Assessment of Learner’s Algorithms

Ismail Bouacha[1] and Tahar Bensebaa[2]
[1]*Ecole Préparatoire aux Sciences et Techniques, Annaba, Algérie*
[2]*LRI, Université Badji Mokhtar, Annaba, Algérie*

Keywords: Assessment, Algorithmic, Learner, Program Comprehension.

Abstract: In this paper we propose a method based on program comprehension to assess learners in algorithmic. This method understands automatically the algorithms proposed by the students using comprehension methods from the domain of the software engineering. To assess students propositions, we use prebuilt models of algorithms. These models are documented with information and pedagogical characteristics, and they are organized into tasks and subtasks. We recognize students propositions based on a distance calculus between the model and the proposition. A first experiment and results are presented.

1 INTRODUCTION

This Communication addresses a central theme for teaching: the assessment of learners. The introduction of LMS, MOOC have shown how difficult it is to automate the evaluation of learners and that the various proposals for generic and automatic evaluations (MCQ or peer assessments) fail to target the same objectives and levels of evaluation and diagnosis (Simkin and Kuechler, 2005)(Sitthiworachart and Joy, 2004).

In computer science and more specifically in algorithmic, there is no miracle solution, there is many difficulties: diversity of elements to teach / assess and levels of abstractions concerned, the plurality of expected possible solutions. Methods of evaluations have been taken into account various aspects punctually: consideration of ”writing style” with methods based on static metric (Mengel and Yerramilli, 1999), brutal assessment of semantic proposals based test sets (Chen, 2004), cognitive evaluation mistakes. (Michaelson, 1996)

Our goal materialized by a first implementation and a first experimentation, is to rely on human expertise to gain in generic, richness and completeness of analysis. To achieve this goal, we moved the automatic evaluation to the problem of automatic recognition of patterns of expected copies, these models of copies being provided by an expert or a teacher. By this pattern recognition, we expect to reach a better assessment by taking advantage of what these expected models of copies have been characterized ”by hand” and we believe also release the teacher from mark attribution by building marks from a distance calculation between the proposals to be evaluated and the model that has been previously marked by the expert.

2 PROPOSED ARCHITECTURE AND GENERAL PROCESS

The overall system architecture includes in the center a tool for understanding the proposals (depending on the model proposed, the proposed problem and the students proposition). The proposed algorithms space has a dual function: ”algorithms edition” and ”models modeling.”

The learner accesses that system through the interface of a specific editor AlgoEditor belonging to the space of algorithms expression that structure the learners production allowing him writing only the usual control structures (declaration , assignment, conditional, etc.) according to predefined patterns to complete. The use of this editor relieves learners of some of the most basic editorial efforts. Thus, the learner can focus on higher-level aspects of writing algorithms and the system ensures that proposals are syntactically correct.

This editor is also used by the teacher to model the proposed models. Initially, the teacher can provide some models of a priori propositions. At recognition attempts of the proposal, on failure of recognition, the teacher is called upon to evaluate the pro-

Bouacha, I. and Bensebaa, T.
Assessment of Learner’s Algorithms.
In *Proceedings of the 8th International Conference on Computer Supported Education (CSEDU 2016) - Volume 2*, pages 73-76
ISBN: 978-989-758-179-3

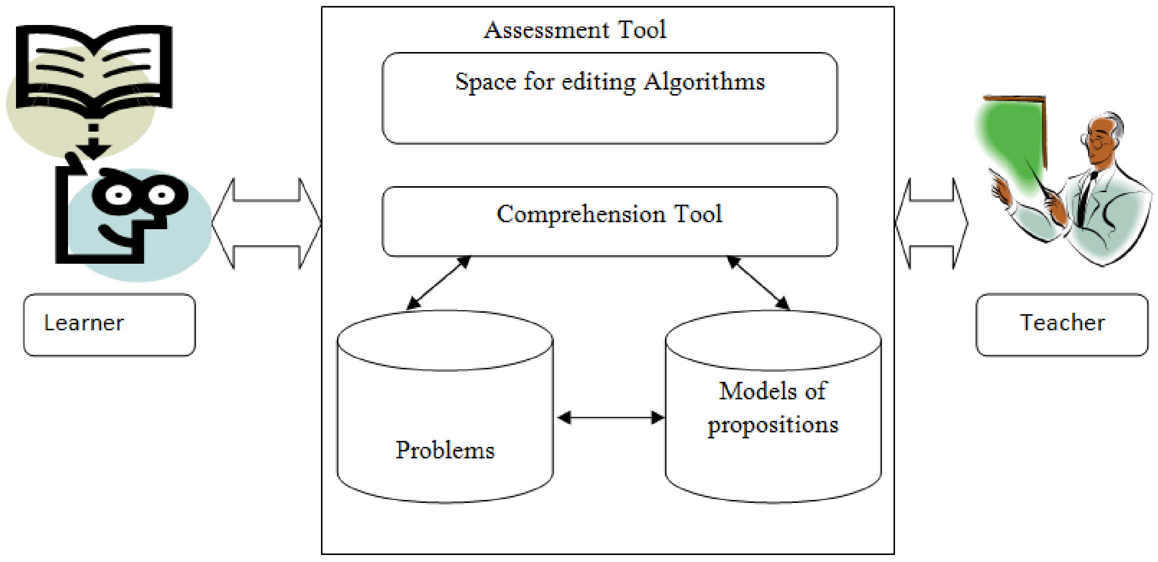


Figure 1: Global architecture of assessment tool.

posal and when the proposal is interesting in its view, the teacher turned it into model to enrich all models of proposition.

3 MODELS OF PROPOSITION AND RECOGNITION ALGORITHMS

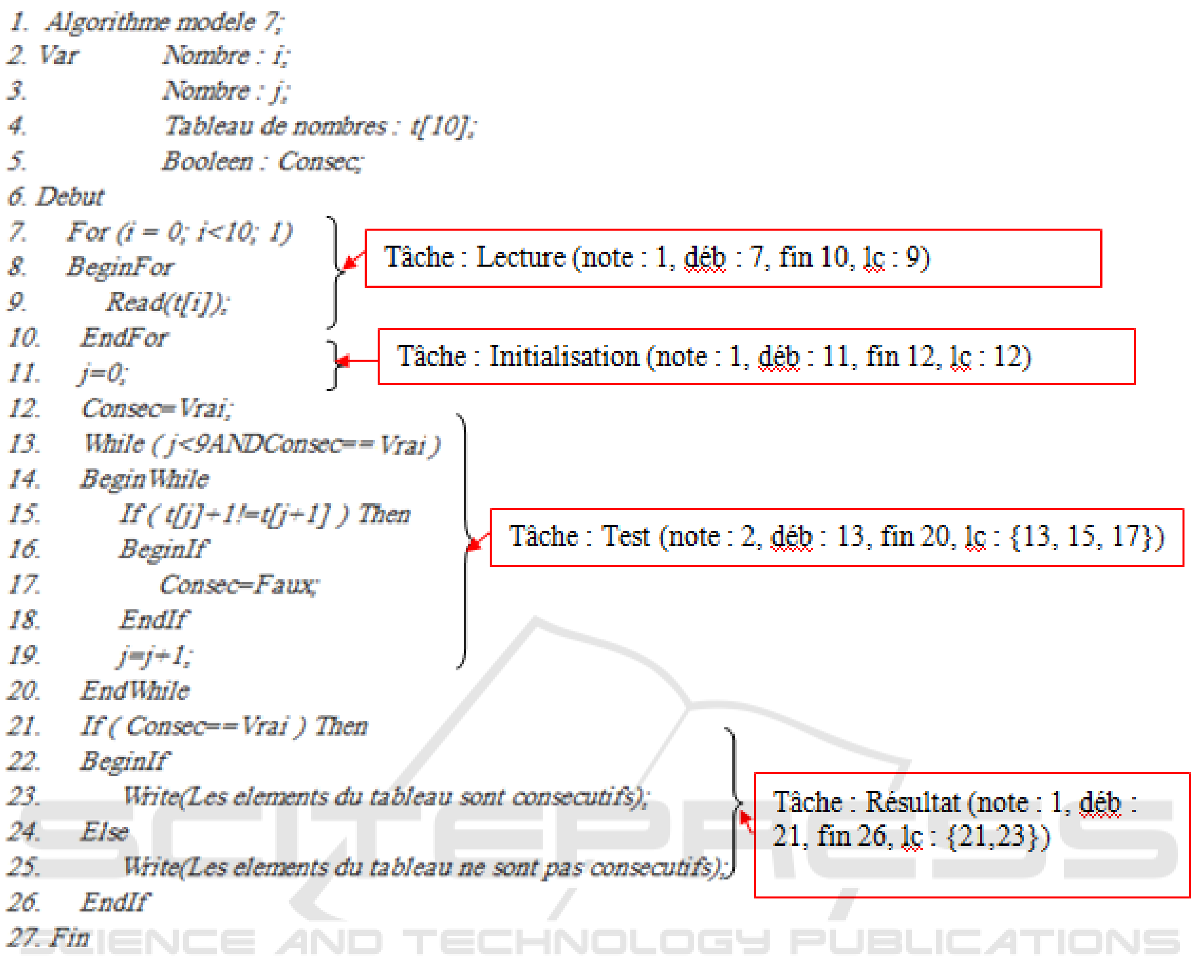
Proposal model is produced from an a priori proposition constructed by the teacher or learners concrete proposal. It contains the code for a specified algorithm and describes what is characteristic in the proposal and the alternatives that this proposal could have. In addition, a model may contain additional information to the highest level (presumed intentions of the programmer, observed errors) and finally an overall score. The model is based on a decomposition of the proposal in terms of tasks and sub-tasks, with associated descriptors. This decomposition and these descriptors are inspired by the Software Engineering program methods of understanding (Corbi, 1989). A model for Proposal is composed of a proposal and the following information:

1. Note: a global mark.
2. Tasks (and subtasks): All tasks and subtasks that make up the proposal. Each task has a name, a local rating, indications of "critical lines" facilitating recognition, those instructions and descriptors (for more details):
   (a) authorize the absence of certain instructions or certain tasks (or penalize),
   (b) authorize any order between some instructions or even between tasks (or penalize disorder),
   (c) alternatives of an expression.
3. free comments

Figure 2 below shows a model for Proposal consists of four tasks: Reading (line 7 to 10), Initialization (line 11-12), Test (line 14-20) and Result (line 21-26, omitted the figure). Assuming that the note of this exercise is on 6 points, we could have the following description for the reading task:

- The reading task is denoted 1 of 6.
- Instructions are between line 7 (starting) and line 10 (end).
- This work contains an indication of critical line (lc) to line 9 (a critical line is an instruction whose appearance in the code proposal indicates the likely presence of a task of a given model.)
- This task is necessary, a penalty of 1 point is awarded absence.
- This task must occur before the tasks and test result, a penalty of 0.5 points is given if the task is after.
- The order between this reading task and the initialization task is indifferent.

The decomposing task and sub-task and adding descriptors can extend the class of proposals accepted for the same model. The objective is thus to achieve greater flexibility for a given model and obtain partial recognition. The models recognition algorithm is inspired by the work in software engineering for program understanding and retro-Eng (Selfridge et al.,



```
1.  Algorithme modele 7;
2.  Var        Nombre : i;
3.             Nombre : j;
4.             Tableau de nombres : t[10];
5.             Booleen : Consec;
6.  Debut
7.     For (i = 0; i<10; 1)
8.     BeginFor
9.        Read(t[i]);
10.    EndFor
11.    j=0;
12.    Consec=Vrai;
13.    While ( j<9ANDConsec==Vrai )
14.    BeginWhile
15.       If ( t[j]+1!=t[j+1] ) Then
16.       BeginIf
17.          Consec=Faux;
18.       EndIf
19.       j=j+1;
20.    EndWhile
21.    If ( Consec==Vrai ) Then
22.    BeginIf
23.       Write(Les elements du tableau sont consecutifs);
24.    Else
25.       Write(Les elements du tableau ne sont pas consecutifs);
26.    EndIf
27. Fin
```

Tâche : Lecture (note : 1, déb : 7, fin 10, lc : 9)

Tâche : Initialisation (note : 1, déb : 11, fin 12, lc : 12)

Tâche : Test (note : 2, déb : 13, fin 20, lc : {13, 15, 17})

Tâche : Résultat (note : 1, déb : 21, fin 26, lc : {21,23})

Figure 2: Proposal Model (decomposition of the task proposal).

1993). It is to abstract the names of variables, find predefined patterns algorithms (task or sub-task, critical lines) and try to assemble these grounds to redial the proposal. To implement the models recognition process, we propose the following algorithm (Simplified):

1. Load the model and the proposal,
2. For each model:
3. Clone the variables of the proposal in the model,
4. Find the critical lines of the model,
5. Repeat:
6. Candidate ← recoverBestCandidate (models)
7. TryModel (candidate)
8. As long as available candidate & unacknowledged proposal.

Upload variable allows a correspondence between each one of the models variables and the equivalent variables in the proposition, from which the clone.

We are looking for in each model critical lines localized in the proposal of the learner, after loading the model variables. This research classifies models with first those most likely to be consistent with the proposal. To choose the best candidate among the proposed models, we estimate the percentage of tasks localized with the percentage of critical lines localized in the proposal of the learner. After getting a candidate model, the step "TryModel (candidate)" is to verify the correspondence between the candidate and the proposal of the learner, by applying the following algorithm (Simplified):

- **For** each model:
- **For** each instruction of the similar proposal at the beginning of the task:
- **For** each instruction as a result of the current task of the model:
- Find a similar statement in the wake of the proposal
- DecideIfProposalIsRecognized

The idea is to try to find the tasks of the model starting with their Beginning. Each possible beginning is identified in the proposal. Next, the rest f the task is sought. In the end, a final procedure determines if the proposal corresponds to the model and what mark (or distance) can be assigned. This procedure operates as a function of matches found and task descriptors. Thus, only tasks with a descriptor in the model that allows them to be absent may be missing. Ditto for the descriptors on the order of tasks. The mark given to the proposal will be maximum if the match is complete and does not require the flexibility allowed by the descriptors.

4 FIRST EXPERIMENTAL RESULTS

To test our approach, we worked with copies of second year computer science exam preparatory school for science and technology Annaba, Algeria, on the following exercise: write a program that checks whether the elements a table (integer) are consecutive or not (for example, the elements 4, 5, 6, 7, 8 are consecutive while the elements 1, 3, 4, 5, 6 are not).

After manual correction of 22 copies, we could extract 7 possible proposal templates (correct and incorrect). With these models, we analyzed 72 new copies automatically (TD from 4 different groups). The average rate of recognition obtained was 56% with a similar recognition rate for the different groups of copies and it was similar for correct copies and incorrect copies.

From models and copies of Groups 1, 2 and 3 we crossed the recognition results with analysis ”by hand” carried out by three teachers. The agreements between the different analyzes were large majority (35 complete agreements, ie 66% of cases where all the judges had the same opinion on the choice of a model or the lack of recognition).

For the 31 copies recognized in groups 1, 2 and 3, an analysis of the assigned rating was performed. Overall, a third of copies (11) received the same mark (mark given on 6); for half the copies (15), the mark received showed a difference of one point with the score on the day of the exam; in 5 cases (16%), the note differed from 2 or 3 points (6 points).

5 CONCLUSION, PERSPECTIVES

We have presented a method for recognition of the learners algorithms. Our method takes advantage of the application context to set up and use a basic algorithm proposals models. The models are enriched with information allowing the use of effective technical of program comprehension used in software engineering, and scoring proposals algorithms from the identified model and the distance between this model and the proposal. An initial experiment from exam papers gave interesting recognition rate (over 50%), similar to the ”hand” recognition rate and marks for the recognized copies closed to manual ratings. To improve the rating, a promising approach would be to combine the recognition algorithm with dynamic algorithm for assessing the proposals based on test cases (Bouhineau, 2013). Beyond the notation, we also believe we can combine the models information on the knowledge and skills relating to each task and sub-tasks and enrich the assessment.

REFERENCES

Bouhineau, D. (2013). Utilisation de traits sémantiques pour une méthodologie de construction d'un système d'aide dans un eiah de l'algorithmique. In *EIAH 2013-6e conférence sur les Environnements Informatiques pour l'Apprentissage Humain*, pages 141–152. IRIT Press 2013.

Chen, P. M. (2004). An automated feedback system for computer organization projects. *Education, IEEE Transactions on*, 47(2):232–240.

Corbi, T. A. (1989). Program understanding: Challenge for the 1990s. *IBM Systems Journal*, 28(2):294–306.

Mengel, S. A. and Yerramilli, V. (1999). A case study of the static analysis of the quality of novice student programs. In *ACM SIGCSE Bulletin*, volume 31, pages 78–82. ACM.

Michaelson, G. (1996). Automatic analysis of functional program style. In *aswec*, page 38. IEEE.

Selfridge, P. G., Waters, R. C., and Chikofsky, E. J. (1993). Challenges to the field of reverse engineering. In *Reverse Engineering, 1993., Proceedings of Working Conference on*, pages 144–150. IEEE.

Simkin, M. G. and Kuechler, W. L. (2005). Multiple-choice tests and student understanding: What is the connection? *Decision Sciences Journal of Innovative Education*, 3(1):73–98.

Sitthiworachart, J. and Joy, M. (2004). Effective peer assessment for learning computer programming. In *ACM SIGCSE Bulletin*, volume 36, pages 122–126. ACM.