents it is possible to explain PSM reasoning processes. The explanation component interacts with the PSM and Task Model components but not with the Domain Model. Thus, explanations are not domain-specific and they can be reused for other PSMs.

The Explanation Component contains the following classes:

- The *ProofGeneration* class used for building proof steps from parameters received from UPML's PSM and Task and for using the parameters as variable bindings in answer's proof tree. This component is also used for encoding the current state of some PSM variables as sentences in Knowledge Interchange Format (KIF).
- The *Proof* class represents a step in a proof. Each step is identified by a conclusion and a set of antecedents. A proof step conclusion can be either derived or told. If the conclusion is derived then the class keeps information about the inference rule used to derive the conclusion, e.g., generalized modus ponens (GMP), and information about the step premises. If the conclusion is asserted directly from a source, e.g. it is stated in an ontology, then the class keeps information about the source. To identify the inference engine generating a proof step, we have created a relationship between this class and its subclasses such as *PSMProof*, *JEOPSProof*, *JESSProof*, etc. In a single proof tree generated by the ProofGeneration class, it is possible to represent inference steps generated by multiple inference engines such as JEOPS (Figueira Filho & Ramalho, 2000) or JESS (JESS, 2005), and inference steps generated by the PSM control structure itself that may be considered as a meta-annotation for the proofs.
- For each proof tree in ProofGeneration, the Handler class is responsible for mapping proof trees into node sets and for generating PML documents.

4 Using the UPML Explanation Component

A KBS can access its PSM and Task components from the UPML component library. The PSM component implements the control structure responsible for the coordination of subtasks within tasks. For each subtask execution, the *PSMComponent* class responsible for calling subtasks invokes the explanation component methods that encode subtask queries into PML queries. Each subtask implements a specific functionality inside the reasoning process, either by means of a rule execution or via an algorithm. If an inference engine processes a rule, the engine encodes the rule applications as a proof steps in PML. Otherwise, the subtask implementation is that should call the Explanation component methods for generating proof steps in PML. The conclusions of PML node sets directly associated with queries correspond to query answers. In accordance with the task execution sequence within PSMs, several pairs of query-and-answer-sets may be generated until final conclusions for the PSM are reached. By using the explanation component, the problem solver encodes the queries, answers, and answer proofs in PML documents. Furthermore, the problem solver embeds the PSM control structure in the sequence of queries. In fact, proof trees extracted from PML

node sets represent the order of the execution of the PSM subtasks and the inference steps on rules in the domain knowledge base. The PML encoding of queries and answers are generated from generic UPML components and inference engines. Therefore, the UPML explanation component as presented here can be reused in any UPML-based KBS

Figure 1 shows an abstract and concrete level view of an example as well as a detailed view [1]of the IWBrowser while presenting PML documents from the ExpertCop System. At the abstract level, the browser shows the PSM reasoning process representing each abstract-and-match PSM reasoning steps as an oval. The diagram describes the PSM control structure as follows: the abstract, specify, select, evaluate and match subtasks are executed in this order, but the select, evaluate and match subtasks may be executed iteratively until a decision is made or all the criteria are evaluated.

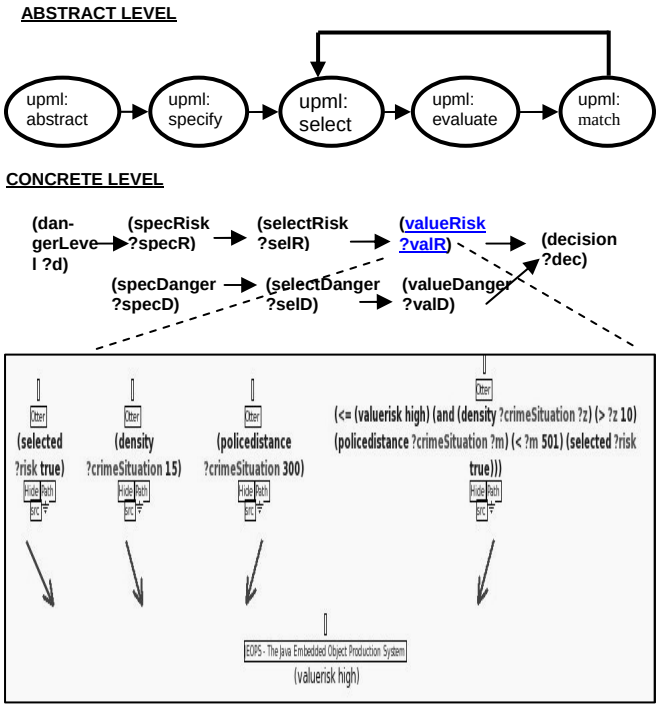

Figure 1. An example of explanation diagram from abstract-and-match PSM

.

At the concrete level, the diagram shows additional details about the PSM reasoning process. One or more queries are executed in support of PSM subtasks and they are presented along with the reasoning steps allowing the browser to show the queries corresponding to each subtask. Further, by selecting a query, a user can ask the browser to show query answers and their proofs thus exposing how proof steps are performed against facts in the underlying knowledge base. For example, the Figure 1 concrete level shows a situation where two criteria are evaluated - *risk* and *danger*. The explanation component generates two queries to the evaluate subtask: (valueRisk ?valR) asking "what is the value for the Risk criterion?", and (valueDanger ?valD) asking "what is the value for the Danger criterion?". The evaluate subtask then tries to apply an evaluation rule to facts in the KBS knowledge base to answer the query. By selecting the query (valueRisk ?valR), the browser presents a justification for the fact that valueRisk is high rendered as "(valueRisk high)". The fact is encoded in a PML node set conclusion and each inference step in the node set corresponds to a fact justification. When rendering the fact justification, the browser shows that the answer, through an application of the GMP inference rule, was inferred from the following facts:

- (policedistance ?crimeSituation 300)
- (density ?crimeSituation 15)
- (selected ?risk true)

and the axiom

- (<= (valueRisk high)
  (and (density crimeSituation ?z)
  (> ?z 10)
  (policeDistance ?crimeSituation ?m)
  (< ?m 501)
  (selected ?risk true)))

The facts and axiom above are in KIF. The axiom represents an evaluation rule in the system and the facts may be told or derived. The ExpertCop concludes that valueRisk is high.

5 Explaining ExpertCop Answers

5.1 ExpertCop Domain Model

The UPML Framework with the new explanation component has been used in the ExpertCop System project to develop a decision-making process KBS for a criminal agent. The system is based on a criminal geo-simulation system modeled for a specific region. This system uses the Multi-Agent Systems (MAS) technology to simulate social environments. In the project, the KBS developers reused UPML generic components, the explanation component, and the classes implementing the abstract-and-match PSM. Developers implemented the following domain classes:

- *CrimeSituation* representing data about the environment at the moment of the assessment of the criteria.
- *CriterionCrime* representing criteria to be evaluated such as risk level, opportunity, crime level, criminal danger, etc.
- *DecisionCrime* representing the following possible decisions: "the criminal does commit the crime" and "the criminal does not commit the crime".
- *AbstractionRules* representing rules for abstracting case data. For example, to abstract the distance between a police officer and a selected point in the geo-simulated system as 'close', ExpertCop uses the fol-

[1] Note that other views of the explanation are possible including abstractions, natural language presentations, etc.

lowing abstraction rule: “if distance < 300 meters then police officer distance is ‘close’”.

- *EvaluationRules* representing rules for evaluating available criteria. For example, ExpertCop has the following evaluation rule for the risk level criterion: “if police officer distance is close then risk of committing a crime is high”.
- *DecisionRules* representing rules for making a decision for the case. For example, ExpertCop uses the following decision rule: “if risk is high then decision is to not commit a crime”.

These classes represent the domain knowledge and are instantiated in the application and mapped to the PSM ontology. The domain knowledge gathered from a knowledge acquisition process with expert police personnel who know which criteria a typical criminal uses when deciding whether to commit a crime.

5.2 ExpertCop Explanations

Through the reuse of UPML components, the KBS implements an executable criminal decision-maker. Users have the option of asking for explanations of the decisions. Explanations are enabled because the decision process is encoded in PML documents. These documents contain information concerning the queries and answers along with the reasoning steps, domain criteria, and domain rules used in the decision process.

Explanations expose the reasoning process used by the criminal when deciding whether to commit the crime. ExpertCop uses this input to induce a critical analysis and provides recommendations for increasing or decreasing the levels of police surveillance and placement aimed at improving police effectiveness.

ExpertCop can explain why crimes did or did not happen. It can also answer follow-up questions such as “how was the Risk evaluated?” or “what is Risk?” Figure 2 shows the ExpertCop system interface composed by a map where crimes are plotted. The figure also shows an example of an explanation at a concrete level after a user selection on the map. For each selected place, ExpertCop identifies whether crimes occurred and allows users to request explanations.

The first explanation in Figure 2 shows the criminal agent decision “to commit the crime” because it knows that crimes are committed if the risk level is “medium”, opportunity level is “medium”, the danger level is “medium” and financial compensation is “medium”. The user requests a follow-up explanation about why the risk level was evaluated to be “medium”. The system responds that the risk level is medium because the police officer proximity is “far”, the place population density is “medium”, and the public illumination is “good”. The user requests a second follow-up explanation about why the police officer proximity is “far”. The answer was that the criminal knows that if the police distance is greater than 500 meters then the police proximity is classified as “far” and in the current situation, the police officer distance is 808 meters.

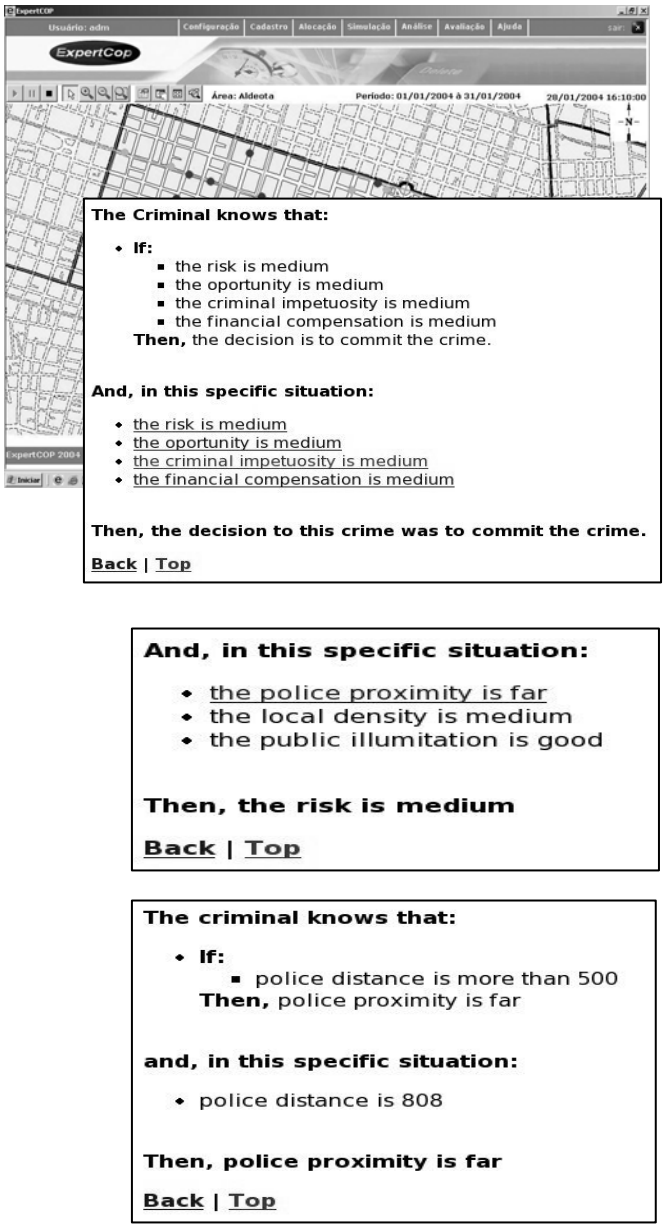

Figure 2. The ExpertCop interface and an example of explanation to the user

The explanations were derived from the PML node sets and rendered as a hypertext. The first explanation was generated from the node set concluding the abstract-and-match PSM decision subtask. The explanation presents the assertions used as premises to derive PSM answers. When premises are also derived, they are presented as a hyperlink for users to request further explanations drilling down through a proof tree branch. It is worth noting that the UPML explanation framework enabled us to provide an explanation component for ExpertCop quite easily. In particular, it was not hard to encode the PSM reasoning steps in PML. This result goes beyond ExpertCop and we claim it applies to all UPML-based KBSs.

6 Conclusion

In this paper, we addressed the issue that problem solvers need explanation components. This need is enhanced when problem solvers are automatically generated, scalable or built for extensibility. Our approach integrates the UPML architecture for KBS development and the Inference Web infrastructure for web explanations. We extended UPML and Inference Web through a new generic component, the UPML-based explanation component, which is responsible for the generation of PML documents encoding reasoning processes embedded in problem-solving methods and tasks.

We identify two major contributions of this work. The first is the definition of an approach for providing explanations about the execution of KBSs both at the abstract level and at the concrete level that can be applied in other domains. We described how the explanation component was used to generate explanations from the abstract-and-match component. Although not described in this paper, we also implemented explanations for the propose-and-revise PSMs in a laptop configuration domain. These PSMs can be used in KBSs for assessment and configuration design tasks. The generality of our solution is at the PSM level since the subtasks know how to invoke the explanation component methods in a systematical way and PSMs can be reused in any domain. Thus, UPML-based KBSs can generate proofs and explanations about their reasoning process and domain knowledge.

The second contribution is the use of PML for dumping proofs therefore enabling proof and explanation interoperability between distributed problem solvers. The focus of this work is to supply explanations of problem solvers execution to users, human or software agents. The use of PML allows the sharing of the problem solver explanations with software agents. PML proofs are not aimed at human consumption. Therefore, we intend to advance this work in the development of explanatory text generation components that produce human-readable KBS explanations from PML proofs. The goal is to project some pragmatic principles of linguistic interactions onto the semantic structures of the PML proofs in order to select the information to be conveyed to users, to simplify proof steps and to reorganize proofs.

References

[Buchanan & Shortliffe, 1984] Buchanan, B.G. and Shortliffe, E.H. Rule based expert systems: The MYCIN experiments of the Stanford Heuristic Programming Project, Addison-Wesley, Reading, MA, 1984.

[Clancey, 1986] Clancey, W.J. From GUIDON to NEOMYCIN and HERACLES in Twenty Short Lessons: ORN Final Report 1979-1985, AI Magazine, 7(3), pp. 40-60, 1986.

[Chandrasekaran, 1986] Chandrasekaran, B. Generic Tasks in Knowledge-Based Reasoning: High-Level Building Blocks. IEEE Expert 1(3), pp. 23-30, 1986.

[Fensel & Benjamins, 1998] Fensel, D. and Benjamins, V.R., Key Issues for Automated Problem-Solving Methods Reuse.13th European Conference on Artificial Intelligence, ECAI98, Wiley & Sons Pub, 1998.

[Fensel at al., 2003] Fensel, D. et al., The Unified Problem-Solving Method Development Language UPML. Knowledge and Information Systems, An International Journal, 5, 83-127, 2003.

[Figueira Filho & Ramalho, 2000] Figueira Filho, C. and Ramalho, G. Jeops – The Java Embedded Object Production System. IBERAMIA-SBIA 2000. LNAI 1952, Berlin: Springer-Verlag, 2000.

[Furtado & Vasconcelos, 2004] Furtado, V., Vasconcelos, E.. Geosimulation in education: The ExpertCop System. Proc. of Agent-based Simulation Workshop, SCS European Publisher, Lisbon, 2004

[JESS, 2005] http://herzberg.ca.sandia.gov/jess, as available on January 17th, 2005.

[McGuinness & Pinheiro da Silva, 2003] Deborah L. McGuinness and Paulo Pinheiro da Silva. Infrastructure for Web Explanations. In Proceedings of 2nd International Semantic Web Conference (ISWC2003), D. Fensel, K. Sycara and J. Mylopoulos (Eds.), LNCS 2870, Sanibel Is., FL, USA. Springer, pages 113-129, October 2003.

[Moore, 1995] Moore, J.D. Participating in Explanatory Dialogues: Interpreting and Responding to Questions In Context. Cambridge, MA, MIT Press, 1995.

[Pinheiro, Furtado & Furtado, 2004] Pinheiro, V., Furtado, E. and Furtado, V. A Unified Architecture to Develop Interactive Knowledge Based Systems. In proceedings of 17th Brazilian Symposium of Artificial Intelligence (SBIA 2004), Bazzan, Ana L.C. e Labidi, S. (Eds), LNAI 3171, São Luís, MA, Brasil, Springer-Verlag, pp 174-183, 2004.

[Pinheiro da Silva et al., 2004] Paulo Pinheiro da Silva, Deborah L. McGuinness and Richad Fikes. A Proof Markup Language for Semantic Web Services. Technical Report KSL-04-01, Knowledge Systems Laboratory, Stanford University, USA, 2004.

[Richards, 2000] Richards, D. User-Centred and Driven Knowledge-Based Systems for Clinical Support using Ripple Down Rules. Proceedings of the 33rd Hawaii International Conference on System Sciences, 2000.

[Schreiber et al, 2000] Schreiber et al., Knowledge Engineering and Management: The CommonKADS Methodology. The MIT Press. Cambridge, MA, 2000.

[Shankar & Musen, 1998] Shankar, R.D., TU, S.W., Musen, M.A. A Declarative Explanation Framework That Uses a Collection Of Visualization Agents. Stanford Medical Institute, Stanford University School of Medicine, Stanford, CA, 1998.

[Swartout, 1983] Swartout, W.R. XPLAIN: A system for Creating and Explaining Expert Consulting Programs, Artificial Intelligence, 21(3), pp.285-325, 1983.

[W3C, 2004] http://www.w3.org/2001/sw/WebOnt/