

A web-based e-learning tool for UML class diagrams

J. Soler, I. Boada, F. Prados, J. Poch

Computer Science and Applied Mathematics Department
University of Girona
Spain

R. Fabregat

Institute of Informatics and Applications
University of Girona
Spain

Abstract— The paper presents a web-based tool designed to give support to teaching and learning of UML class diagrams. The proposed UML environment is capable to correct automatically UML class diagrams exercises providing feedback to the student immediately. The tool is part of a more general framework, denoted ACME, which provides the main functionalities of an e-learning platform. The tool has been used in a first experimental group of an introductory database course.

Keywords: *UML class diagrams, web-based tools, e-assessment*

I. INTRODUCTION

Traditionally, a database course uses entity-relationship diagrams (ERD) when teaching conceptual modeling. Describe ERD and how to use them to illustrate conceptual database design is one of the main objectives of an introductory database course [1]. But, despite the popularity of ERD, in last years, different authors have proposed the UML (Unified Modeling Language) class diagram as an alternative to be used instead of ERD notation.

The UML [2] is the standard language for modeling object-oriented applications and is commonly used in the development of software applications. Although, this methodology was developed mainly for software design, a major part of software design involves designing the databases that will be accessed by the software modules. Hence, an important part of the UML is the class diagrams. A class diagram is a type of diagram that describes the static structure of a system by showing the system's classes and the relationships between them. Therefore, it can be considered that class diagrams are similar to ERD in many ways [3]. Moreover, UML provides an advantage with respect to entity-relationship model (ER) since it is widely understood within the computing community, whereas ER is limited primarily to the database community.

In non-academic environments, UML has emerged as an effective modeling tool for database design. On the contrary, it is still rare to see UML class diagram as the primary notation when teaching conceptual data modeling [4]. In academic textbooks the primary modeling techniques are based on ER notation. However, this tendency is changing in recent versions of the books, and chapters presenting UML methodology for conceptual modeling are included [5]. There are also different books focused only on the use of UML for database design [6,7]. These books claim that UML make closer database design with the real business applications.

We agree with [8,9] that this new tendency has to be taken into account in basic database courses, and hence UML class diagram notation has to be taught as well as ERD. Since building UML class diagrams is a complex process, we decided to develop a web-based tool to support its teaching and learning. This tool has being integrated in a more general e-learning framework, denoted ACME-DB, used in database courses of our university. The ACME-DB is used as a support for teaching and learning of the main topics of the course, amongst them, entity-relationship diagrams, relational database schemas, database normalization, SQL and relational algebra. This environment also provides the functionalities required to automatically correct exercises, continuous assessment, etc. The goal of this paper is to present the UML developed tool.

The paper is organized as follows. Section 2 describes related work on UML class diagrams and presents the ACME-DB e-learning environment. Section 3 presents the proposed UML correction tool. Section 4 describes experimental results. Finally, Section 5 presents the conclusions.

II. RELATED WORK

In this section, we review the main concepts of UML class diagrams and their equivalences with ERD. Then, we present previous work on UML modeling tools and finally, the ACME-DB environment where our proposed tool has been integrated.

A. UML class diagram and ERD

There are many similarities between UML class diagrams and ERD [3]. Below we describe them by considering the examples extracted from [5] and illustrated in Fig. 1 and 2, where the same database has been modeled using UML and ERD, respectively.

The entity types in ER are modeled as classes in UML notation. An entity in ER corresponds to an object in UML. In UML class diagrams, a class is displayed as a box that includes three sections: the class name, the attributes and the operations that can be applied to these objects. Operations are not specified in ERD.

Relationship types in ERD are called associations in UML terminology, and relationship instances are called links. A binary association is represented as a line connecting the participating classes and optionally may have a name. A relationship attribute is placed in an association class that is connected to the association's line by a dashed line.

The cardinality in ER model is equivalent to multiplicity in UML terminology and denotes the number of objects that can participate in the relationships. Multiplicities are specified in the form *min..max*, and an asterisk (*) indicates no maximum limit on participation. Possible multiplicities are 0..1, 1..1 (or 1), 1..*, n..m and 0..* (or *).

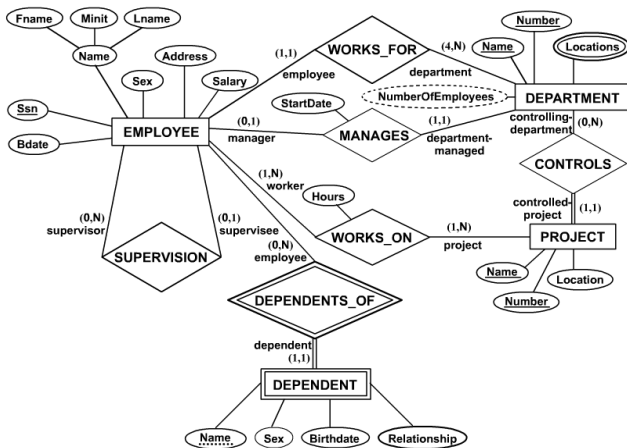


Figure 1. Entity-Relationship diagram of a company database from [5]

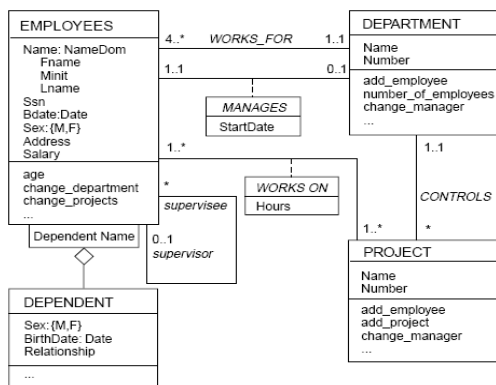


Figure 2. UML class diagram of a company database from [5]

In UML, there are two types of relationships: association and aggregation. An aggregation is meant to represent a relationship between a whole object and its components parts. In UML, it is graphically represented as a hollow diamond shape on the containing class. A composition is a stronger form of aggregation where the part is created and destroyed with the whole. The composition is drawn like the aggregation, but this time the diamond shape is filled. UML also distinguishes between unidirectional associations, which are displayed with an arrow to indicate that only one direction for accessing related objects is needed, and bidirectional that are the default. Weak entities in the ER model can be modeled using the construct called qualified association in UML that can represent both the identifying relationship and the partial key, which is placed in a box attached to the owner class. UML has the ability to specify methods, but when modeling database this feature is not needed as we only deal with data.

The UML is the standard notation for modeling business and software application needs. The main advantage of UML with respect to ER is that it can serve as a unifying framework that facilitates the integration of database design with the rest of a system design.

B. UML modeling tools

Building diagrams is a complex process and different modeling tools have been proposed for creating them. The majority of these tools has been designed for advanced users and only gives support to diagram drawing requirements. In the case of beginners, these tools are not appropriate since more advanced features are required. For instance, it is desired to obtain automatic correction of the diagram and feedback in case of detecting errors. In this section, we will focus on this second group of tools that can be very valuable in the academic context, i.e., tools that not only support drawing needs. Below we describe some of them.

Hoggarth and Lockyer [10] proposed an automated student diagram assessment system that provides a verification mechanism where the student manually compares his solution with the one designed by the teacher. At the end of the comparison process, the system generates a list with the differences with comments that can be used by the student to improve his diagram. H.Ali et al [11] proposed a UML class diagram environment that using notation extraction compares two Rational Rose files. The first file is a description of the UML class diagram proposed by the student and the second one the description of a correct solution defined by the teacher. The system compares these files line by line and generates a list with the differences. Feedback is returned according to this list. Baghaei, Mitrovic and Irwin [12] presented a constraint-based tutoring system for learning UML class diagrams. The system observes students' actions and adapts to their knowledge and learning abilities. The system compares the student solution with the ideal one proposed by the tutor. The comparison process is based on a set of rules. They proposed a single user and also a collaborative version of the tool. Virvou and Tourtoglou [13] proposed an environment for the adaptive support to a software engineering trainer. It considers two users, the trainer, considered an expert, and the student. The system assigns a degree of knowledge to the student and also an expert to monitor his work. N.Le [14] proposed an extension of the ArgoUML tool that allows creating UML class diagrams in a free-form way, i.e., with any restriction with respect to the name of the classes or the attributes. The system provides a set of guidelines to the student to create the diagram.

Despite the capabilities of these tools, in the academic context, it is desired that the e-learning tools provide not only the functionalities for which have been designed but also other functionalities related with academic issues such as continuous assessment, tracking student work, etc. Another limiting factor of these tools is that all of them requires to be installed and the majority requires an additional software for drawing the diagrams [10,11,14]. On the other hand, the tools are independent in the sense that they can not be integrated in a more general framework for giving support to different topics of database courses.

C. The ACME environment

ACME is a web-based environment developed in our university to give support at teaching and learning of different subjects. Originally, it was designed for teaching mathematics in engineering and technical degrees. ACME has a common repository of problems. The teacher can select problems from it and the system automatically generates workbooks. The student enters solutions and the system corrects them returning feedback. All the work is stored in a database and the teacher can extract information for continuous assessment. For a detailed description of the platform please see [15].

The good academic results obtained when applying ACME, encourages us to extend the platform in order to support other subjects. Currently, amongst others, it supports mathematical [15], computer programming [16], database problems, etc.

The set of modules of the platform related with database topics is denoted ACME-DB. We started to develop this environment in 2003 with the development of the relational database schemas module [17]. Since then, normalization [18], entity-relationship diagrams [19], SQL and relational algebra [20] correctors have been developed and integrated in the environment.

Compared with other e-learning platforms, ACME-DB presents, amongst others, the following advantages: (i) It integrates in a single environment, different tools for different database topics; (ii) supports automatic correction of exercises, providing feed-back and (iii) supports automatic student assessment.

Taking into account the limitations of current UML class diagrams tools and the advantages of our environment, we decided to develop the modules required to support UML class diagram correction. Such functionality will be a very valuable tool for teaching and learning conceptual database design in database courses.

III. THE UML CLASS DIAGRAM TOOL

In this section, we present the proposed tool for teaching UML class diagrams. First, we introduce the main design decisions that have been taken into account to develop the tool. Then, we describe the tool from a technical perspective, giving a detailed description of the main modules that compose it.

A. Design decisions

The proposed tool has to be capable of automatically correct UML class diagram exercises in order to reinforce teaching and learning of this topic. To design this tool, we consider that our system has to be able to assign UML class diagram exercises and also the tools required for the students entering the solution, i.e., a graphical interface that allows drawing UML class diagrams.

A second main requirement is the capability to support online correction. The system has to be able to automatically correct the exercises solved by the students. That is, when students enter a solution to a UML problem, it is desired that

the system returns immediate feedback with detailed information about errors.

The system has to allow students to enter more than one solution until the correct one is obtained or a deadline is reached. The system has to record all student work, i.e. all attempts entered until the correct one is obtained. Such information will be very valuable for the teacher to carry out a tracking the student progress. The information recorded in the system can be used for continuous assessment purposes. In addition usability requirements are also desired. For instance, the tool has to be easily accessible for teachers and students, no special installation has to be required, only a web browser. The system has to provide a communication channel that enhances student-teacher contact, whether students ask about doubts or teachers give hints about how to solve diagrams.

The ACME environment provides some of the desired requirements, such as, easy access, record of student work, continuous assessment and the student-teacher communication channel. Therefore, to reach our objective we have to focus on the functionalities not provided by the environment and required to support the correction of UML class diagram exercises.

At broad level, we can reduce our problem as it is illustrated in Fig. 3. Our system has to propose exercises, the student will enter solutions, the system will provide feedback according to the correctness of the solution, and this process will be repeated until a certain deadline or the correct solution is obtained. Hence, we have to design two main modules: the class diagram student interface and the class diagram correction module. Both are described in detail the next sections.

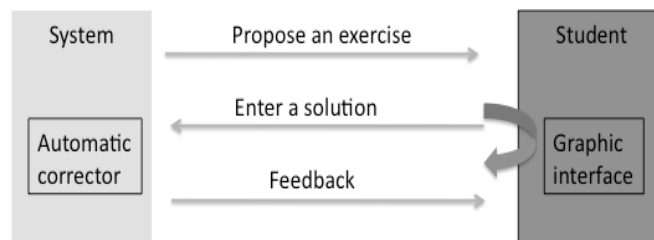


Figure 3. Main functionalities to be considered for the design of the UML class diagram correction tool.

B. The problem structure

Previous to the definition of the different modules that will compose our tool, we have to determine the structure of an UML class diagram problem. This structure is crucial for defining the correction strategy that has to be applied.

We define the problem structure following the guidelines used for the definition of the database problems that are also supported by ACME-DB. It is important to take into account that a UML class diagram problem can have more than one correct solution. The proposed structure is illustrated in Fig. 4. There is a first part with a description of the problem that has to be solved. It consists of a set of requirements for a real life application. A constraint of our approach is that attributes have

to appear in the problem descriptor with a specific name and in brackets. Although, the identification of attributes is an important skill to be acquired, for students of introductory database courses we consider more important the capability to properly group attributes in the correct class or attach to the corresponding relationships.

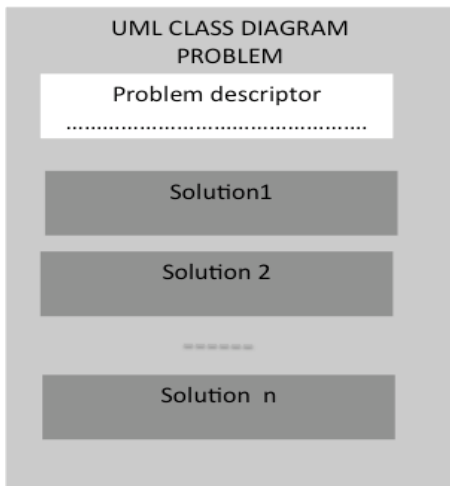


Figure 4. Main parts of a UML class diagram problem.

For each problem we also store a set of possible correct solutions. In Fig. 5, we illustrate the information codified for each solution. We have to specify all the classes and the relationships that define it. We assign one name to each one of the classes and also the list of all its attributes. In the case of association classes, we have to enter also the name of the association with which is related. For the relationships, we have to specify its name, the related classes, its multiplicity and the direction (unidirectional or bidirectional). We also have to indicate if it is an association, aggregation, composition or a generalization. In the case of qualified associations, we have to indicate the qualified class and the qualifier attribute.

```

CLASS      < class name >
ATTRIB    < a1,a2, ... >
.....

RELATION  < relation name >
CLASSES  < class1,class2 >
MULTIPLI < multiplicity > ( 0..1,1,*,1..*,n:m )
NAV       < navigation > (->,<-)
TYPE     < type > ( ASS, AGR, COM, ASQ(qualifier atrib),GEN)
.....

CLASS ASOC < class name >
ATTRIB    < a1,a2,... >
ASS       < assoc name >
.....

```

Figure 5. Parameters recorded for each correct solution

The teacher enters the problems and the solutions into the system using a specific editor integrated in the tool that makes easier this task. Each problem is recorded in an independent

file, and once it has been created the system verifies that all the parameters have been entered correctly. If the system detects that the entered file does not fit the expected structure, it generates a message indicating where is the error. Problems are stored in a common repository and once saved they are available for being assigned to the students' workbooks. For creating the workbooks the teacher selects exercises from the repository and then the generation module assigns it to the students. Once the student has assigned a workbook, he can select exercises and solve them using the module presented in the next section.

C. Class diagram interface

Before presenting the UML class diagram student interface, just to put in context, we briefly describe how the information is showed to the student. Students enter into the ACME environment by using their password. Then, the system presents a list of topics and one of them corresponds to UML class diagram exercises. The student selects this and then accesses to the list of class diagram exercises that have been assigned to his workbook. When he selects one of these exercises it appears an interface with all the information and the tools required to solve it.

In Fig. 6 we illustrate the student interface. Its main parts are: (i) At the top, the problem descriptor area that is the space where the system displays the problem descriptor. (ii) On the left, there is a button bar with the drawing tools that contains a set of buttons where all the elements required for creating a diagram are represented. (iii) In the middle, the working area where the student design the UML class diagram. In this example we show the UML class diagram designed for a database problem from [5]. Since the dimension of the class diagram can be larger than the working area, there is a zoom button to select the part of the diagram it has to be displayed in it. (iii) At the bottom, there is an area with different tabs, each one related with a diagram element. This space is designed for entering the information and the main parameters related to it. For example, when defining a class we enter the attributes, for an association the multiplicity, the type of association, etc. In this example, the association between the two highlighted classes has been selected and hence the corresponding tab allows entering the type of association, related classes, etc.

The student enters the solution and then presses the correct button. In this moment, the correction process (described in next section) starts and when it finishes it returns feedback to the student. Feedback messages appear in an independent window jointly with the sent solution. If the solution is not correct, the returned message gives some advice of how to solve it (see Table I).

D. Correction module

The correction strategy is based on a comparison process that proceeds as follow. The solution entered by the student is compared with the possible correct solutions of the problem stored in the repository of the system. If any one of the correct solutions coincides with the one of the student, the solution is considered incorrect. In this case the feedback module selects the solution of the repository more similar to the one proposed

- A company is organized into departments. Each department has a unique name (depname), a unique number (depnumber), and a particular employee who manages the department. We keep track of the start date when that employee began managing the department (startdate). A department may have several locations (location).
- A department controls a number of projects, each of which has a unique name (projname), a unique number (projnumber), and a single location.
- We store each employee's name (fname), social security number(ssn), address(address), salary (salary), sex(sex) and birth date(bdate). An employee is assigned to one department but may work on several projects, which are not necessarily controlled by the same department. We keep track of the number of hours (hours) per week that an employee works on each project. We also keep track of the direct supervisor (superssn) of each employee.
- We want to keep track of the dependents of each employee for insurance purposes. We keep each dependent's first name (depname), sex (depsex), birth date (depbdate), and relationship (depenrelationship) to the employee.

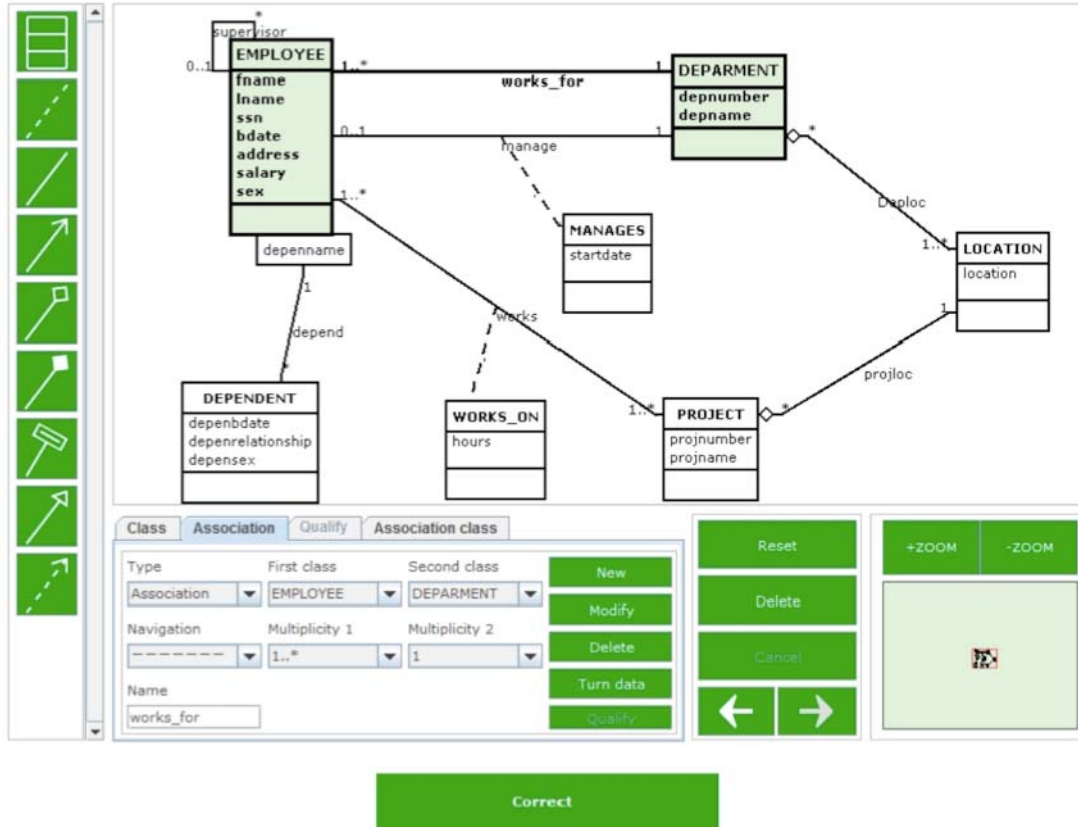


Figure 6. Student interface designed for entering a UML class diagram.

by the student and sends a feedback message to the student. The content of the message aims at driving him to the selected correct solution. The student can enter a new solution and repeat the process until the correct solution is obtained or a deadline has been reached.

The key of the correction process strategy is on the name of the attributes. Note that the only restriction it has been imposed to design the solution is that only attributes marked in brackets in the problem descriptor can be used. There is not any restriction on the names of the class nor the relationships. Classes are corrected by considering the set of attributes attached to it. Relationships are evaluated in terms of the classes they relate, its multiplicity and its type (associations, aggregations, compositions and generalization).

The differences between the correct solution and the student solution determine the message it has to be sent. We have classified the different types of errors that can be found

and we have assigned a message to each one of them. The feedback module maintains this set of possible messages and according to the type of error detected by the correction module it automatically selects the message to be sent. In Table I, we report some the message errors.

TABLE I. ERROR MESSAGES GENERATED WHEN AN INCORRECT SOLUTION IS PROPOSED.

TYPE OF ERROR	RETURNED MESSAGE
Number of classes is incorrect	More/less classes are required
Number of associations is incorrect	More/less associations are required
Incorrect class	Class named ... is not correct
Association type not correct	Association named ... is not correct
Multiplicity error	Multiplicity of association named ... is not correct
...	...

E. Integration in the ACME-DB

All presented modules have been integrated in the ACME-DB platform. The modularity of the platform allows such integration to be done easily.

The main advantage of this integration is that ACME-DB becomes a framework capable to give support to main topics of a database course. The student can use it to solve entity-relationship diagrams, relational database schemas and normalization, SQL and relational algebra queries. With the integration of the new tool, now it can also solve UML class diagrams. As far as we know, this is the unique environment that integrates all these features in a single framework.

IV. EXPERIMENTAL RESULTS

To evaluate the proposed tool we consider two groups of students of an introductory database course. The 48 students of the course were divided into two groups of 24 students. In the first one, we used the proposed tool and in the second one any tool was used. In both groups to teach UML we spent the same number of hours. The teacher introduced main UML concepts and then the same exercises were proposed to the students to acquire practice. These exercises require for being solved UML class diagrams with a number of classes that range from four to eight and from six to twelve associations.

In the first group (group A), the teacher introduced some example exercises through the proposed UML tool. Then, a personalized workbook with four different UML exercises was assigned to each student. In the second group (group B), the teacher presented the same examples of group A, but without using the tool. The students solved the exercises by hand. The teacher encouraged students to solve all the exercises and in case of questions, to go to his office.

For each group we collected the number of student and the number of exercise that they solve correctly. The obtained results are reported in Table II. Note that the best results are obtained using the tool.

TABLE II. NUMBER OF STUDENTS AND NUMBER OF PROBLEMS CORRECTLY SOLVED

Number of problems	4	3	2	1	0
Group A	16	4	2	1	1
Group B	10	2	4	4	4

We also evaluate the advantages of the tool from the teacher’s point of view. The tool provides information of the type of student’s errors. For instance, we detected that the 34,2% of errors are multiplicity errors, 31,3% are association type errors, 17,5% are definition class errors and 17,0% other errors. We also detect that to solve a problem correctly the student requires in average 3,2 attempts.

At the end of UML sessions, students of both groups have to pass an exam. The exam consists in one exercise similar to the exercises assigned to the students. The exam was the same for both groups. In Table III, we illustrate the obtained results, where capitals from A to D represent from excellent to failing the exam, respectively. Although, these results are

experimental and need further evaluation with more students, we detect that the students that used the tool obtained better results than the other ones.

TABLE III. EXAM RESULTS

Mark	A	B	C	D
Group A	12	6	3	3
Group B	10	4	6	4

From the teacher’s first impressions, we can remark that the environment is easy to use. It does not require any installation, only a web browser. More importantly, it provides gains with respect to the classical teaching methodology in the sense that it offers a system for the continuous assessment of the student’s progress, makes personalized attention to the student easier and assesses the degree of participation of the students. The students’ impressions have also been positive especially the fact that to access the system they only need an internet connection. During the different sessions students were asked to comment on the problems they faced while using the system. The responses were very positive. The students feel motivated to solve the proposed problems. The possibility to correct a problem in real time encourages them to work until a correct solution is found.

V. CONCLUSIONS

We have presented a UML class diagram tool that provides support at teaching and learning of this topic. This tool has been integrated in the ACME-DB, an e-learning framework for database topics. The tool has been used in a first experimental group with very satisfactory results.

REFERENCES

- [1] M. Robbert and C.M. Ricardo, “Trends in the evolution of the database curriculum,” in Proc. 8th Conf. on Innovation and technology in computer science education ITiCSE, pp 139-143, 2003.
- [2] I. Jacobson, G. Book, and J. Rumbaugh, The unified software development process, Reading, Addison Wesley Longman, 1999.
- [3] J. Rumbaugh, “ER is UML,” J. Inform. Syst. Educ., vol. 17, no. 1, pp. 21-26, 2006.
- [4] J. Suleiman and M.J. Garfield, “Conceptual Data Modeling in the Introductory Database Course: Is it Time for UML?,” J. Inform. Syst. Educ., vol. 17, no. 1, pp. 93-99, 2006.
- [5] R. Elmasri and S.B. Navathe, Fundamentals of Database Systems, 5th Edition. Addison Wesley, 2007.
- [6] J. Naiburg and R.A. Maksimchuk, UML for database design, Addison-Wesley, 2001.
- [7] R.B. Muller, Database Design for Smarties: Using UML for Data Modeling, Morgan Kaufmann Publishers Inc., 2003.
- [8] T.A. Carte, J. Jasperson, and M.E. Cornelius, “Integrating ERD and UML concepts when teaching data modeling,” J. Inform. Syst. Educ., vol. 17, no. 1, pp. 55-63, 2006.
- [9] D. Golden and V. Matos, “Introducing the Unified Modeling Language into the Information Systems Curriculum,” J. Inform. Syst. Educ., vol. 17, no. 1, pp. 83-92, 2006.

- [10] G. Hoggarth and M. Lockyer, "An Automated Student Diagram Assessment system," in Proc. 6th Conf. on Innovation and technology in computer science education ITiCSE, pp 122-124, 1998.
- [11] N.H. Ali, Z. Shukur, and S. Idris, "Assessment System For UML Class Diagram Using Notations Extraction," Int. J. of Comput. Sci. and Network Security, vol.7, no. 8, pp. 181-187, 2007.
- [12] N. Baghaei, A. Mitrovic and W. Irwin, "Supporting Collaborative learning and problem-solving in a constraint-based CSCL environment for UML class diagrams," in Int. J. Computer-Supported Collaborative Learning, vol 2, pp 159-190, 2007.
- [13] M. Virvou and K. Tourtoglou, "An adaptive Training Environment for UML," in Proc. 6th Int. Conf. on Advanced Learning Technologies ICALT, 2006.
- [14] N. Le, "A Constraid-based Assesmnet Approach for FreeForm Design of Class Diagrams using UML," in Proc. 6th Int. Conf. Intelligent Tutor Systems, ITS, 2006.
- [15] J. Soler, J. Poch, E. Barrabés, D. Juher, and J. Ripoll, "A tool for the continuous assessment and improvement of the student's skills in a mathematics course" in Proc. Int. Symp. Technology of Information and Communication in Education for engineering and industry, TICE, pp. 105-110, 2002.
- [16] I. Boada, J. Soler, F. Prados and J. Poch, "A teaching/learning support tool for introductory programming courses" in Proc. Int. Conf. Information Technology based higher Education and Training, pp. 604-609, 2004.
- [17] F. Prados, I. Boada, J. Soler, and J. Poch, "An automatic correction tool for relational database schemas," in Proc. Int. Conf. Information Technology based higher Education and Training, S3C pp 9-14, 2005.
- [18] J. Soler, I. Boada, F. Prados, and J. Poch, "A web-based Problem-Solving Environment for database Normalization" in Proc.8th Int. Symp. on Computers in Education SIIE, 2006.
- [19] F. Prados, I. Boada, J. Soler, and J. Poch, "A web based-tool for Entity-Relationship Modeling," in Proc. Int. Conf. Computacional Science and its Applications ICCSA 2006, LNCS 3980, pp. 364-372, 2006.
- [20] J. Soler, I. Boada, F. Prados, J. Poch, and R. Fabregat, "An automatic correction tool for relational Algebra Queries," in Proc. Int. Conf. Computational Science and its Applications ICCSA, LNCS 4706, pp 861-872, 2007.