Modeling a Tutor for E-Learning Support

Peter Svorada*

Slovak University of Technology in Bratislava
Faculty of Informatics and Information Technologies
Ilkovičova 3, 842 16 Bratislava, Slovakia
psvorada@gmail.com

Peer tutoring is a process where one student supports another in the learning process consisting mostly of diversity of different tasks belonging to the certain domain (mathematics, physics, programming etc.) It has been proven [2] [4], that students involved in such a process (both tutor and tutee) show greater progress compared to process where they study alone.

However peer tutoring provides challenges for the pedagogue to give it a form when this process will not bear too much risk. Students are not educated teachers, sometimes they do not understand the problem very well and can reach dead ends or get repetitively wrong results. This can lead to frustration, demotivation [3].

In our model we try to broaden the existing abilities of PC tutor by adding additional functionality, to see how it will affect the actual learning process. We propose a model of PC tutor capable of telling students that they are working on "a wrong" solution (solution that does not resemble any known correct solution) while they are still working on it. To archive this we created method that can compare two pieces of code from which one is only being written.

There are several different methods used to determine similarity between two pieces of code. While creating our method we used the method of Baxter et al. [1] For comparing abstract syntax trees for its bases. As basis of our method we used the formula used in [1] to calculate similarity between two parts of abstract syntax trees. We have proposed a novel formula and process. The reason for this is mainly based on fact that the original method and our method are both used in different situations. While the original method is used to compare two unknown, complete pieces of code our method is used to compare pieces of code that we suppose should solve the same task and code we compare does not have to be necessarily complete.

The exact process that we use to determine whether the currently written solution is similar to some of the known correct solutions can be divided into several parts:

1. Code pre-processing
2. Creation of abstract syntax tree
3. Determining the similar nodes in abstract syntax trees

* Supervisor: Jozef Tvarožek, Institute of Informatics and Software Engineering

Spring 2012 PeWe Workshop, April 1, 2012, pp.79–80.

4. Calculating of similarity
5. Method application

Code pre-processing must be done in order to prepare code to be parsed into abstract syntax tree. The nodes we create are of different types and we use these types later to compare two nodes and their sub-trees to count similar nodes in each sub-tree. Based on actual number of similar and dissimilar nodes we calculate the actual similarity. For this calculation we used our novel formula.

This method does not tell students that they are working on a wrong solution. It can only determine that the solution they work on is similar to some known correct solution. This, however, can provide increased guidance toward a good/better solution. That is why our tutor does not stop students when the level of similarity is low, it just recommends more deeply correctness of their solution simply because it does not come close to any solution known thus far. Novel solutions that students submit only improve the dataset we can work with.

This is not the only way how a method like this can be used by a PC tutor. The other ability, our tutor has, that derives from the one mentioned, is ability to give students automated advices depending on the current context. Tutor with this functionality can guide students through the tasks using tutorial steps and hints which are provided to the student depending on how he advances in his solution.

Acknowledgement. This work was partially supported by the Scientific Grant Agency of Slovak Republic, grant No. VG1/0971/11.

References

[1] Baxter I., Yahin A., Moura L., Sant Anna M. *Clone Detection Using Abstract Syntax Trees.* In Proceedings of the 14th International Conference on Software Maintenance (ICSM'98), pp. 368-377, Bethesda, Maryland, November 1998.Vardhan, A.: Distributed Garbage Collection: A Transformation and its Applications to Java Programming. Master's thesis, (1998).

[2] Fantuzzo, J. W., Riggio, R. E., Connelly, S., & Dimeff, L. A. (1989). *Effects of reciprocal peer tutoring on academic achievement and psychological adjustment: A component analysis.* Journal of Educational Psychology, 81(2), 173-177.Friedman, A.D., Menon, P.R.: Theory and Design of Switching Circuits. Computer Science Press, Inc., (1975).

[3] Medway, F. & Baron, R. (1977). *Locus of control and tutors' instructional style. Contemporary Educational Psychology*, 2, 298-310.

[4] Roscoe, R. D. & Chi, M. (2007*). Understanding tutor learning: Knowledge-building and knowledge-telling in peer tutors' explanations and questions, Review of Educational Research.* 77(4), 534-574.Henrio, L.: *Asynchronous Object Calculus: Confluence and Determinacy*. PhD. thesis, Universite´ de Nice-Sophia Antipolis, (2003).