Design of Pedagogical Feedbacks in a Learning Environment for Object-Oriented Modeling

Dominique Py, Mathilde Alonso, Ludovic Auxepaules, and Thierry Lemeunier

Laboratoire d'Informatique de l'Université du Maine (LIUM), Avenue Laënnec, Le Mans, France
{Dominique.Py, Mathilde.Alonso, Ludovic.Auxepaules, Thierry.Lemeunier}@lium.univ-lemans.fr

Abstract. This paper describes the design of the pedagogical feedbacks in Diagram, a learning environment for UML object-oriented modeling. The project is rooted in several years of teaching practice and actual diagrams produced by novice students. Diagram is based upon an interaction framework that supports the learner's metacognitive activity. It includes a diagnosis module which compares the student diagram with a reference diagram, and produces the list of the structural differences between these diagrams. We show that these structural differences can be mapped onto differences that are relevant from a pedagogical viewpoint. The differences that are noticed in the student diagram give rise to feedbacks of three kinds: indications, questions and suggestions. We illustrate the feedback elaboration process by giving an example of a student diagram with the diagnosis results and the messages generated by the learning environment.

Keywords: modeling, UML, interaction, feedback, diagnosis, metacognition

Introduction

The conception of models, a task which represents an important part of the scientific activity, has been deeply modified in the last decades by the development of new software tools. In the same time, the scientific curricula have given more and more place to the modeling activity, and research works in cognitive sciences have proven that using software tools for modeling could contribute to the learning processes [1].

In software engineering, modeling is a critical part of the analysis step of a software system development process. Nowadays, modeling is core topic in most Computer Science and Software Engineering curricula, especially since the object-oriented modeling language UML (Unified Modeling Language) has been accepted throughout the software industry as the standard graphical language for specifying software systems. Recent research works have focussed on the design of computer-supported learning environments for modeling, especially for object-oriented modeling [2], [3] and database modeling [4]. In these environments, the relevance of the pedagogical feedbacks relies on the system's ability to evaluate the correctness of the student's work. Due to the lack of deterministic methods for elaborating a solution or checking the consistency of a model, performing this evaluation is still a difficult problem.

The work presented in this paper is part of a project of the LIUM laboratory, which aims at designing models, methods and tools for computer-supported learning environments in the area of object-oriented modeling. In this context, Diagram, a learning environment for UML class diagrams, is being developed and tested in real situations [5]. Diagram is designed like a class diagram editor that provides specific interaction modes and help features for novice users. The environment is open: there is no predefined exercise base, the teacher can define his/her own exercises by giving the problem's statement and a solution diagram. The problem we consider here is to provide the students with feedbacks (in natural language) which take into account the correctness of his/her diagram. The approach we have adopted consists in comparing the student's diagram and the reference diagram, and trying to match them. The structural or content differences that are identified by the comparison module are then analysed at the pedagogical level in order to produce appropriate feedbacks. This approach relies on the assumption that the differences between the two diagrams do not necessarily mean that an error has been done, but they suggest that the learner may have omitted an element of the diagram, or misrepresented a concept. The feedback messages displayed by Diagram aim at drawing the learner's attention to some part of the diagram and drive him/her to check its correctness.

1. Metacognitive Aspects of the Modeling Task

The modeling task is not a well-defined process, as the task is open-ended. There are no deterministic methods to design a UML diagram or to check its appropriateness to a textual description. This is the reason why we focus on the acquisition by the learner of metacognitive skills like procedures of control, correction and validation of his/her productions.

According to Flavell [6], metacognition consists of both metacognitive knowledge and metacognitive experiences or regulation. Metacognitive knowledge refers to acquired knowledge about cognitive processes. Metacognitive regulations are processes that one uses to control cognitive activities. These processes help to regulate and oversee learning, and involve *planning*, *monitoring* and *checking* the outcomes of those activities [7]. We claim that an efficient computer-supported learning environment for UML modeling should support both reflective and metacognitive activity according to these three components of metacognitive regulation. The interaction in Diagram has been designed so as to encourage this metacognitive activity throughout the sessions.

2 The Interaction Scenario in the Diagram Environment

The teaching of object-oriented design in university course consists of theoretical and practical sessions. During these practical works, students use professional UML tools, which have not been designed for initiation purposes: they are generally complex, they include unnecessary features and they provide few help for novice users who are just beginning to learn UML modeling. Taking these aspects into consideration, we designed Diagram like a class diagram editor that provides a subset of classic editor functions, specific interaction modes and help features for novice users [5]. This environment is intended to be used during practical sessions, with a human teacher who supervises the group and intervenes when a student encounters too many difficulties.

Diagram gives the opportunity to work simultaneously on the problem text and the class diagram, which facilitates the visual control of the current modeling task. This feature, missing in classic UML tools, is available in some learning environments, like COLLECT-UML [2]. It gives greater opportunities for interaction because, even though the text cannot be modified, it is possible to change its appearance.

Moreover, Diagram reifies a pedagogical method for solving UML modeling problems. This method has been designed by the fourth author, who teaches UML at the Université du Maine, and uses it with his students. This method follows three steps: the first step consists in reading and understanding the text of the problem, the second step consists in designing the diagram, and the third step consists in reading again the whole text in order to check the diagram's correctness. This organization supports both *planning* and *checking* functions of metacognitive regulation. This method has been reified into Diagram by means of specific graphical tools.

During the first step, the text alone is displayed on the interface and the student discovers the general subject of the text requirements. When they work with pencil and paper, some students use a pen to highlight the words which seem to be relevant to the future diagram. An underlining function has been added to Diagram so as to provide a similar tool.

The second step consists in creating the class diagram in relation to the given requirements. Diagram allows to draw a component of the diagram (class, attribute or relationship) by using a word or a phrase from the text problem. The highlighted expression and the UML element are displayed in the same color, in order to facilitate the visual control (Fig. 1). This modality of interaction is an original concept in Diagram and aims at strengthening the link between the text requirements and the diagram. It is also possible to draw a "free" element of the diagram, i.e. without textual link (in order to model an implicit relationship, for example). Any free element can be linked to a textual expression and any textual expression can be linked to another expression of the text. Then, the free element or expression takes on the same color as the expression to which it is linked.

This step includes different contextual helps that favour the learner's control on his/her own activity. For example, the interface provides textual reformulations of the diagram's parts. When the mouse pointer goes over an element (class or relationship), a tooltip explaining its semantic is displayed. This device has been tested with students, and we showed that it improves the user's self-correction ability [5].

During the last step, the student must read again the text problem and compare it to his/her class diagram in order to check the diagram correctness and completeness in relation to the text. At the beginning of this step, the diagram is hidden and the text alone is displayed in black and white on the interface. When the student moves the mouse over an expression linked to an element of the diagram, this expression recovers its color and at the same time, the corresponding element in the diagram reappears on the interface.

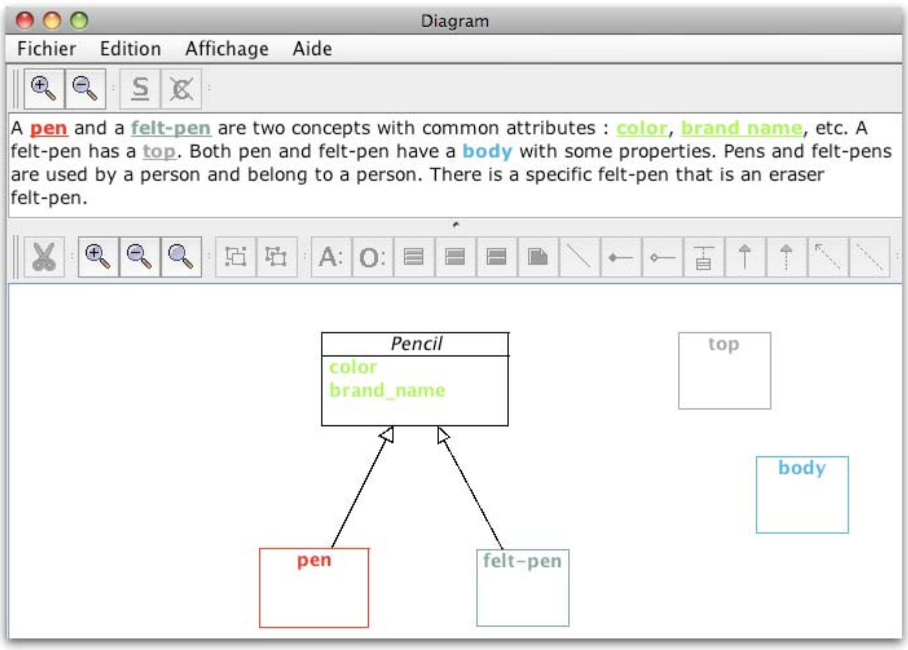
A pen and a felt-pen are two concepts with common attributes : color, brand name, etc. A felt-pen has a top. Both pen and felt-pen have a body with some properties. Pens and felt-pens are used by a person and belong to a person. There is a specific felt-pen that is an eraser felt-pen.

Fig. 1. Interface of Diagram during the design step

The generic helps described above do not take into account the correctness of the student’s diagram. Therefore, a diagnosis tool has been developed as a part of the Diagram project. It analyses the learner’s solution and gives a basis for more specific feedbacks, depending on the diagram’s features.

3 The Diagnosis Module

Basically, the diagnosis method in Diagram compares the student’s diagram with an expert’s diagram (provided by the teacher), and enumerates the differences between the two diagrams. We use a reference diagram because of the lack of automated solver in the domain. The limit of this approach is that it is restricted to simple diagrams, where variations are not numerous, i.e. problems for novice students. This method could not be directly extended to more complex problems, where structurally different solutions exist.

Due to space considerations, we only briefly sketch the diagnosis algorithm. A more detailed description can be found in [8].

In order to automate comparison and matching, we consider class diagrams as graphs, where the vertices are classifiers and the edges are relationships. Thus, the problem of diagram matching can be seen as a variant of graph matching, with graphs characterized by specific features.

We introduce characteristic structures called *patterns* that correspond to different granularity levels. The simple patterns (classifiers, relationships, features) represent the elements of the meta-model. These patterns are organized and structured into complex patterns. The complex patterns, respectively Association sequence and Generalization hierarchy, are built from several simple patterns and a relationship type, respectively association and generalization. These patterns have been introduced because of their interesting structural and semantic properties that can be exploited by the matching algorithm in order to match diagram parts at a higher granularity level.

Sorlin, Solnon and Jolion [9] propose a generic similarity measure for graph matching. This measure is parameterized by similarity functions that express domain dependent similarity knowledge and constraints. The diagnostic algorithm in Diagram relies on the same principles. Similarity functions compute a score for each couple of patterns to be compared. The diagram matching algorithm proceeds top-down. Complex patterns, then simple ones, are compared and the pairs are ordered by decreasing score. The diagram matching algorithm produces pairs of fully or partially matched patterns. Partial matches are labeled with the type of the difference, according to the Structural Differences Taxonomy (SDT) below.

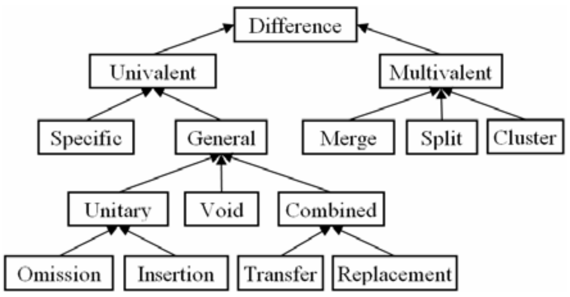

Fig. 2. Structural differences taxonomy (SDT)

An univalent difference occurs when two single patterns match. Among these differences, the specific ones are related to pattern properties and semantic. Specific differences for simple pattern concern the name, visibility, abstraction, aggregation, relationship orientation and so on. Specific differences for generalization hierarchies concern the propagation of properties and relationships from mother class to child class and vice versa. General differences describe the patterns organization in the diagrams. The two basic ones are the Omission and the Insertion of an element. These differences can be combined and form more complex ones, like Transfer and Replacement (one omission plus one insertion).

A multivalent difference occurs when a single pattern matches a group of patterns. Three cases can occur: a single pattern in a diagram may match several patterns in the other diagram (Split) or vice versa (Merge), and a group of patterns in the first diagram may match another group in the second one (Cluster).

The diagnosis algorithm is implemented in Java. UML2 (an Eclipse Tools sub-project) of Eclipse modeling project has been used to represent the diagrams. A preliminary evaluation has been conducted with students' diagrams (collected during the previous experiments with Diagram) in order to assess the robustness and speed of the diagnosis method [10]. The results show that the diagnosis quality is quite good for simple and medium problems, and remains correct on complex problems. On the whole diagram base (n=82), 90% of the matches are relevant at 85% or more. The calculus time depends on the diagram's size: it varies between 0.2 and 4 seconds, in function of the diagram complexity, which appears to be fast enough to provide pedagogical feedbacks online.

4 Pedagogical Feedbacks

The diagnosis component produces a list of differences between the student's diagram and the reference diagram, according to the structural differences taxonomy. These differences cannot be directly used to provide feedback. We need to convert them into differences according to a second taxonomy, designed for pedagogical purpose (Pedagogical Differences Taxonomy, or PDT). Once the conversion is completed, a feedback message can be elaborated to handle each difference.

4.1 Pedagogical Differences Taxonomy

We designed the pedagogical differences taxonomy according to the following method. At first, we collected diagrams built by learners during the experiment studies of Diagram in autumn 2006 and in March 2007 [5]. Then, we chose the two problems for which we had obtained the greatest number of diagrams (26 and 30 diagrams). These two problems appeared to be representative of our base, with regard to the difficulty and the number of elements in the diagram. For each problem, we compared the learners' diagrams with a reference diagram, provided by the teacher, and we established a list of elementary differences that can be found between two diagrams. Each difference is associated with a feedback message.

The pedagogical differences are classified in eight categories:

- Omission of an element: an element of the reference diagram has been omitted by the learner.
- Addition of an element: the learner added an element which does not appear in the reference diagram.
- Transfer of an element: an element has been transferred to another part of the diagram. For example, a relationship between two classes A and B in reference diagram is shifted between classes A and C.
- Duplication of an element: an element of the reference diagram is replaced by several elements of the same type in the learner's diagram.
- Merging elements: several elements of the same type are substituted by a single element.

- Misrepresentation: an element of the reference diagram is changed to another one, of a different type. For example, a class is replaced by an attribute or a composition is replaced by an association, and so on.
- Reversion of the direction of a relationship: an oriented relationship (inheritance, aggregation or composition) has been reversed by the learner.
- Wrong multiplicity: multiplicities of a relationship in the learner's diagram are different from those in the reference diagram.

We observed in students' diagrams that some differences often go with other ones. For example, a "class omission" usually implies a "relation omission" or a "relation transfer", because the relations supported by the missing class are themselves moved or omitted. When this situation happens, it is better to produce a single feedback covering the group of differences than to produce separate feedbacks. Therefore, we defined a set of fifteen compound differences that can be treated as a whole. A compound difference is made of a main difference and a set of secondary differences. For example, the compound difference "relationship duplication and transfer" is composed of a main difference "relationship duplication" and one or several secondary differences amongst "relationship transfer", "direction reversion", "wrong type of relationship" and "wrong multiplicity".

Once all the single differences have been listed in a student's diagram, the compound differences can be searched for. The compound differences have priority on simple differences and lead to specific feedback during the interaction. The simple differences which occur separately produce the usual feedback.

4.2 Correspondence between Taxonomies

The diagnosis module produces a list of structural differences, expressed in the SDT, and the feedbacks rely upon pedagogical differences, expressed in the PDT, but these two taxonomies do not correspond in a straightforward manner. Therefore, we precisely compared them and established a general correspondence table between the two taxonomies, SDT and PDT (Table 1).

Table 1. Correspondence between structural and pedagogical differences.

Structural differences (SDT)	Pedagogical differences (PDT)
Multivalent - Split	Duplication
Multivalent - Merge	Merging
Multivalent - Cluster	*no match*
Univalent - Specific	Misrepresentation, reversion, bad type…
Univalent - Omission	Omission
Univalent - Insertion	Addition
Univalent - Transfer	Transfer
Univalent - Replacement	*no match*
Univalent - Void	*no match*

4.3 Feedbacks Organization and Formulation

The diagnosis method requires the student's diagram to be complete or near-complete. On an incomplete diagram, elements not yet modeled and omitted elements cannot be distinguished. Therefore, the diagnosis step has been integrated after the rereading step. Next, the student turns back to the second step and modifies his/her diagram according to the messages.

The comparison between the learner's diagram and the reference diagram produces a list of differences. These differences do not mean that the learner is wrong, strictly speaking, but they suggest that he could have made a bad choice. Thus, the feedback messages in the Diagram environment are not classical feedback messages: they point specific parts of the diagram out to the learner so as to solicit the metacognitive regulation. As the diagnosis is performed once the modeling task has been completed, we mainly target the *checking* function. After reading the messages and checking the diagram, if the student is still convinced that his/her alternative solution is correct, he/she is free to leave it unchanged. However, some differences may suggest that the student does not master some UML basic concepts: in this case, the feedback relates to the cognitive level and the environment displays a course reminder, for example.

The form of the feedback messages must be adapted according to the degree of certainty about the presence of an error. Starting from the classification proposed by [11], we identified three forms of intervention: notify, question, and propose. "Notify" draws the student's attention to some part of the diagram, or to the way he has modeled a particular concept. "Question" consists in asking the learner about the diagram's properties. The questions are on a binary mode (the answer is "yes" or "no") and encourage the learner to mentally check whether the diagram satisfies a given property. "Propose", the most direct modality, consists in suggesting how to correct the diagram. This kind of help is provided only when the presence of an error is near-certain. During the interaction, feedback messages are graduated from the more general (notify) to the more precise (propose): for each observed difference, the most general message is displayed first, and the learner can bring up a more precise message if he/she wants to.

4.4 Example

We present in this section a complete example to illustrate the production of feedback from the diagram of a learner. Firstly, the diagram designed by the learner and the reference diagram (Fig. 3) are compared by the diagnosis component.

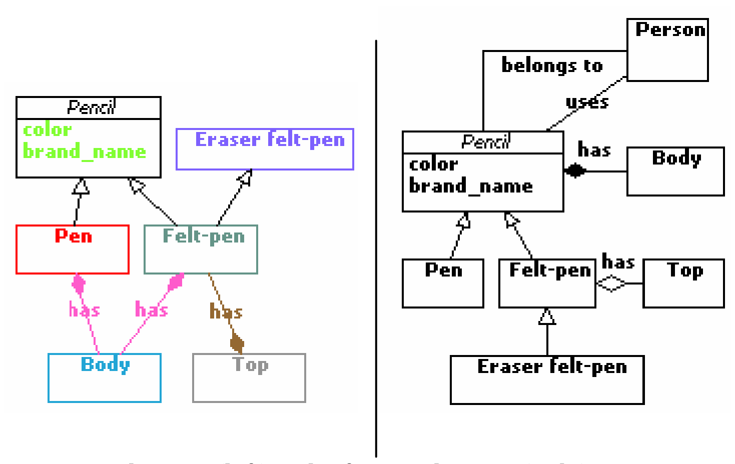

Fig. 3. Learner's diagram (left) and reference diagram (right).

The diagnosis produces thirteen structural differences and thirteen strict matching (*void*). Nine of these thirteen differences match pedagogical differences (Table 2), the others can be ignored because they are factorized by differences at a superior level. Three sets of simple differences correspond to compound differences (A, B and C), the last simple difference is isolated (D).

Table 2. Correspondence between structural and pedagogical differences in the example.

SDT Differences	Simple differences PDT	Compound differences PDT
OMISSION {Person}	Omission of a class	(A) Omission of a class and its linked elements
OMISSION {belongs to (Person --- Pencil)}	Omission of a relationship	
OMISSION {uses (Person --- Pencil)}	Omission of a relationship	
{has (Body---Felt-pen) has (Body --- Pen)} SPLIT { has (Body ---Pencil)}	Duplication of a relationship	(B) Duplication and transfer of a relationship
{has (Body --- Felt-pen)} TRANSFER LOWER {has (Body ---Pencil)}	Transfer of a relationship	
{has (Body ---Pen)} TRANSFER LOWER { has (Body ---Pencil)}	Transfer of a relationship	
{ has (Felt-pen ---Top)} REVERSE { has (Top--- Felt-pen)}	Reversion of a relationship	(C) Misrepresentation of a relationship and reversion of the direction
{ has (Felt-pen --- Top)} COMPOSITION_TO_AGGRE--GATION { has (Top--- Felt-pen)}	Bad kind of relationship	
{Eraser felt-pen---Felt-pen}REVERSE { Felt-pen --- Eraser felt-pen }	(D) Reversion of a relationship	

For each of the four pedagogical differences, a feedback message is produced following the modalities described in the previous paragraph. In the learner's diagram of our example, the "Person" class and the "belongs to" and "uses" relationships between "Pencil" and "Person" are missing. The diagnosis component detects three structural differences which correspond to three simple differences in the PDT. These simple differences constitute a compound pedagogical difference, pointing out the omission of a class and its linked elements (difference A). From this difference, we obtain a pedagogical feedback according to two modalities:

```
Notify: The diagram is incomplete. A main concept has not been represented.
Question: Have you represented the "Person" concept?
```

In the learner's diagram, the "has" composition between the "Pencil" and "Body" classes is duplicated in two compositions and these relationships are transferred to the children classes of "Pencil". These differences result in a compound difference (B), called "duplication and transfer" in the PDT, which is associated to this feedback message:

```
Notify: You say 'Pen has Body' and 'Felt-pen has Body'.
Question: Do the relationships 'Pen-Body' et 'Felt-pen - Body' represent the same
relationship?
Propose: You must merge them into one single relationship, using the "Pencil" and
"Body" classes.
```

The "has" aggregation between "Top" and "Felt-pen" has turned into a composition. At the same time, the direction has been reverted. These two structural differences result in a compound pedagogical difference (C), which contains a bad kind of relationship difference and a reversion difference. The associated message focuses on the reversion of the direction of the relationship.

```
Notify: You say 'Top has Felt-pen'.
Question: Does Top have Felt-pen?
Propose: I would rather say 'Felt-pen has Top'.
```

Finally, the orientation of the generalization relationship between "Eraser Felt-pen" and "Felt-pen" has been reverted. This structural difference corresponds to a simple pedagogical difference (D) and leads to an isolated feedback message according to three modalities:

```
Notify: You say 'Felt-pen is a type of Eraser felt-pen'.
Question: Is Felt-pen a type of Eraser felt-pen?
Propose: I would rather say 'Eraser felt-pen is a type of Felt-pen'.
```

A total of four feedbacks are produced for this diagram, three of them corresponding to compound differences and the last one corresponding to a simple difference. Each feedback come in two or three modalities, as appropriate, and give rise to more or less directive messages. These messages are presented to the learner in a graduated way, at the end of the rereading stage in Diagram.

5 Related work

Different approaches to computer-supported learning environments for object-oriented modeling have been explored, mainly the curriculum-based approach, and the constraint-based approach. Both approaches require an “ideal solution”, given by an expert, which serves as a reference to evaluate the student’s solution.
In the constraint-based approach, the basic principles of the domain are described as constraints, and the student’s answers have to satisfy these constraints. This approach has been used to teach conceptual database design with Kermit [4] or object-oriented design with COLLECT-UML [2]. In both environments, the diagnosis relies on syntactic and semantic constraints that express the possible differences between the student’s answer and the typical correct answer. When a local constraint is violated, the answer is considered to be wrong, and an error message is displayed. This approach is generic and well suited to open environments, but the effectiveness of the diagnosis relies on the number and relevance of the constraints, and in OOM, only a few general semantic constraints exist and can be exploited.
The curriculum-based approach is illustrated by CIMEL-ITS [3], an intelligent tutoring system that provides one-to-one tutoring and helps beginners to learn object-oriented analysis and design during a Computer Science course. It is based on a “design-first” curriculum that introduces object-oriented design using elements of UML before coding. The system compares the student’s solution with its own solution(s) of the current problem and provides real-time guidance. However, this environment works with a limitative list of exercises, and the interaction seems to be very constrained, as opposed to the systems above.

Conclusion

We have described the method applied in Diagram to conceive pedagogical feedbacks. Our approach is based upon the separation between, on the one hand, structural differences that can be identified by comparing the student’s diagram and a reference diagram, and on the other hand, the differences between diagrams considered from a pedagogical point of view, in order to produce feedbacks. We have shown that a correspondence can be established between these two levels, and thus the outputs of a diagram matching algorithm can be exploited to elaborate specific feedback messages that take the student’s production into account. This approach requires more complex tools than the classical approaches (constraint-based or curriculum-based), but the independence of the two levels – structural and pedagogical - make it more generic, because the differences are not automatically considered as errors, they are interpreted from a pedagogical point of view.

The module that produces and displays the feedback messages has been developed and integrated into Diagram. We plan to experiment the whole system with students during fall 2008, and to evaluate the effect of the messages on the learners.

A limitation of the approach is that it relies on a single reference diagram. Nevertheless, the diagnosis algorithm could be extended to handle other knowledge sources, like alternative solutions (or variations in the reference diagram) provided by the teacher.

At present, the feedbacks merely rely on the analysis of the current diagram, and do not take into account the successive versions of the diagram that have been elaborated by the student along the session. The successive diagnosis could be memorized, so as to avoid the repetition of the same messages and to produce more global feedback if the student repeats the same error several times.

References

1. Smyrnaiou, Z.; Weil-Barais, A.: Évaluation Cognitive d'un Logiciel de Modélisation auprès d'Élèves de Collège. Didaskalia 27, 133–149 (2005).
2. Baghaei, N.; Mitrovic, A.: A Constraint-Based Collaborative Environment for Learning UML Class Diagrams. In: Proceedings of the Eighth conference on Intelligent Tutoring Systems (ITS'06), 176–186. Jhongli, Taiwan (2006).
3. Moritz, S.; Wei, F.; Parvez, S.; Blank, G. D.: From Objects-First to Design-First with Multimedia and Intelligent Tutoring. In: Proceedings of the Tenth Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE'05), 99–103. Monte da Caparica, Portugal (2005).
4. Suraweera, P.; Mitrovic, A.: An Intelligent Tutoring System for Entity Relationship Modelling. International Journal of Artificial Intelligence in Education 14 (3–4), 375–417 (2004).
5. Alonso, M.; Py, D.; Lemeunier, T.: A Learning Environment for Object-Oriented Modeling, Supporting Metacognitive Regulations. In: Proceedings of the Eighth IEEE International Conference on Advanced Learning Technologies (ICALT'08). Santander, Spain (2008).
6. Flavell, J. H.: Cognitive Development. Prentice Hall (1977).
7. Brown, A. L.: Metacognition, Executive Control, Self-regulation and Other More Mysterious Mechanisms. In: F.E Weinert and R.H. Kluwe (eds), Metacognition, Motivation and Understanding, chapter 3: 65–116. Lawrence Erlbaum Associates, Hillsdale, New Jersey (1987).
8. Auxepaules, L.; Py, D.; Lemeunier, T.: A Diagnosis Method that Matches Class Diagrams in a Learning Environment for Object-Oriented Modeling. In: Proceedings of the Eighth IEEE International Conference on Advanced Learning Technologies (ICALT'08). Santander, Spain (2008).
9. Sorlin, S.; Solnon, C.; Jolion, J.-M.: A Generic Graph Distance Measure Based on Multivalent Matchings. In: Applied Graph Theory in Computer Vision and Pattern Recognition, 151–182 (2007).
10. Auxepaules, L. : Une Méthode de Diagnostic Basée sur l'Appariement de Modèles dans un EIAH pour la Modélisation Orientée Objet. Actes des Secondes Rencontres Jeunes Chercheurs en EIAH (RJC EIAH'08). Lille, France (2008).
11. Lemeunier, T. : L'Intentionnalité Communicative dans le Dialogue Homme-Machine en Langue Naturelle. Rapport de thèse, Université du Maine (2000).