The use of ontologies in ITS domain knowledge authoring

Pramuditha Suraweera, Antonija Mitrovic and Brent Martin

Intelligent Computer Tutoring Group
Department of Computer Science, University of Canterbury
Private Bag 4800, Christchurch, New Zealand
{psu16,tanja,brent}@cosc.canterbury.ac.nz

Abstract. Acquiring the domain knowledge is a task that requires a major portion of the time and effort when building an ITS. Researchers have been exploring ways of automating the knowledge acquisition process since the inception of ITSs with limited success. All past research attempts have focussed on acquiring knowledge for procedural domains. Our goal is to develop an authoring system that acquires knowledge for procedural as well as non-procedural domains. We propose a four phase approach: composing an ontology of the domain, extracting syntax constraint from it, learning semantic constraints from the examples provided by the domain expert and finally verifying the generated constraints. This paper presents an overview of the knowledge acquisition system for acquiring knowledge for constraint-based tutors. It mainly focusses on composing the ontology and acquiring syntax constraints from it. Further work on this project will focus on learning from examples and validating the generated constraints.

1 Introduction

Acquiring domain knowledge is a major hurdle in building Intelligent Tutoring Systems (ITS) [1]. Although there have been several attempts to ease the burden on ITS developers by automating the process, they have met with limited success. All previous attempts have focussed on acquiring knowledge required for teaching procedural tasks. Our goal is to drastically reduce the time and effort required for acquiring domain knowledge by automating knowledge acquisition for intelligent tutors for both procedural and non-procedural domains.

Constraint based modelling (CBM) [2] is a student modelling approach that somewhat eases the knowledge acquisition bottleneck by using a more abstract representation of the domain compared to other commonly used approaches [3]. CBM is based on Ohlsson's theory of learning from performance errors. It focuses on correct knowledge rather than describing the student exactly as with model tracing. However, building a complete constraint base still remains a major challenge. Mitrovic reported that she took just over an hour to produce a constraint for SQL-Tutor [4], which currently contains more than 650 constraints. Therefore, the task of composing the knowledge base of SQL-Tutor would have taken over 4 months to complete. Our goal is to dramatically reduce the time and effort required for composing the knowledge base required for constraint base tutors by automating the knowledge acquisition process.

We envisage ontologies to play a central role in the whole knowledge acquisition process. A preliminary study conducted to evaluate the role of ontologies in manually composing a constraint base showed that constructing a domain ontology indeed assisted the composition of constraints [5]. The study showed that ontologies can be used to organise the constraint base into meaningful categories. This enabled the author to visualise the constraint set and to reflect on the domain assisting them to create more complete constraint bases.

The remainder of the paper is organised into five sections. The next section presents a brief description on related automatic knowledge acquisition systems. Section 3 gives an overview of our project. Details on developing the ontology is given in Section 4. Section 5 discusses the process of acquiring syntax constraints from the ontology. Conclusions and future work are presented in the final section.

2 Related work

Past research on acquiring knowledge for ITSs have solely focused on acquiring knowledge for teaching procedural tasks such as tasks in simulated environments and solving mathematical algebraic problems. The knowledge acquisition systems that acquire domain knowledge as a runnable model for evaluating student solutions include KnoMic [6], Disciple [7] and Demonstr8 [8]. All these systems acquire knowledge by observing the domain expert performing a task and generalising it to be applicable for other problems.

KnoMic is a learning-by-observation system for acquiring procedural knowledge in a simulated environment. The system observes and records the procedure taken by the domain expert in performing a task within the simulated environment. While performing the task the expert has to annotate the points where he/she had changed goals because it was either achieved or abandoned. The resulting set of observation traces are generalised by the system to learn the conditions of actions, goals and operators. During an evaluation to test the accuracy of the procedural knowledge learnt in an air combat simulator, KnoMic acquired 140 productions. Out of the total 140 created, 101 were fully correct and 29 of the remainder were functionally correct [6]. Although the results are encouraging KnoMic's applicability is limited to only simulated environments.

Disciple is a shell for developing personal agents. It relies on a semantic network of the domain that describes the domain, which can be either composed by the author or imported from a repository. Initially the shell has to be customised to the domain by building a domain-specific interface, which gives a natural way of solving problems for the domain expert. Disciple also requires a problem solver for the domain. The domain expert has to initiate the knowledge elicitation process by providing problem-solving examples. The agent generalises the provided example using a generalisation algorithm with the assistance of the domain expert. The generalised example is refined by requesting the expert to validate the examples generated by the system. As Disciple depends on problem solving instances provided by the domain expert, they should be carefully selected to reflect significant problem states. The task of selecting significant problem states requires expertise in knowledge engineering which is scarce. Furthermore, building a problem solver for some domains is extremely difficult, if not impossible.

Demonstr8 is an authoring tool for building model-tracing tutors for arithmetic. It relies on the domain expert to specify all the algebraic functions that can be used and their outcomes in the form of a table. It uses programming by demonstration to reduce the authoring effort. The system provides a drawing tool like interface for building the student interface of the ITS. The system automatically defines each GUI element as a working memory element (WME), while WMEs involving more than a single GUI element must be defined manually. The system generates production rules by observing problems being solved by an expert. Demonstr8 performs an exhaustive search in order to determine the problem-solving procedure used to obtain the solution. If more than one such procedure exists, then the user would have to select the correct one.

3 Automatic Constraint Acquisition

Existing approaches to knowledge acquisition for ITSs acquire procedural knowledge by recording the domain expert's actions and generalising recorded traces using machine learning algorithms. Even though these systems are well suited to simulated environments where goals are achieved by performing a set of steps in a specific order, they fail to acquire knowledge for non-procedural domains. Our goal is to develop an authoring system that can acquire procedural as well as declarative knowledge.

The authoring system will be an extension of WETAS [9], a web-based tutoring shell that facilitates building constraint-based tutors. WETAS provides all the domain-independent components for a text-based ITS, including the user interface, pedagogical module and student modeler. The pedagogical module makes decisions based on the student model regarding problem/feedback generation and the student modeler evaluates student solutions by comparing them to the domain model and updates the student model. The main limitation of WETAS is its lack of support for authoring the domain model.

The domain model for CBM tutors consists of a set of constraints, which are used to identify errors in student solutions. CBM focuses on correct knowledge of the domain rather than describing the student problem solving procedure [3]. As the space of false knowledge is much grater than correct knowledge, in CBM knowledge is modelled by a set of constraints that identify the set of correct solutions from the set of all possible student inputs. CBM represents knowledge as a set of ordered pairs of relevance and satisfaction conditions. The relevance condition identifies the states in which the constraint is relevant, while the satisfaction condition identifies the subset of the relevant states in which the constraint is satisfied.

As WETAS does not provide any assistance for developing the knowledge base, typically a knowledge base is composed using a text editor. Although the flexibility of a text editor may be adequate for knowledge engineers, novices tend to be overwhelmed by the task. Our goal is to reduce the time and effort required for building a constraint base by adding support for automatic constraint acquisition to WETAS. We propose a four-stage process initiated by modelling the domain as an ontology. The ontology would be composed by a domain expert using the ontology modelling tool. Once the ontology is completed, the system would analyse the ontology and extract syntax constraints directly from the completed ontology. During the third phase, the system would acquire constraints by analysing sample solutions provided by the expert. Finally the constraint set is validated with the assistance of the domain expert, where the expert would label the system generated examples as correct or incorrect.

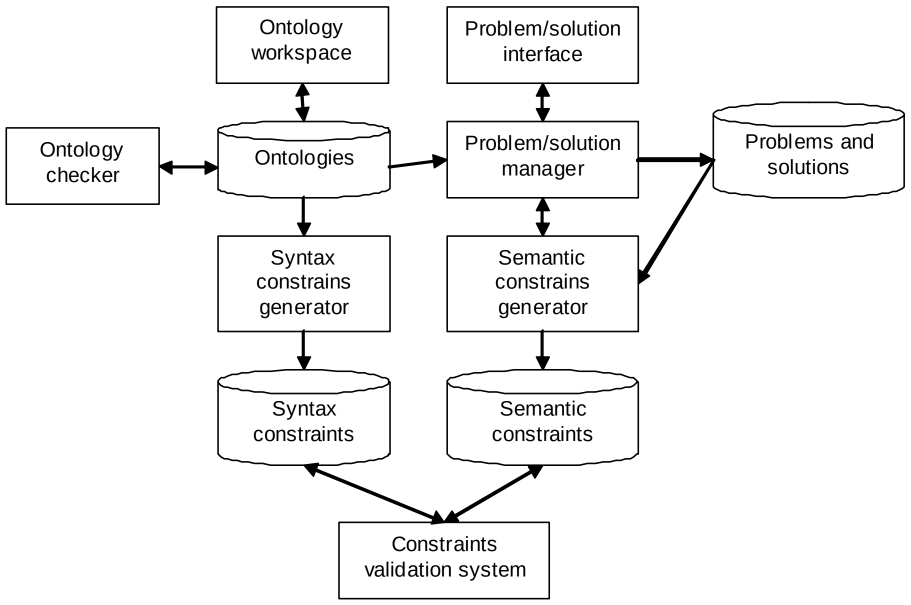


Fig. 1. Interface of WETAS front end

The architecture of the knowledge acquisition system for CBM consists of an ontology workspace, ontology checker, problem/solution manager and syntax and semantic constraint generators, as depicted in Figure 1. During the initial phase the domain expert models an ontology of the domain in the ontology workspace. The ontology checker validates the ontology during the ontology composition state. The completed ontology is stored in the ontology repository.

The syntax constraints generator analyses the completed ontology and generates syntax constraints from it. The generated syntax constraints are stored in the syntax constraints repository. The generation of constraints from a domain ontology is discussed further in Section 5.

The domain expert has to specify the representation for solutions prior to entering problems and sample solutions. The solution representation is a decomposition of the solution into components consisting of a list of instances of concepts. For example, a sentence in English consists of a list of words and a list of punctuation marks.

The domain expert has to enter sample problems and their solutions during the third phase of knowledge acquisition. The problems/solution interface assists the user by providing a dynamic form that consists of input boxes for populating each property of the concept instance. The expert is requested to provide different solutions that depict different ways of solving the same problem. While the expert enters in an alternative correct solution, the system attempts to match each component of the solution to components of the initial solution. These matches are later used to compose a set of semantic constraints that compare the student's solution against the system's ideal solution. The expert is also encouraged to supply solutions containing typical errors made by students. The system would use these erroneous solutions to identify typical errors in student solutions and provide more detailed assistance. The system also verifies the solutions provided by the expert using the generated syntax constraints. If a discrepancy is identified, the user is alerted and the solution may be modified to comply with the ontology or vice versa.

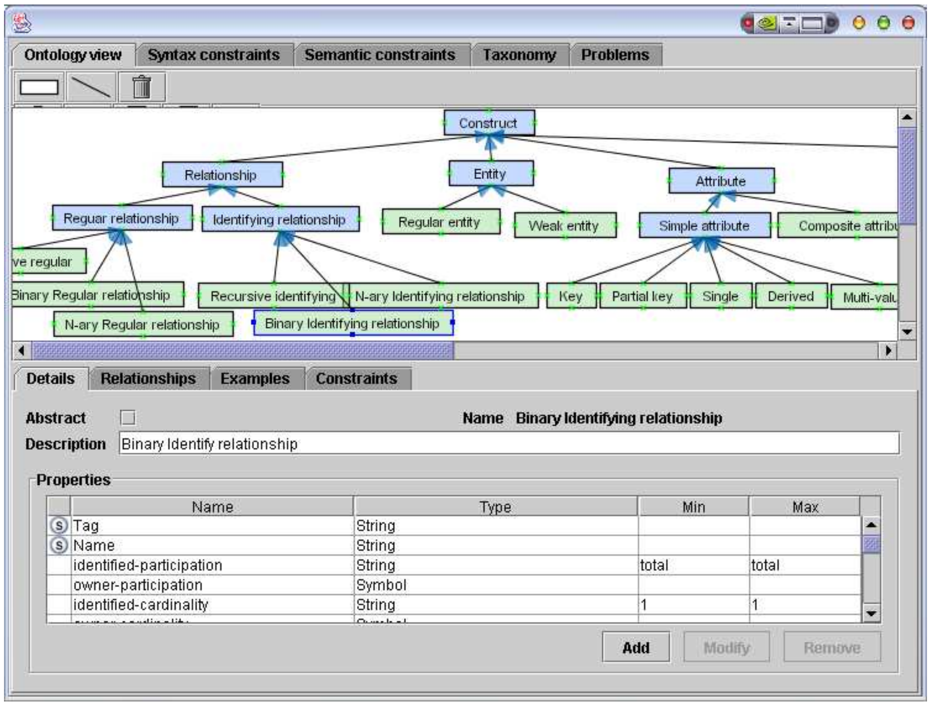
The final phase involves ensuring the validity of each and every constraint. During this phase the domain expert may wish to investigate the complete set of constraints to identify redundancies or erroneous constraints. The erroneous constraints can be either directly modified by expert users or the user may opt to provide new examples to illustrate why the constraint is invalid.

4 Developing the ontology

As discussed earlier, we envision the knowledge authoring process to be initiated by developing the domain ontology. The ontology will be later used to generate constraints automatically. An ontology describes the domain, by identifying all the important domain concepts and various relationships between them. The ontology workspace provides an environment for composing the domain ontology in terms of concepts and their sub concepts as shown in Figure 2. All concepts are represented using rectangles and they are related to their sub concepts using arrows. The interface has no restrictions in placing concepts within the workspace. The user can position the concepts to display a hierarchical structure. The completed ontology is saved on a central server in XML format.

The ontology displayed in Figure 2 represents the concepts of ER modelling, a popular database modelling technique. The ER model describes data as entities, attributes and relationships. An entity is the basic object represented in the ER model, which is a 'thing' in the real world with an independent existence. Each entity has particular properties, called attributes, that describe it. A relationship is an association between two or more entities.

The ER ontology depicted in Figure 2 contains *Construct* as the most general concept. *Relationship*, *Entity*, *Attribute* are sub-concepts of *Construct*. *Relationship* is specialised into *Regular* and *Identifying*, which are the two types of relationships and *Entity* is specialised, according to its types, as *Regular* and *Weak*. Subclasses of *Attribute* are *Simple* or *Composite* attributes and *Simple* attributes are further specialised into five categories namely *Key*, *Partial key*, *Single*, *Derived* and *Multi-valued*.



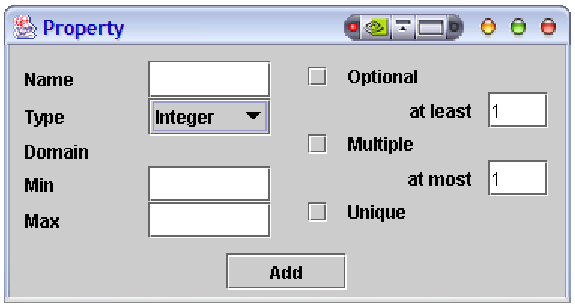
Abstract ☐ Name Binary Identifying relationship

Description: Binary Identify relationship

Name	Type	Min	Max
Ⓢ Tag	String		
Ⓢ Name	String		
identified-participation	String	total	total
owner-participation	Symbol		
identified-cardinality	String	1	1

Fig. 2. Ontology workspace interface

Each concept has a set of properties that describes itself. The property addition interface shown in Figure 3 allows the specification of the property's type (integer, float, boolean, string or symbol) and restrictions on its domain. The range of values that the property may hold can be specified in terms of minimum and maximum values or as a set of distinct values. Other restrictions include specifying that the value of a property is unique, optional or can contain multiple values.



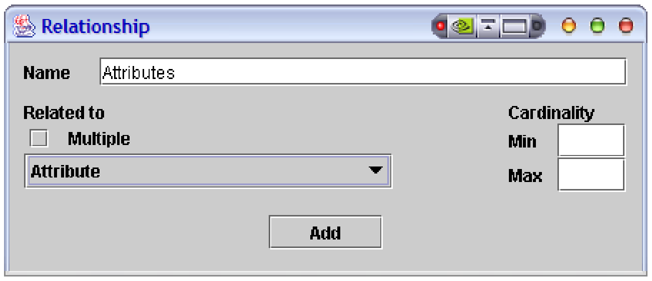
Name		☐ Optional	
Type	Integer	at least	1
Domain		☐ Multiple	
Min		at most	1
Max		☐ Unique	

Fig. 3. Concept property interface

An identifying relationship in ER modelling relates a regular entity type (the owner) to a weak entity type. It is necessary to specify the participation of the owner and the weak entity type in the relationship.

The properties of the *Binary identifying relationship* concept are shown in Figure 2. Its properties include *name*, *identified participation*, *owner participation* and *identified cardinality*. Most properties are of type 'string' except *owner participation* and *owner cardinality* are of type 'symbol'. The value of *identified participation* property is always 'total' and the value of *identified cardinality* is always 1.

Figure 4 illustrates an interface for specifying relationships between two or more concepts. The number of concept instances that participate in the relationship can be restricted by specifying a minimum and maximum cardinality.



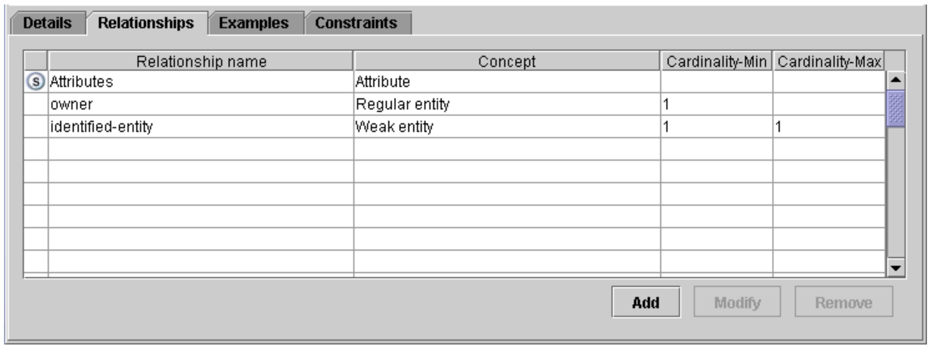
Name: Attributes

Related to
☐ Multiple
Attribute

Cardinality
Min
Max

Fig. 4. Concept relationship interface

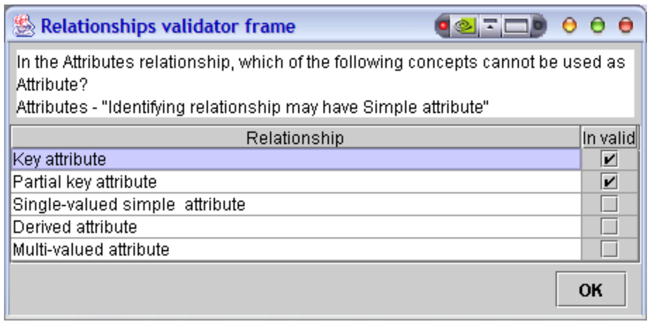
The relationships that are involved with *Binary identifying relationship* concept are detailed in Figure 5. They include *attributes*, *owner* and *identified entity*. The *attributes* relationship is a relationship between *Binary identifying relationship* and *Attribute* with no restrictions on the cardinality. The *owner* relationship with *Regular entity* has a minimum cardinality of 1 and the *identified entity* relationship with *Weak entity* has a minimum and maximum cardinality of 1.



	Relationship name	Concept	Cardinality-Min	Cardinality-Max
S	Attributes	Attribute		
	owner	Regular entity	1	
	identified-entity	Weak entity	1	1

Fig. 5. Relationships for 'Binary identifying Entity' concept

During the task of composing an ontology, the domain experts may add relationships that are too general. Since constraints are composed directly from the relationships found in the ontology, it is imperative that the relationships are valid. In order to ensure that all added relationships are completely accurate, the system engages the user in a dialog. During this dialog the user is presented with lists of specialisations of concepts involved in the relationship and is asked to label the specialisations that violate the principles of the domain. As an example consider the relationship between *Binary identifying relationship* and *Attribute*. As shown in Figure 5 the initial question posed asks whether each of the specialisations of *attribute* (*key*, *partial key*, *single-valued* etc) are applicable to the *attributes* relationship. The user would indicate that *key* or *partial key* attributes cannot be used instead of attribute in the *attributes* relationship. The system replaces the original relationship with a more specific one at the completion of the dialog.



In the Attributes relationship, which of the following concepts cannot be used as Attribute?
Attributes - "Identifying relationship may have Simple attribute"

Relationship	In valid
Key attribute	✓
Partial key attribute	✓
Single-valued simple attribute	
Derived attribute	
Multi-valued attribute	

Fig. 6. Relationship validate dialog for 'Entity has attribute' relationship

The specialisations of the relationship that are labelled as invalid by the user are later used by the constraint generator to generate syntax constraints.

The specialisations marked as invalid by the user in Figure 6 would be used to generate two constraints: *Binary identifying relationship* cannot have a *key* attribute and *Binary identifying relationship* cannot have a *partial key* attribute.

5 Acquiring Syntax Constraints from the Domain Ontology

An ontology contains a lot of information about the domain and is much easier to create than the final domain model. The restrictions on attributes and relationships specified in the ontology can be directly translated into constraints that deal with the syntax of the domain. As an example, consider the *owner* relationship of *Binary identifying relationship*: it has a minimum cardinality of 1. This restriction specifies that each *Binary identifying relationship* has to have at least one *Regular entity* participating as the *owner.* It can be translated to a constraint that asserts that an *Identifying relationship* has to have at least one *Regular entity* as its owner.

The system generated six constraints from the restrictions specified in the *Binary identifying relationship*;

- *Binary identify relationship* must have at least 1 *Regular entity* as the *owner*
- *Binary identify relationship* must have exactly 1 *Weak entity* as the *identified entity*
- The *identified participation* property of *Binary identify relationship* must be total
- The *identified cardinality* property of *Binary identify relationship* must be 1
- The *name* property of *Relationship type* has to be unique
- *Relationship type* must have exactly 1 *name*

The syntax constraints generator produced a total of 48 syntax constraints from the ontology for ER modelling, depicted in Figure 2. The generated set of constraints covered all syntax constraints that existed in KERMIT [10], a CBM based ITS developed for ER modelling. Although the initial results are derived from only a single domain, we believe that the system would be able to successfully handle most non-procedural domains. The ontology workspace would be enhanced to handle procedural domains by adding further constructs.

6 Conclusions and Future Work

We provided a brief overview of our main research objective: automatically acquire domain knowledge required for constraint-based tutors. We propose a four phase process, initiated by modelling a domain ontology. The system then analyses the completed ontology and extracts syntax constraints from it. During the third phase, the expert provides problems and their solutions and the system generates semantic constraints by analysing the solutions. Finally, the induced constraint set is validated with the assistance of the user.

The paper included a detailed description of the first two phases: modelling the ontology and extracting constraints from it. The initial tests conducted on acquiring constraints from an ontology composed for ER modelling produced encouraging results. The system generated the complete set of syntax constraints found in KERMIT, a constraint based tutor developed for the same domain.

Currently we are working on acquiring semantic constraints from examples provided by the domain expert. We will be exploring machine learning algorithms such as learning from examples and learning from analogy for automatically acquiring semantic constraints. The ontology workspace will also be enhanced to handle procedural domains. Several new constructs would have to be introduced to the workspace to cater for procedural domains.

Finally the system will be thoroughly evaluated to test its effectiveness. Most importantly, the quality and the correctness of the knowledge base generated by the system have to be evaluated. Since this research aims to produce a system that is capable of acquiring knowledge for a vast range of domains not restricting itself to a particular set, it will be tested in different domains. The usability of the system will also be tested.

Acknowledgements The work reported here has been supported by the University of Canterbury Grant U6532.

References

1. Murray, T.: Expanding the knowledge acquisition bottleneck for intelligent tutoring systems. IJAIED **8** (1997) 222–232
2. Ohlsson, S.: Constraint-based student modelling. In: Student Modelling: the Key to Individualized Knowledge-based Instruction, Berlin, Springer-Verlag (1994) 167–189
3. Mitrovic, A., Koedinger, K., Martin, B.: A comparative analysis of cognitive tutoring and constraint-based modeling. In Brusilovsky, P., Corbett, A., Rosis, F.d., eds.: UM2003, Pittsburgh, USA, Springer-Verlag (2003) 313–322
4. Mitrovic, A.: Experiences in implementing constraint-based modelling in sql-tutor. In Goettl, B.P., Halff, H.M., Redfield, C.L., Shute, V.J., eds.: ITS 98, San Antonio (1998) 414–423
5. Suraweera, P., Mitrovic, A., Martin, B.: The role of domain ontology in knowledge acquisition for ITSs. In: ITS 2004. (2004) to appear
6. van Lent, M., Laird, J.E.: Learning procedural knowledge through observation. In: International conference on Knowledge capture, Victoria, British Columbia, Canada, ACM Press (2001) 179–186
7. Tecuci, G.: Building Intelligent Agents: An Apprenticeship Multistrategy Learning Theory, Methodology, Tool and Case Studies. Academic press (1998)
8. Blessing, S.B.: A programming by demonstration authoring tool for model-tracing tutors. IJAIED **8** (1997) 233–261
9. Martin, B., Mitrovic, A.: Domain modeling: Art or science? In U. Hoppe, F.V..J.K., ed.: AIED 2003, IOS Press (2003) 183–190
10. Suraweera, P., Mitrovic, A.: Kermit: a constraint-based tutor for database modeling. In Cerri, S., Gouarderes, G., Paraguacu, F., eds.: ITS 2002, Biarritz, France (2002) 377–387