Intelligent Assistance in a Problem Solving Environment for UML Class Diagrams by Combining a Generative System with Constraints

Hilke Garbe

Department of Informatik
Carl von Ossietzky University Oldenburg
26111 Oldenburg
hilke.garbe@uni-oldenburg.de

Abstract: In this paper we present a Problem Solving Environment for UML Class Diagrams. To support the students' learning process, we use a new approach for the expert system generating the help. This new approach uses a generative expert system enhanced with a constraint based system to combine the advantages of the two specific systems: complete a correct solution proposal to a solution and give hints for correcting an erroneous solution proposal. We aim at developing the combined system in such a way that enables us to provide an authoring system for new tasks. Using the authoring system new tasks can be supported by the system without extension of the expert systems.

1 Introduction

UML Class Diagrams are a fundamental technique for computer scientists. In the program of study at our university they are taught in the third semester and used throughout the rest of the program. Often the tutors of the courses are asked for more modeling tasks that help the students to prepare themselves for the exams.

We aim at helping the students to train their skills in developing such UML Class Diagrams by providing them with an Intelligent Problem Solving Environment (IPSE) for such diagrams. In this IPSE they can solve given modeling tasks and ask the system for different types of help supporting the problem solving process.

Mainly four requirements lead the design of our IPSE:

1. **Free form design of UML Class Diagrams**
   Van Merrienboer and Kirschner state that "*authentic learning tasks based on real life tasks […] help learners to integrate knowledge, skills and attitudes, stimulate them to learn to coordinate constituent skills, and facilitate transfer of what is learned to new problem situations*" [MK07].
   So we wanted to integrate the IPSE into a UML modeling tool and restrict the modeling facilities of the tool as little as possible.

2. **Support problem solving with respect to the ISP-DL theory (Impasse - Success - Problem - Solving - Driven – Learning) [MAG+03]**
   The system has to be able to test learners' hypotheses about their solution proposal like: "Is (this part of) my solution proposal correct?" The system then tries to embed the solution proposal of the student into a system generated solution. If it is embeddable, the system should generate the next step(s) towards this solution. As the user's part of the solution is embedded in the system's solution the system is able to give adaptive feedback. If the user's solution proposal cannot be embedded the user is asked to mark a part of his proposal which he assumes is correct and ask the system again.

3. **Give hints to correct erroneous student solution proposals**
   In case the learner's solution proposal contains errors and therefore cannot be embedded in a solution, the system should give hints about the parts that have to be corrected and how this can be done.

4. **Support the authoring of new tasks**
   The expert system generating the help should be designed in a way that enables tutors to incorporate new tasks without any knowledge of the knowledge representation. This is because the building of new tasks should take as little time as possible.

2 Expert systems for intelligent learning environments

In the area of intelligent learning environments two different approaches for the expert systems have been established as successful:

On the one hand, there are the systems using a generative expert system, e.g. the Cognitive Tutors developed at the Carnegie Mellon University described e.g. in [ACK+95] or the Intelligent Problem-Solving Environments (IPSEs) [MAG+03] of the Department "Learning and Cognitive Systems" at the University of Oldenburg. On the other hand there are the constraint-based tutoring systems e.g. of the Intelligent Computer Tutoring Group at the University of Canterbury e.g. described in [MO06a] and [Ba07]. In [MKM03], [KWR05] and [MO06b] both approaches are compared.

2.1 Generative expert systems

These systems use generative expert systems that model the users' problem solving process and therefore are able to generate solutions for the given task. By doing this they can e.g. complete a correct solution proposal to a solution. The knowledge representations used for these expert systems vary: Cognitive Tutors use production rules, IPSEs use e.g. goals-means-relations [MAG+03] depending on the domain. But to enable these systems to correct erroneous solution proposals is very cumbersome because the possible errors have to be incorporated in the system's knowledge of the problem solving process, e.g. by incorporating rules representing these errors.

2.2 Constraint Based Tutors

The advantage of Constraint Based Tutors on the other hand is the ability to give hints for correcting an erroneous solution proposal without having to model all possible errors. In general they do not include a problem solving algorithm [Ma01]. Instead constraints are used that have to be satisfied for the different states of the solution. "*Each constraint consists of two parts: a relevance condition, Cr, and a satisfaction condition, Cs. The relevance condition identifies the pedagogically relevant states and the satisfaction condition identifies states where this piece of knowledge has been correctly applied. A generic constraint states that if Cr is true, then Cs has to be also true. Otherwise, something has gone wrong and the feedback message attached to the violated constraint is shown to the student*."[Ba07] The weakness of these systems is that they are not able to complete learners' solution proposals because they consider only states of the solution and not the necessary steps of a Problem Solving Process. This drawback means that our requirement 2 is not met.

2.3 Tutors for UML-Class-Diagrams

At least two approaches to assist users in learning UML-Class-Diagrams using a constraint based tutor are described in literature.
Baghaei developed a “Collaborative Constraint-Based Intelligent System for Learning Object-Oriented Analysis and Design Using UML” [Ba07]. The system supports single users as well as collaborative work. A major drawback of the system with respect to our requirements is that it uses its own Class Diagram editor which allows only the class, attribute and method names that are predefined in the task description. This violates our requirement 1.

Le, too, describes in [Le06] a constraint based approach of a learning system for Class Diagrams. The system is integrated in ArgoUML, allows free-form-modeling of diagrams and therefore meets our requirement 1. But according to [Le06] the system only analyses classes and associations with respect to an ideal solution. Attributes, methods and type information aren’t considered.

3 Combining a generative expert system and constraints

To meet all our requirements we combine the two approaches mentioned above. We use a generative expert system to generate the correct solutions and constraints to detect errors in the learner’s solution proposal. As the semantic constraints used in the constraint based tutors compare the learner’s solution proposal with one ideal solution implemented in the system (e.g. [Ba07]) one major challenge of our approach was to detect the solution generated by the system that is most similar to the user’s proposal. This solution is used to evaluate the constraints.

The bases of the expert system are graphs that describe the solutions. As nodes they contain both the elements of the UML diagram as well as automatically generated metadata based on the formal design description language LePUS3 [EN11] such as inheritance hierarchies of classes and methods. Generating these abstract metadata automatically for the given ideal solutions and learner’s solution proposals is one of the implemented strategies to identify correct variations of diagrams independent of a special task. These graphs are used as well for the generation of completions for the solution proposals of the students as for the definition of the constraints.
Using the same data structure containing the abstract metadata for the diagrams reduces the development effort for the combined approach.

As mentioned above, it is necessary to identify the system's solution that is most similar to the user's proposal to evaluate the constraints. This is done stepwise using e.g. an exact or relaxed graph matching implemented as Constraint Satisfaction Problem using the Degree Heuristic [RN10]. Having calculated the most similar solution the constraints evaluate the user's solution proposal against this ideal solution.
An example for an incorporated constraint is:
Relevance Condition Cr: IF the solution contains a method hierarchy but the method is just contained in the Superclass AND the diagram of the user contains an according hierarchy of classes and methods
Satisfaction Condition Cs: THEN any of the Subclasses in the user diagram should redefine the according method.
If the constraint is violated, the system gives the hint: "The method X should be implemented in the Superclass Y and not in the Subclasses." The system instantiates the variables X and Y with the names of the according elements in the learner's solution proposal.

4 Working with the InPULSE system

Working with the system the learner is given a modeling task. In the shown example his task is to model the relation between a sportsman, his manager and a sponsor using the Proxy design pattern. The IPSE is integrated in the UML-Modeling Tool Fujaba[1] and the learner is not restricted in using certain names for his diagram elements (Requirement 1). Figure 1 shows a possible learner's solution proposal.

[1] http://www.fujaba.de/

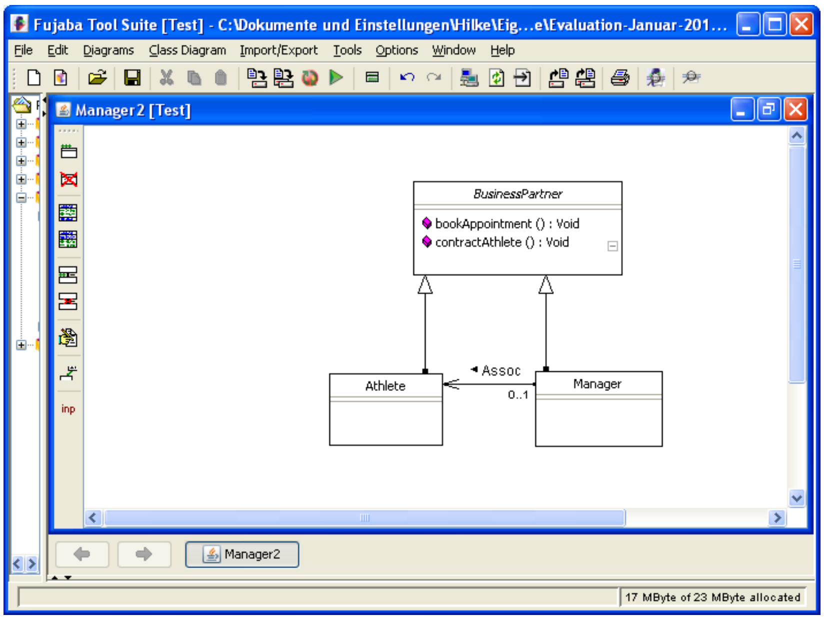
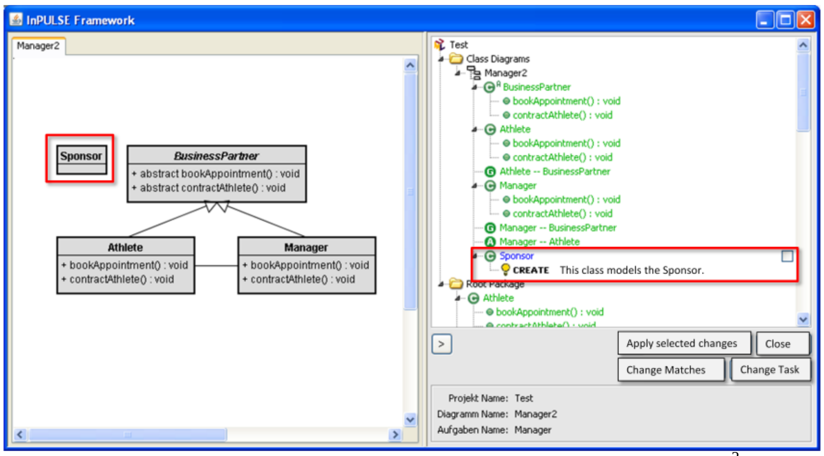

Fujaba Tool Suite [Test] - C:\Dokumente und Einstellungen\Hilke\Eig...e\Evaluation-Januar-201...
File Edit Diagrams Class Diagram Import/Export Tools Options Window Help
Manager2 [Test]
BusinessPartner
bookAppointment () : Void
contractAthlete () : Void
Assoc
0..1
Athlete
Manager
inp
Manager2
17 MByte of 23 MByte allocated

Figure 1: Proposal of a learner modeled in Fujaba

If the user wants to ask the system for help, the system analyses his proposal by trying to find a graph matching with the solution graphs. To reduce the complexity of the graph matching it is performed stepwise with an increasing set of considered diagram element types. Then the learner is asked a few questions about the meaning of his diagram elements, to affirm or decline the element matches. In the given example this e.g. would be: "Does the class "Athlete" model the sportsman?" In case he declines all or some of the matches the system tries to find a new match. Having found a match that is affirmed by the user the system is able to give different types of help:

1. **Analyse Diagram**: analyses if the diagram or a marked part of it is embeddable in a correct solution. (Requirement 2)
2. **Hint**: Using the incorporated constraints the system gives a textual hint about errors or missing elements (Requirement 3). In the case of the diagram shown in figure 1 this would be:"*The method bookAppointment of the class BusinessPartner should not implement any behaviour. This should be done in the subclasses.*" Asking again for a hint, the system answers:*"The method bookAppointment of the class BusinessPartner should be an abstract method."*

3. **Show next step**: if the (marked part of the) solution proposal is embeddable in a correct solution, the system generates a new element of the solution and presents the new diagram to the user (Figure 2) together with a text explaining the new element. Now, the user can choose to add the new element to his diagram.



Projekt Name: Test
Diagramm Name: Manager2
Aufgaben Name: Manager

Figure 2: Generating the next step for the learner's solution proposal[2]

4.1 Authoring new tasks

The new hybrid approach enables us to provide authors with an authoring system which is, in contrast to CTAT ([ASM+06], [AMS+09]) or Aspire ([MMS+09], [SMM10]), domain specific. Using this domain specific authoring system has the advantage that authors do not need any knowledge about the knowledge representation or the specification of the problem solving process. Due to the automatic generation of metadata, new tasks can be created by specifying one or more solution diagrams using Fujaba and specifying the questions about the meaning of the diagram elements (Requirement 4). This is nearly the same effort as preparing tasks for face-to-face classroom tutorials.

[2] The texts displayed by the system have been translated for this paper.

5 Evaluation

An evaluation of the system is taking place at the moment. During an evaluation session the students were asked to answer some questions e.g about their experience in using UML-modelling tools and eLearning systems. Then they worked about 90 minutes with the system. As a first exercise they were asked to solve the task presented above (Figure 1 and 2). Having solved this they could choose between two tasks about metamodelling which were parts of the exams in the “Software Engineering” classes in the last years. After using the system they were again asked to answer a questionnaire about how they liked to work with the system.

After the first three evaluation sessions with students of this year’s “Software Engineering” class and students of the “eLearning” class the majority of the students answered the question how they liked it to work with the system with “well” or “very well”. The different types of help seem to be accepted equally well, with some students showing preferences for the one or the other kind.

6 Conclusion

Using our new hybrid approach for the underlying expert system our Intelligent Problem Solving Environment is able to (1) support problem solving with the hypothesis testing approach with respect to the ISP-DL theory and (2) give hints to correct erroneous student solution proposals. Only the correct solutions are modeled and used for the hypothesis testing. Learners’ errors are determined using the constraints which are used to give correcting hints. This can be done without having to perform studies recording the learners’ possible mistakes.

Using the same data structure containing the abstract metadata for both approaches reduces the development effort for the combined approach.

Furthermore, the system is, in contrast to the most constraint based tutors, able to work with more than one “ideal solution” for each task by using an error correcting graph matching algorithm to detect which ideal solution is most similar to the user’s proposal.

A first evaluation showed that users liked to work with the system.

7 Acknowledgements

The development of the system started within the project InPULSE[3]. The following people helped me to implement the system: Jan-Patrick Osterloh implemented the data exchange between Java and Prolog, Andreas Schäfer the Fujaba Plug-In, and Lars Weber the component displaying the completion proposal of the system (Figure 2, left side).

8 References

[ACK+95] Anderson, J. R.; Corbett, A. T.; Koedinger, K. R.; Pelletier, R.: Cognitive tutors: Lessons learned. In: The Journal of Learning Sciences, 4, 1995, p.167-207.

[AMS+09] Aleven, V.; McLaren, B.M.; Sewall, J.; Koedinger, K.R.: A New Paradigm for Intelligent Tutoring Systems: Example-Tracing Tutors. In: International Journal of Artificial Intelligence in Education, 19(2), 2009, p. 105-154.

[ASM+06] Aleven, V.; Sewall, J.; McLaren, B.M.; Koedinger, K.R.: Rapid authoring of intelligent tutors for real-world and experimental use. In: Proceedings of the 6th IEEE International Conference on Advanced Learning Technologies (ICALT 2006), Los Alamitos, CA: IEEE Computer Society,2006, p. 847-851.

[Ba07] Baghaei, N.: A Collaborative Constraint-Based Intelligent System for Learning Object-Oriented Analysis and Design Using UML. PhD Thesis, University of Canterbury, 2007

[EN11] Eden, A.H.; Nicholson, J.: Codecharts: Roadmaps and blueprints for object-oriented programs, John Wiley & Sons; 2011.

[KWR05] Kodaganallur, V.; Weitz, R.R.; Rosenthal, D.: A Comparison of Model-Tracing and Constraint-Based Intelligent Tutoring Paradigms. In: International Journal of Artificial Intelligence in Education Volume 15, Issue 2 (April 2005), p. 117-144.

[Le06] Le, N.-T.: A constraint-based assessment approach for free-form design of class diagrams using UML. In: Proceedings of the Workshop on Intelligent Tutoring Systems for Ill-Defined Domains, the 8th Conference on ITS, 2006, p. 11 – 19.

[3] http://inpulse.uni-oldenburg.de/inpulse/

[Ma01] Martin, B.: Intelligent Tutoring Systems: The Practical Implementation of Constraint-Based Modelling University of Canterbury, PhD thesis, University of Canterbury, 2001.

[MAG+03] Möbus, C.; Albers, B.; Garbe, H.; Hartmann, St.; Thole, H.J.; Yakimchuk, V.; Zurborg, J.: Towards an AI-Specification of Intelligent Distributed Learning Environments. In: KI - Zeitschrift Künstliche Intelligenz Heft 1/03 "Schwerpunkt:Lernen: Modellierung und Kommunikation", p. 19-24, Bremen: arendtap Verlag, 2003.

[MK07] Van Merrienboer, J.J.G.; Kirschner, P.A.: Ten Steps to Complex Learning: A Systematic Approach to Four-Component Instructional Design, Lawrence Erlbaum Assoc Inc, April 2007

[MKM03] Mitrovic, A.; Koedinger K.R.; Martin, B.: A Comparative Analysis of Cognitive Tutoring and Constraint-Based Modelling. In: User Modeling 2003: 9th International Conference (UM 2003), Johnstown, PA, USA, LNCS 2702, p. 313-322.

[MMS+09] Mitrovic, A.; Martin, B.; Suraweera, P.; Zakharov, K.; Milik, N.; Holland, J.; McGuigan, N.: ASPIRE: An Authoring System and Deployment Environment for Constraint-Based Tutors. In: International Journal of Artificial Intelligence in Education, 19(2), 2009, p.155-188.

[MO06a] Mitrovic, A.; Ohlsson, S.: Constraint-Based Knowledge Representation for Individualized Instruction. In: Computer Science and Information Systems (COMSIS), vol 3(1), 2006, p. 1-22.

[MO06b] Mitrovic, A.; Ohlsson, St.: A Critique of Kodaganallur, Weitz and Rosenthal, "A Comparison of Model-Tracing and Constraint-Based Intelligent Tutoring Paradigms". In: International Journal of Artificial Intelligence in Education, Volume 16, Issue 3, 2006, p. 277-289.

[RN10] Russell, St.; Norvig, P.: Artificial Intelligence: A Modern Approach, Prentice Hall, 2010.

[SMM10] Suraweera, P.; Mitrovic, A; Martin, B.: Widening the Knowledge Acquisition Bottleneck for Constraint-based Tutors. In: International Journal of Artificial Intelligence in Education,, vol. 20(2), 2010, p. 137-173.