TextOO: An Object-Oriented Learning Tool Based on Enunciates

Carlos A. Lázaro Carrascosa, J. Ángel Velázquez Iturbide, Raquel Hijón Neira, Isidoro Hernán Losada

Departamento de Lenguajes y Sistemas Informáticos
Universidad Rey Juan Carlos, Madrid
C/ Tulipán s/n
28933 Móstoles, Madrid, España
{carlos.lazaro,angel.velazquez,raquel.hijon,isidoro.hernan}@urjc.es

Abstract. This work describes an educational tool, named TextOO, to assist in the application of object-oriented concepts to design. Depending on the nature and difficulty of the problem, it supports the application, analysis or design level of Bloom's taxonomy. TextOO is based on the use of natural language specifications, that we call "enunciates". It allows students to select parts of an enunciate and to associate them to program elements (classes, objects, messages, etc.). It supports class, object and sequence diagrams. Assessment is also supported, giving feedback to students about their good and bad choices. We evaluated TextOO with our students, yielding good results in two issues: they performed better in simple modelling exercises and they considered the tool useful and easy to use.

1 Introduction

The object-oriented programming (OOP) paradigm has become the most important for software development. It is dominant in computing companies, and also in computer science education. Computer science faculty needs new pedagogic approaches to teach OOP effectively. It is a challenging problem which can be faced from many different points of view. There is not even consensus about whether OOP must be taught in the first years or it is better to teach it after structured programming is well known.

We consider that it is necessary the use of a pedagogic framework to measure the degree of learning achieved by students or, at least, to estimate it. We have adopted Bloom's taxonomy [1], which is based on six levels of knowledge (knowledge, comprehension, application, analysis, synthesis and evaluation). Applying the taxonomy to programming is not trivial [5], but it provides a general framework.

We have developed several tools to assist in OOP teaching at different levels. A first application was intended to assist in learning inheritance at the knowledge and comprehension levels; its definition and evaluation can be consulted elsewhere [4].

Other applications have just been developed to assist in understanding object creation; they are inspired in the "problettes" of Amruth Kumar (e.g. [7]).

Achieving higher levels on OOP in Bloom's taxonomy is difficult if we confine ourselves to simple problems on implementation details. Thus, we studied the possibility of moving to a complementary approach, i.e. modelling. Based on this new approach, we have developed the educational tool TextOO. It intends to assist students in OO modelling, as well as in assessing their models.

The structure of the paper is as follows. The second section describes different aspects of TextOO: interaction, pedagogical goals, correcting features, and several technical issues and availability. The third section describes our experience with the tool, including an experimental evaluation performed with our students. Finally, sections 4 and 5 report on related works, our conclusions and future work.

2 TextOO

TextOO is a tool designed to assist in OO modelling. The student is given a specification and he/she must model an OO design to represent the specified problem. We assume that the student knows basic OO concepts (encapsulation, inheritance, etc.).

Traditionally, designs are crafted very informally, without making explicit the relationship of its elements to the specification elements. We adopt the opposite approach, consisting in making explicit such relationship. Specifications (and explanations) of the teacher are given in natural language. In the rest of the paper, we use the term "enunciates" to refer to specifications in natural language.

TextOO delivers a static OO design. It can be used as documentation but it neither can be translated into Java nor executed.

2.1 Interaction with TextOO

The tool supports two different roles. From the teacher's view, the tool gives the possibility of editing his/her own enunciates and diagrams. The latter can be enriched with comments annotated in its different graphical elements.

From the student's view, the tool provides him/her with a written enunciate of a problem and he/she must model it with OOP. The student must select all the words in the enunciate relevant to him/her, and link them to graphical representations of OO concepts in a UML diagram. If necessary, new graphical elements can be created. Each student association is explicitly displayed. UML syntax is forced by means of a constraint user interaction.

Figure 1 shows a snapshot of TextOO. Notice that the window is split into two parts. In the upper part, an enunciate is shown, as well as general menu options. In the lower part, three different views can be selected: class diagrams, object diagrams and sequence diagrams. In this example, the words "figures" and "pictures" are in the first line of the enunciate. The class diagram also contains at the top classes "Figure" and "Picture" because the student created and associated them to their first occurrences in the enunciate.

These views correspond to three groups of concepts in a typical enunciate:

- Concepts associated to classes. They describe the static part of the problem.
- Concepts associated to objects. They describe the static part of an example.
- Concepts associated to execution. They describe the dynamic part of an example.

The selection of one view leads to a different UML diagram, with a different set of buttons for visual edition:

- Class diagram. The user may select class, attribute, method or relationship (inheritance, association or composition) (see Figure 1).
- Object diagram. The user may select object, attribute value or relationship.
- Sequence diagram. The user may select object or message.

The procedure to create a new graphical element is simple: the user selects a term in the enunciate and then presses a button; a dialog is opened for additional information, e.g. the class where a method will be integrated. Code-specific information requested in these dialogs is then displayed using UML syntax. Terms selected in the enunciate are highlighted in different colors: classes in red, methods in green, properties in blue, objects in pink, and property values in light green.

We want to notice that class and object relationships and messages often require a different treatment. Typically, they do not appear at the enunciate, so they are created in their corresponding view without being requested any reference to the enunciate.

User interaction is enriched with a number of mouse operations performed over the graphical elements of each view. They allow deleting and modifying graphical elements, as well as performing other graphical operations such as grouping.

TextOO allows drawing UML diagrams without using enunciates. This feature is necessary to create graphical elements not present in the enunciate. It also is a complementary facility that allows using TextOO as a visual editor.

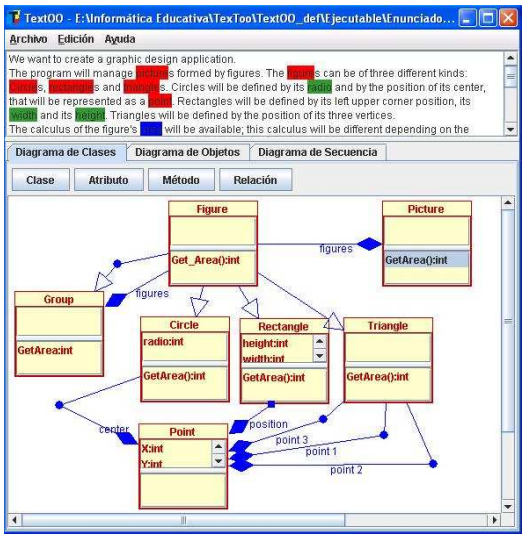

Fig. 1. Class diagram view of Textoo

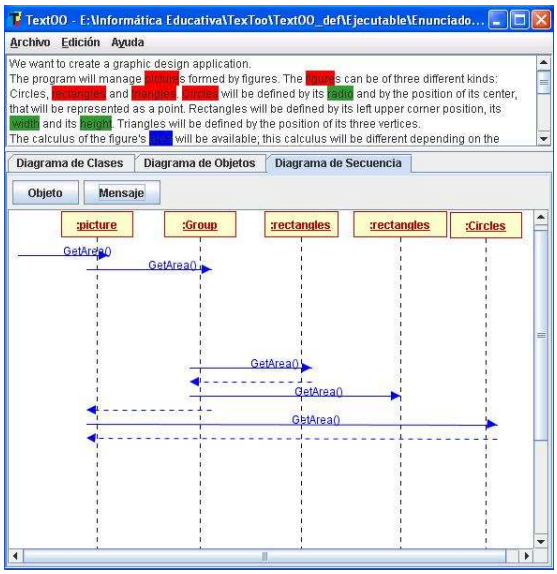

Fig. 2. Sequence diagram view of Textoo

2.2 Pedagogical Goals

TextOO was designed for simple modelling tasks in courses on object-oriented programming and, to a lesser extent, on software engineering. The student know, understand and apply OOP basic concepts to design, and well as being able to distinguish the static part of an OO program (class, properties and methods) and its dynamic part (objects, states and messages).

In terms of Bloom's taxonomy, TextOO assumes that students know and understand OOP basic concepts. In addition, TextOO intends to assist in the application, analysis and synthesis levels. Analysis is supported, given that students may add, delete and modify elements in a design, decomposing it in different ways and experimenting with alternatives. The application level is supported for simple problems, where it is easy to translate the enunciate to a design. For more complex enunciates, the student has to make personal choices, as corresponds to the synthesis level.

2.3 Evaluation of OO Designs

The student may consult at any moment the teacher's solution. The solution cannot be modified and may include comments distributed over the graphical elements of the three views, available by means of context menus.

When the student considers that his/her work is finished, he/she may validate his/her proposal. TextOO provides a rigid assessing component which proceeds validating the solution in three phases:

- Enunciate. It checks whether the terms selected by the student and the teacher match, both in position and type.
- Elements. It checks whether the graphical elements defined by the student and the teacher match. Elements considered include classes, attributes and methods (in the class diagram), objects and attribute values (in the object diagram), and sequence objects (in the sequence diagram). It checks their element type and the element they belong to (in the case of attributes, methods and attribute values).
- Relations. It checks whether the relations defined by the student and the teacher match. Relations considered include class and object relations, as well as messages. It is checked their source and target element, name and type.

TextOO gives a report in tabular format to students, differentiating the three phases. For each phase, three results are given: good choices (when the student and the teacher decision match), differences (when they do not match), and omissions (when the teacher makes decisions that the student does not). For each result and phase, the student may require more detail. A button allows him/her examining a table of differences.

2. 4. Technical Issues and Availability

TextOO was developed using the Java language. User interaction is performed using the JSD API (http://vido.escet.urjc.es/JSD). Enunciates and diagram information are stored as XML files using XML-Schemes we developed.

TextOO has been published as open source software under GPL licence, and it is available at the following web site: http://vido.escet.urjc.es/textoo. There are two invited passwords: 'p' for teachers (for the Spanish word "profesor"), and 'a' for students (for the Spanish word "alumno", student). (If there is interest enough from other universities, we will consider translating the user interface into English.)

3 Experience and Evaluation

We have performed a preliminary evaluation of the tool in the course "Software Engineering I", offered in the fourth year of the major in Computer Science offered by our university. The subjects were students of two groups. A previous test on simple concepts of software engineering was carried out in order to check the randomness of both groups.

Afterwards, a classical statement of object-oriented modelling (describing a geometrical problem) was given. Students had to model it using class, object and sequence diagrams. Students in the control group solved the problem either using Rational (13 students) or without any computer tool (15 students). Students in the experimental group solved the problem using TextOO (7 students).

Table 1 shows the scores obtained in the post-test. Values range between 0 (minimum value) and 3 (maximum value). Each cell contains the corresponding number of

answers. The average score for the group using no tool is 2; for the group using Rational Rose, 1.77, and for the group using TextOO, 2.21. Consequently, students using TextOO outperformed students in the control group. We then applied the independent-samples T test by grouping data in three different ways (TextOO vs. no TextOO, TextOO vs. Racional Rose, and TextOO vs. no tool). Unfortunately, the analysis reveals that results are not statistically relevant.

Table 1. Scores of the post-test

Score	No tool	Rational	TextOO
0	1	0	0
0.5	0	2	0
1	0	0	0
1.5	3	5	1
2	6	1	3
2.5	3	5	2
3	2	0	1

Finally, students who used TextOO filled a questionnaire about its usefulness and usability. Table 2 contain the questions and the answers in a scale ranging between 1 (very bad) to 5 (very good).

Table 2. The questionnaire

	Question	Average	Standard deviation
Q1	The tool is easy to use	3.71	0.49
Q2	Having available the statement on screen while I modelled the problem was useful	4.29	0.76
Q3	Interacting with the enunciate is useful	3.71	0.95
Q4	The corrector is useful	4.00	0.82
Q5	In general, I think that the tool may contribute to better understanding modeling concepts	3.86	0.69

In general, the students' reaction was very positive. In particular, we want to remark the results in question 2, given that it is one of the key elements of the design of our application.

4 Related Work

We analyzed different tools somehow related to our approach. In the first place, we found several related educational programming environments. Some environments also provide visual design (e.g. BlueJ [11] or FUJABA Life [9]). TextOO also works with design diagrams, but it does not generate code from diagrams. However, it offers some implementation facilities, like assisting in property and method syntax.

Other programming environments visualize code execution (e.g. Jeliot 2003 [8] or JGrasp [2]). TextOO is a modelling tool, so it provides visualizations of the design, not of the code. With respect to execution, TextOO statically visualizes case execution based on the examples information provided in the enunciate.

A second group of related tools are professional software engineering tools. A first difficulty using these tools lies in their difficulty of use. They also support the relationship between design and implementation, connecting the changes produced in both views. However, they lack facilities to make learning explicit. For instance, Visual-Paradigm [12] has a textual analysis functionality that allows direct interaction with enunciates. However, it does not consider relationships among the different components of enunciates. Other well known tools (e.g. IBM Rational Rose) also lack this support.

Some educational software engineering tools have also been developed to assist students in the use of other methodologies. An example is ECoDe, a tool that supports the use of CRC cards [3].

5. Conclusions and Future Work

We have described TextOO, an educational tool to assist students in the application of basic object-oriented concepts to object-oriented modelling. The tool is useful for OOP courses and, to a lesser extent, on software engineering. It uses natural language specifications to guide design. Support is given to class, object and sequence diagrams. TextOO also provides automatic assessment of students' design by comparing their designs and design-enunciate relationships to the solution provided by the teacher.

Depending on the nature and difficulty of the problem, the student is placed on the application, analysis or design level of Bloom's taxonomy. We evaluated TextOO with our students, yielding good but no statistically relevant results. The students also answered a questionnaire about its usefulness, yielding a general feeling of usefulness of the tool.

These results must be considered with care, because the evaluation was not performed over the most adequate group of students. We believe that the tool can be useful for software engineering students, but it is mainly designed for novice object-oriented programming students. We intend to evaluate TextOO with OOP students as soon as the scholar schedule allows us.

We are working on several improvements. Firstly, teachers in design courses at our university plan to apply TextOO to a collection of solved problems in order to obtain further feedback on strengths and drawbacks of the tool. In particular, we want feedback on the limitations of the tool for problems with a large enunciate.

Secondly, we want to add extra functionalities to the tool. We consider very useful to include more OOP elements (both at the design and implementation levels), as well as generating code.

Finally, we want to make more flexible the assessment facility. The assessment mechanism can be improved in three directions: considering syntax and semantics of enunciates, extending the display of the teacher's solutions (adding interactivity to the tool), and giving more flexibility to the corrector (read e.g., [6][10]).

Acknowledgments

This work has been supported by project TIN2004-07568 of the Spanish Ministry of Education and Science. Thanks to Francisco Gortázar-Bellas and Micael Gallego-Carrillo for their technical support. Special thanks to María Cristina Garrigós Fernández for her work and dedication.

References

[1] Bloom, B., and Krathwohl, D. R. *Taxonomy of Educational Objectives: Handbook I, The Cognitive Domain*. Addison-Wesley: New York, 1956.

[2] Cross II, J.H., et al. Using the debugger as an integral part of teaching CS1. In *Proc. 32th ASEE/IEEE Frontiers in Education Conf. (FIE 2002)*, 2002.

[3] Gray, K.A., Guzdial, M., and Rugaber, S. Extending CRC cards into a complete design process. In *Proc. 8th Annual Conf. on Innovation and Technology in Computer Science Education (ITiCSE 2003)*, ACM Press, 2003, 226.

[4] Hernán-Losada, I., Lázaro-Carrascosa, C., and Velázquez-Iturbide, J.Á. Una aplicación educativa basada en la jerarquía de Bloom para el aprendizaje de la herencia de POO. In *Proc. VII Simposio Internacional de Informática Educativa (SIIE 2005)*, 2005.

[5] Hernán-Losada, I., Lázaro-Carrascosa, C., and Velázquez-Iturbide, J.Á. On the use of Bloom's taxonomy as a basis to design educational software on programming. In *Proc. World Conf. on Engineering and Technology Education (WCETE 2004)*, 2004, 351-355.

[6] Hoggarth, G., and Lockyer, M. An automated student diagram assessment system. In *Proc. 3rd Annual Conf. on Innovation and Technology in Computer Science Education (ITiCSE'98)*, ACM Press, 1998, 122-124.

[7] Kostadinov, R., and Kumar, A. A tutor for learning encapsulation in C++ classes. In *Proc. World Conf. on Educational Multimedia, Hypermedia and Telecommunications (ED-MEDIA 2003)*, AACE Press.

[8] Moreno, A. et al. Visualizing programs with Jeliot 3. In *Proc. International Working Conf. on Advanced Visual Interfaces (AVI 2004)*, ACM Press, 2004, 273-276.

[9] Nickel et al. The FUJABA environment. In *Proc. 22nd International Conf. on Software Engineering*, ACM Press, 2000, 742-745.

[10] Thomas, P., Waugh, K., and Smith, N. Experiments in the automatic marking of E-R diagrams. In *Proc. 10th Annual Conf. on Innovation and Technology in Computer Science Education (ITiCSE 2005)*, ACM Press, 2005, 158-162.

[11] Van Haaster, K., and Hagan, D. Teaching and learning with BlueJ: An evaluation of a pedagogical tool. In *Information Science + Information Technology Education Joint Conf.*, Rockhampton, QLD, Australia, 2004.

[12] Visual Paradigm Co. *Visual Paradigm for the UML 2.0 User's Guide*.