Chen, W. et al. (Eds.) (2013). Doctoral Student Consortia Proceedings of the 21st International Conference on Computers in Education. Indonesia: Asia-Pacific Society for Computers in Education

Incorporating Anchored Learning in a C# Intelligent Tutoring System

Budi HARTANTO[a*], **Jim REYE**[b*]
[a, b] *Science and Engineering Faculty, Queensland University of Technology, Australia*
*b1.hartanto@qut.edu.au, j.reye@qut.edu.au

Abstract: Learning programming is known to be difficult. One possible reason why students fail programming is related to the fact that traditional learning in the classroom places more emphasis on lecturing the material instead of applying the material to a real application. For some students, this teaching model may not catch their interest. As a result they may not give their best effort to understand the material given. Seeing how the knowledge can be applied to real life problems can increase student interest in learning. As a consequence, this will increase their effort to learn. Anchored learning that applies knowledge to solve real life problems may be the key to improving student performance. In anchored learning, it is necessary to provide resources that can be accessed by the student as they learn. These resources can be provided by creating an Intelligent Tutoring System (ITS) that can support the student when they need help or experience a problem. Unfortunately, there is no ITS developed for the programming domain that has incorporated anchored learning in its teaching system. Having an ITS that supports anchored learning will not only be able to help the student learn programming effectively but will also make the learning process more enjoyable. This research tries to help students learn C# programming using an anchored learning ITS named CSTutor. Role playing is used in CSTutor to present a real world situation where they develop their skills. A knowledge base using First Order Logic is used to represent the student's code and to give feedback and assistance accordingly.

Keywords: Anchored learning, C# programming, Intelligent Tutoring System

1. Introduction

Learning programming is known to be difficult. This situation is shown by the significant number of students in computer science departments who fail programming (Robins, Rountree, & Rountree, 2003; Teague & Roe, 2008). One possible reason why students fail programming is related to the fact that traditional learning in the classroom places more emphasis on lecturing the material, instead of applying the material to a real application. Because of this, some students may lose their interest in learning and not give their best effort to understand the material.

Anchored learning on the other hand, puts the emphasis of learning on solving an interesting real life problem. This learning can increase student interest in learning because they can see how to apply facts and skills to solve the real life problem, instead of just learn the facts and skills themselves. In addition, because the students are presented with a realistic environment, it is likely that they will be able to recall the information more easily when they face another similar problem in the future.

There are some applications that have been built with an anchored learning in their design (Eliot & Woolf, 1996; Schank & Cleary, 1995). However, none of them have been developed to support learning in the programming domain. On the other hand, a number of Intelligent Tutoring Systems (ITSs) have been developed in the programming domain (Anderson, Corbett, Koedinger, & Pelletier, 1995; Johnson & Soloway, 1985; Mitrovic, 2003; Sykes, 2007; Weragama & Reye, 2013). However, none of these ITSs have been designed to support anchored learning. Therefore the potency of anchored learning in the programming domain has not been fully explored.

This research tries to help students learn programming by incorporating anchored learning into a programming domain ITS. This ITS, named CSTutor, will be used by students in a programming course in C# at the Queensland University of Technology (QUT) as a supplement to the classroom instruction and is not meant to entirely replace the role of the teacher. CSTutor is being developed as an integrated tool that runs inside Visual Studio. Through this approach, the students can write their programs in Visual Studio directly and CSTutor will provide assistance and feedback when the students request it.

2. Anchored Learning in CSTutor

Anchored learning is known to motivate students to learn and to enhance their problem solving skills (Shyu, 2000). One reason behind this finding is that the learning process is placed in an authentic context. This context makes the learning become interesting because it reflects the true nature of problems in the real world (Grabinger & Dunlap, 1995).

In anchored learning, the learning or teaching activities should be designed around a story or situation that includes a problem or issue. In this research, role playing is used to let the student "play" as they learn programming. In this role playing, a simulated Head of Development officer appears in the application and offers a programming job to the student if he/she can show his/her ability at writing programs, for some given problems.

The Head of Development also tells student that he/she can use CSTutor – an intelligent tutor that can help the student in writing the program by providing help, and feedback. CSTutor will also be used by the Head of Development to give the problems to be solved by the student.

In the role play, the first goal for the student is to get accepted into the company. This can be achieved by writing several correct programs for the given problems. After the student gets accepted, the next goal that must be achieved by the student is to get promoted to Intermediate Programmer, and later on, to Senior Programmer. The role playing finishes when the student achieves the position of a Senior Programmer. Achieving a certain position in the company is expected to become the student's higher-level goal when they write their programs. By having a higher-level goal, the work that need to be done becomes more interesting and less burdensome (Schank & Cleary, 1995).

The salary that would be received by the students when they get accepted into a company, and when they get promoted to a higher position, is shown to the students. This is expected to give more encouragement to the students, knowing that they can earn that much money if they really get that kind of job in a company. The salary amount is based on similar real-world programming positions in the year 2012.

In addition, to make the role playing become more realistic, all of the problems given to the students are designed within a particular context. Therefore, the student will never be given a problem with a simple instruction like: "swop the numbers that exist in two variables", etc. There is always a real life problem behind the instruction.

3. Providing Assistance to the Student as they Learn Programming

Another principle of anchored learning is the existence of resources that can be accessed by the student as they solve each problem. CSTutor provides resources for the students when they have problems while writing their programs. CSTutor gives help and feedback to the students, based on the current state of their program.

In anchored learning, the students should be challenged to think about and work on each problem. Based on this principle, the help and feedback that is given by CSTutor are presented as hints or clues, instead of a direct solution. These hints or clues are presented gradually from more general to more detailed, when the student keep requesting help for the same problem. This encourages the students to think and solve the problem by themselves. Only when the student is completely stuck, will the final answer eventually be given.

The process of generating the help and feedback is started by parsing the student's program and converting it into facts or actions in a predicate logic form. The actions are used to change some existing facts into another facts. All of these facts are then checked using rules in the knowledge base to see if they can satisfy the goal of the given problem or not. In this approach, several important steps are traced to see if the student program can solve the given problem or not.

For example, let's say that the student is given a task to find the number of real roots that exist for a quadratic equation $ax^2+bx+c = 0$. The number of roots – for example, stored in a variable named *root* – is based on the value of *d* which is defined as: $d = b^2 – 4ac$. In this kind of problem the steps that are traced in CSTutor are:

- The existence of the required variables (*a*, *b*, *c*, *d*, *root*)
- The input process to variable *a*, *b*, *c*
- The calculation to get the value in *d*
- The correctness of the conditional statement to assign the value in *root*
- The display of the variable *root*

CSTutor uses rules to check the correctness of the student code in every step. Using this approach, the students have some flexibility in writing their program. Due to limitations on the length of this paper, it is not possible to explain all of the processes used in CSTutor to analyse the student code. One process that will be explained in here is how CSTutor analyses student code that include conditions (the fourth step of the required steps above)

Figure 1 shows partial code of some possible solutions in the step of assigning the value to variable *root*. The number of roots of a quadratic expression can be determined based on the value of *d*. If *d*> 0 then the quadratic expression will have 2 roots, if *d* = 0 the quadratic expression will have 1 root, and if *d*< 0, the quadratic expression will have no roots.

Version 1.
```
    if (d < 0)
      root = 0;
    else if (d == 0)
      root = 1;
    else
      root = 2;
```

Version 2.
```
    if (d > 0)
      root = 2;
    else if (d == 0)
      root = 1;
    else
      root = 0;
```

Version 3.
```
    root = 0;
    if (d >= 0){
      if (d > 0)
        root = 2;
      else
        root = 1
    }
```

Figure 1. Several possible solutions to assign value to variable *root*

For this, the goal that is specified in CSTutor is shown in **Figure 2**. In this case the value in variable *root* should be the same as:

- 2 if the value in variable *d* is greater than (GT) 0,
- 1 if the value in variable *d* is equal (EQ) 0
- 0 if the value in variable *d* is less than (LT) 0.

```
Goal:
    HasVarValue(varID_d, val_d)
    HasVarValue(varID_root, 2) ← GT(val_d, 0)
    HasVarValue(varID_root, 1) ← EQ(val_d, 0)
    HasVarValue(varID_root, 0) ← LT(val_d, 0)
```

Figure 2. The goal specification to find the roots of a quadratic expression

The first process performed in CSTutor is to parse the student's code and store it as facts or actions. For code that contains *if-else* statement, all the implicit conditions will be made explicit. This implicit condition can occur because of an *else* part, or because of a nested *if*. For example, the code in version 3 of **Figure 1** will be converted into the following predicates. The conditions shown in bold are the explicit conditions that occur from the process.

```
HasVarValue(varID_root, 2) ←GE(val_d, 0)∧ GT(val_d, 0)
HasVarValue(varID_root, 1) ←GE(val_d, 0)∧NGT(val_d, 0)
HasVarValue(varID_root, 0) ←NGE(val_d, 0)
```

These conditions are then simplified to become:

HasVarValue(varID_root, 2) ← GT(val_d, 0)

HasVarValue(varID_root, 1) ← EQ(val_d, 0)
HasVarValue(varID_root, 0) ← LT(val_d, 0)

The results from this parsing process are then checked using rules in knowledge base to see if the predicates from the student code can satisfy the problem's goal or not. Using these rules, CSTutor can recognize if two conditions are logically the same even though they were written differently. For example the condition LE(1, val_d) can be recognized to be logically the same as GT(val_d, 0) for an integer number. By using these kinds of rules, the student does not have to write code in exactly the format as the conditions specified in the goal.

4. Expected Contributions of this Research

This research aims to help students learn C# programming by incorporating the concept of anchored learning in an Intelligent Tutoring System. By having an ITS as a private tutor, the students can have as much assistance as they need. In addition, the use of anchored learning in CSTutor is expected to make the learning process more enjoyable and less burdensome for the students.

A knowledge base is used in CSTutor to check whether the student's program code is correct. And if incorrect, then where it is incorrect. This knowledge base makes it possible to check a variety of student code. The knowledge base is used to give feedback and assistance to student.

This research is expected to give contribution in revealing the effectiveness of an ITS that incorporates anchored learning to help students learn programming. An evaluation will be performed to measure this effectiveness. Another evaluation will be performed to see whether the students regard the anchored learning approach as making the learning more interesting, or not. If the results are positive, these findings will enable educators to consider using this approach when teaching programming.

References

Anderson, J. R., Corbett, A. T., Koedinger, K. R., & Pelletier, R. (1995). Cognitive tutors: Lessons learned. In *Journal of the Learning Sciences* (pp. 167-207): JSTOR.

Eliot, C., & Woolf, B. (1996). Iterative development and validation of a simulation-based medical tutor. In *Third International Conference on Intelligent Tutoring Systems* (pp. 540-549): Springer.

Grabinger, R. S., & Dunlap, J. C. (1995). Rich environments for active learning: A definition. *Research in Learning Technology, 3*(2).

Johnson, W. L., & Soloway, E. (1985). PROUST: Knowledge-based program understanding. *Software Engineering, IEEE Transactions on*(3), 267-275.

Mitrovic, A. (2003). An intelligent SQL tutor on the web. *International Journal of Artificial Intelligence in Education, 13*(2), 173-197.

Robins, A., Rountree, J., & Rountree, N. (2003). Learning and teaching programming: A review and discussion. *Computer Science Education, 13*(2), 137-172.

Schank, R. C., & Cleary, C. (1995). *Engines for education*. New Jersey: Lawrence Erlbaum.

Shyu, H. Y. C. (2000). Using video-based anchored instruction to enhance learning: Taiwan's experience. *British Journal of Educational Technology, 31*(1), 57-69.

Sykes, E. (2007). Developmental process model for the Java intelligent tutoring system. *Journal of Interactive Learning Research, 18*(3), 399-410.

Teague, D., & Roe, P. (2008). Collaborative learning: Towards a solution for novice programmers. In *Proceedings of the Tenth Conference on Australasian Computing Education* (Vol. Volume 78, pp. 147-153). Wollongong, NSW, Australia: Australian Computer Society, Inc.

Weragama, D., & Reye, J. (2013). The PHP Intelligent Tutoring System. In *Artificial Intelligence in Education* (pp. 583-586): Springer.