INTELLIGENT ASSISTANCE IN A PROBLEM SOLVING ENVIRONMENT FOR UML CLASS DIAGRAMS USING A HYBRID EXPERT SYSTEM

Hilke Garbe
Carl von Ossietzky University Oldenburg
26111 Oldenburg
Germany

ABSTRACT

In this paper an Intelligent Problem Solving Environment for UML Class Diagrams is presented which uses a new hybrid expert system. This hybrid expert system combines a generative expert system and a constraint based approach to support learners while solving UML modeling tasks. Using this combination we are able to give different types of help without having to model all possible errors that can occur while solving the tasks. The generative expert system is used to complete an embeddable learner's solution proposal and the constraints are used to correct erroneous student proposals. New tasks can be added to the system without any knowledge of the underlying expert systems.

KEYWORDS

Intelligent Problem Solving Environment, UML Class Diagrams, generative expert system, constraints

1. INTRODUCTION

UML is a standard modeling language in the field of computer science. We aim at supporting the students while learning to use UML class models. As they often want to train their skills and get feedback for their solution proposals, we developed an Intelligent Problem Solving Environment (IPSE). Using this IPSE they can solve given tasks using the UML modeling tool Fujaba[1] and ask the system for help.
As two user groups are intended for our system we aim at supporting both of them. On the one hand, learner's should get more than one type of help: According to the hypotheses testing approach described in the ISP-DL-theory (Möbus et al 2003) students should be supported in building hypotheses about their solution proposal and ask questions like: "Is my solution proposal or a part of it embeddable in a correct solution?" If it is, the system should be able to perform the next solution step and present it to the student. If it is not, the system should additionally be able to correct the erroneous solution proposal.
On the other hand, authors should be able to integrate new tasks in the IPSE as easy as possible without any knowledge about the underlying expert systems.

2. EXPERT SYSTEMS FOR INTELLIGENT LEARNING ENVIRONMENTS

Generative expert systems are used e.g. for the Cognitive Tutors developed at the Carnegie Mellon University or the intelligent problem-solving environments (Möbus et al 2003, Garbe 2012) of the Department "Learning and Cognitive Systems" at the University of Oldenburg. One of the advantages of these systems is their ability to generate solutions for the given tasks, which enables them e.g. to complete an embeddable learner's solution proposal to a complete solution. Different types of knowledge representations

[1] www.fujaba.de

are used for these expert systems: Cognitive Tutors use production rules, IPSEs use e.g. goals-means-relations depending on the domain. But to give correcting hints for erroneous student proposals is difficult, because knowledge about the possible errors has to be incorporated in the expert system. So, often studies about the errors occurring in students' solution proposals have to be performed. Modeling these errors is very cumbersome, because in most domains more erroneous proposals occur than correct solutions.

On the other hand the **constraint-based tutoring systems** e.g. of the Intelligent Computer Tutoring Group at the University of Canterbury (Baghaei 2007) use a constraint based approach. Instead of modeling the problem solving process they describe correct solutions with constraints (like e.g. in Figure 1) that have to be fulfilled. These are evaluated with regard to an ideal solution. If a domain constraint is violated an error occurred. Possible correct variations of the ideal solution have to be considered in the constraints. E.g. in a class diagram it is possible to model a relationship between two classes as an association or as an attribute in one or both of the classes. Using the constraints errors can be detected in a learner's proposal without having to model them explicitly. But without an incorporated problem solving process they are not able to perform the next problem solving step to help the learner.

The following articles compare both approaches in detail: (Mitrovic et al 2003), (Kodaganallur et al 2005), and (Mitrovic and Ohlsson 2006). (Baghaei 2007) and (Le 2006) describe constraint based UML modeling tutoring systems.

2. THE INTELLIGENT PROBLEMSOLVING ENVIRONMENT

2.1 The hybrid approach: Integrating a generative expert system and constraints

Using our hybrid approach we are able to combine the above described advantages of the two specific systems. We use a generative expert system that is able to generate the possible solutions and a constraint based approach to detect errors.

The generative expert system uses graphs to represent the UML class diagrams as well as additionally generated metadata e.g. about class or method hierarchies (Eden and Nicholson 2011). Transforming the UML diagrams for the solutions into these graphs we are able to detect correct variations of diagrams that are valid for all tasks: e.g. if a link between two classes is modeled as association or as attribute. Additionally authors may assign more than one solution to a task containing task specific variations (Figure 2, left side).

One major challenge in enhancing this system with constraints is the identification of the most similar solution with regard to a possibly erroneous solution proposal. This is done by a stepwise exact and error-correcting graph matching process using A*-search and constraint satisfaction problems with degree heuristic (Russell and Norvig 2010). Having detected the most similar solution diagram it can be used to generate the next solution step or to evaluate the constraints (Figure 2, left side). As the correct variations of the diagrams are described in the graphs they need not be considered in the constraints anymore. Figure 1 shows a simple constraint incorporated in the system.

Constraint

- **Relevance Condition Cr**:
  IF the user diagram contains a concrete class X and the solution contains an according class Y
- **Satisfaction Condition Cs**:
  THEN this solution class Y should also be concrete.
- **Hint:**
  If the constraint is violated, the system generates the hint:"No objects should be generated from the class X." A second hint would be:"The class X should be an abstract class." Within these sentences the constraint interpreter instantiates the names of the classes.

Figure 1. Example of a constraint

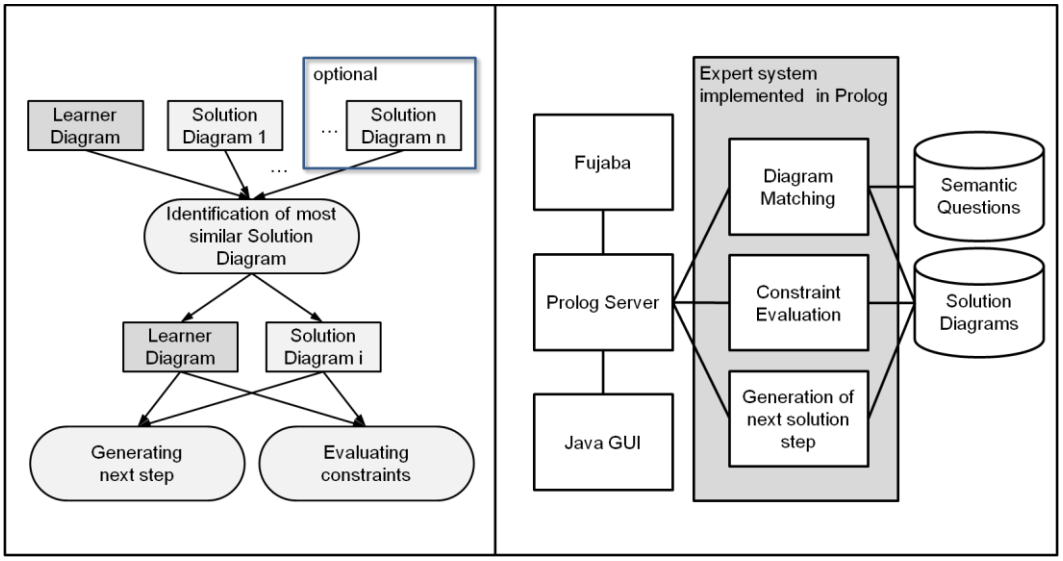

Figure 2. Architecture of the system

2.2 Working with the IPSE

The learner is given a modeling task which he can work on in the UML-Modeling Tool Fujaba. In the shown example he is asked to develop a metamodel for SADT-Diagrams. This task was also part of the evaluation study because metamodeling is part of the software-engineering class at our university and the students wanted to train their metamodeling skills.

Solving this task learners are not restricted in using certain names for their diagram elements (Figure 3). Asked for help the system tries to find a graph matching with the solution graphs. To confirm or decline the graph matching the learner is asked some questions concerning the semantic of his diagram elements relating them to the text of the task, e.g.: "Does the class "Void" model the environment?" As long as the user declines some matches the system tries to find new matches. Having found a match that is affirmed by the user the system is able to give the different types of help described above. As the shown proposal is not embeddable in a correct solution, because the class "Flow" should be an abstract class, the system gives the hints associated with the constraint shown in Figure 1. After correcting the class the user can ask for a next solution step. The system presents the new diagram to the user (Figure 4) in a new window. Now, the user can choose to add the new element to his diagram.

2.3 Architecture

The architecture of the system is shown in Figure 2 on the right side. It is integrated in Fujaba as a Plug-In[2]. As we wanted to allow the user to choose whether he wants to add the elements proposed by the system as next solution step to his diagram, we implemented a display component for UML class diagrams[3] in Java (Figure 4, left side). The hybrid expert system is implemented in SWI-Prolog[4]. It contains e.g. the diagram matching algorithms and a metainterpreter for the constraints. The solution diagrams and the semantic questions like "Does the class X model the environment?" are stored in simple text files. Only these have to be changed when new tasks are added by an author using the authoring tool.

[2] Andreas Schäfer implemented the Fujaba Plug-In and Jan-Patrick Osterloh the data-exchange between Java and Prolog.
[3] Lars Weber implemented the component displaying the completion proposal of the system.
[4] www.swi-prolog.org

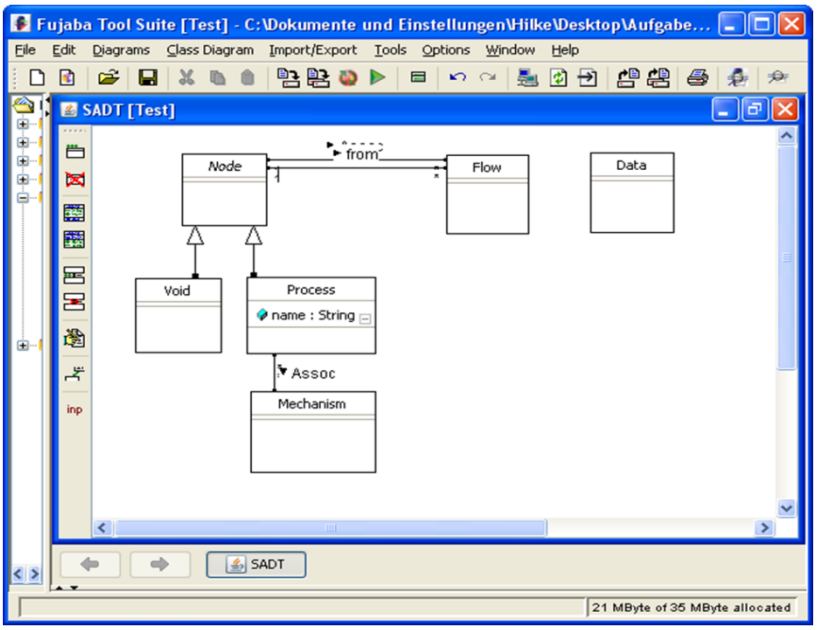

Figure 3. Learner's solution proposal modeled in Fujaba

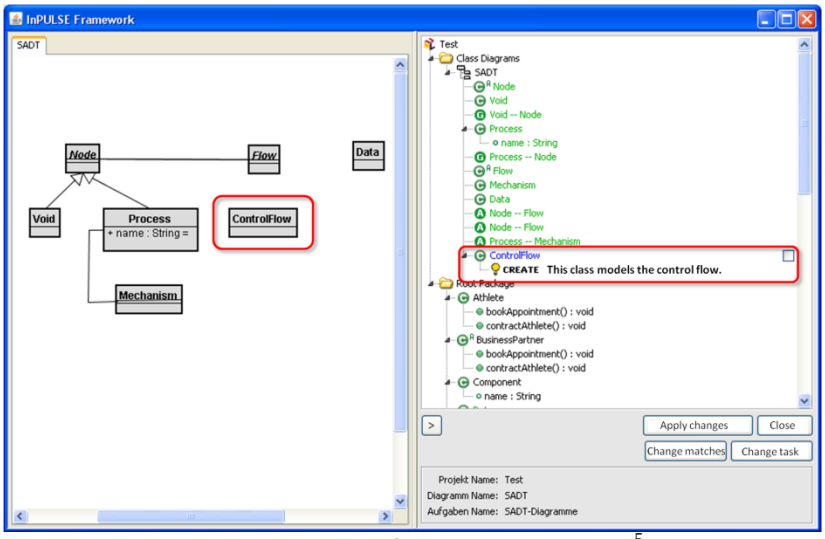

Figure 4. Generation of the next solution step[5]

2.4 Evaluation

So far we have evaluated the system with 14 students from a "Software Engineering" class and an "eLearning" class. The evaluation sessions lasted about 2 hours containing (1) a prephase with a questionnaire e.g. about their experience in using UML modeling tools and eLearning tools, (2) about 1.5 hours of working with the system, and (3) a postphase with a feedback questionnaire about how they liked to work with the IPSE and the different types of help. A majority of the students answered that they liked it "very well" or "well" to work with the system. As we hoped some of them showed preferences for the one or

[5] The texts displayed by the system have been translated as the original messages are in german.

the other kind of help. Another evaluation as think-aloud session is planned with the tutors of the Software Engineering course to further improve our feedback messages.

3. CONCLUSION

As described above we use a new hybrid expert system to support learners in training their UML modeling skills. This hybrid expert system enables us to support hypothesis testing with respect to the ISP-DL theory as well as to give correcting hints for erroneous student proposals without the need to model the possible errors explicitly. Using graph matching algorithms we are, in contrast to most constraint based tutors, able to work with more than one ideal solution for every task. Especially in the domain of UML class modeling this is important. New tasks can be added easily by authors, as they just need to specify the solution diagram(s) using Fujaba and the semantic question concerning the diagram elements using an authoring system.

In order to support authentic learning tasks (Van Merrienboer and Kirschner 2007) we integrated our learning environment in the UML modeling tool Fujaba.

REFERENCES

Baghaei, N., 2007. *A Collaborative Constraint-Based Intelligent System for Learning Object-Oriented Analysis and Design Using UML*. PhD Thesis, University of Canterbury.

Eden, A.H. and Nicholson, J, 2011. *Codecharts: Roadmaps and blueprints for object-oriented programs*, John Wiley & Sons, Hoboken, NJ, USA.

Garbe, H., 2012, Intelligent Assistance in a Problem Solving Environment for UML Class Diagrams by Combining a Generative System with Constraints, accepted for *eLBa 2012*.

Kodaganallur, V. et al, 2005. A Comparison of Model-Tracing and Constraint-Based Intelligent Tutoring Paradigms. *International Journal of Artificial Intelligence in Education*, Volume 15, Issue 2 (April 2005), p. 117-144.

Le, N.-T., 2003. A constraint-based assessment approach for free-form design of class diagrams using UML. *Proceedings of the Workshop on Intelligent Tutoring Systems for Ill-Defined Domains, the 8th Conference on ITS*, p. 11 – 19.

Mitrovic, A. et al, 2003. A Comparative Analysis of Cognitive Tutoring and Constraint-Based Modelling. *User Modeling 2003: 9th International Conference (UM 2003)*, Johnstown, PA, USA, LNCS 2702, p. 313-322.

Mitrovic, A. and Ohlsson, St., 2006. A Critique of Kodaganallur, Weitz and Rosenthal, "A Comparison of Model-Tracing and Constraint-Based Intelligent Tutoring Paradigms". *International Journal of Artificial Intelligence in Education*, Volume 16, Issue 3, p. 277-289.

Möbus, C. et al, 2003: Towards an AI-Specification of Intelligent Distributed Learning Environments. *KI - Zeitschrift Künstliche Intelligenz Heft 1/03 "Schwerpunkt: Lernen: Modellierung und Kommunikation"*, Bremen: arendtap Verlag, p. 19-24.

Russell, St. and Norvig, P., 2010: *Artificial Intelligence: A Modern Approach*, Prentice Hall.

Van Merrienboer, J.J.G., Kirschner, P.A., 2007: *Ten Steps to Complex Learning: A Systematic Approach to Four-Component Instructional Design*, Lawrence Erlbaum Assoc Inc.