Diagnostic Web-based Monitoring in CS1

Olle Bälter
KTH CSC
100 44 STOCKHOLM
SWEDEN
+46 8 790 6341
balter@kth.se

ABSTRACT
Students that fall behind during a course are a concern in any teaching situation. Falling behind has negative effects both for students, teachers and the university. Close monitoring of the learning and development can be effective, but is in general time-consuming and expensive. The use of a web-based diagnostic system that can generate a large (infinite) number of questions could make monitoring both time and cost effective.

Categories and Subject Descriptors
K.3.2 [**Computers and Education**]: Computer and Information Science Education – *computer science education, learning.*

General Terms
Experimentation.

Keywords
Computer Science Education, Pedagogy, Generic questions.

1. INTRODUCTION
Science teachers too often experience how a student approach them towards the end of a course and reveal that they did not understand the topic of the 2nd week of the course and therefore have been unable to understand the rest. Teachers are of course aware of this problem and therefore introduce various minor tests and/or lab assignments before the final to promote continuous learning. However, as s a natural part of laboratory assignments there are also lot of support available from teachers and assistants. While this support is essential to help some students forward it can also be unintentionally misleading for some that can produce lab results (reports or in computer science: source code), but without understanding exactly why.

In some cases teachers blame the students that do not study or seek help early enough, but after spending a semester at an American top college with excellent students and still observing the same phenomena, it is clear that this happens even among very talented students. In general, an experienced teacher get a sense rather quickly which students are in danger of failing, but without hard evidence of the case it is difficult to initiate a discussion with the student. The teacher may be wrong, and the student may be in denial.

If we take the idea of assessment during the course to an extreme, we would constantly be assessing the students. This might have benefits, but takes time from teaching and interaction with the students and also feels a lot like baby-sitting.

One alternative that adds only a little workload to the teacher is to ask the students to hand in reflections over their learning. However, although beneficial in many ways, it adds to the workload for the students, and students with authoring skills may hand in reflections that seems right, but still has misunderstood some concepts, in similar ways that a verbally skilled student may slip through an oral examination of a lab assignment.

The solution should therefore minimize the time spent both for the teacher and the students and contain precise questions that can be assessed automatically. This way, the teacher only have to read a summary of the results and does not have to spend any time reading answers that are correct, which normally should be the vast majority.

2. RELATED RESEARCH
Introductory courses in computer science is a constant topic of discussion among academics and the hurdles to learn programming have been lowered by various tools, such as narratives, visual programming, robots, Lego [15] and visualizations of programs [14, 16]. One of the criticisms is that many students do not know how to program after an introductory course [13] and that programming assignments are subject to plagiarism [5]. Students report that it is acceptable to copy the majority of an assignment from a friend [19] and in one study, 40% of students plagiarized at least one assignment [3]. There are reports of 20% of the students failing the course [12].

Computer Assisted Assessment (CAA) is often presented as THE solution for education of the future. The opinions on Computer Aided Assessment (CAA) is split among academics, but there are claims that this is mostly due to experience with CAA[3]. There are several systems for CAA [11, 12] and there are studies that report no significant difference in examination between online exercising and classroom exercising [8]. Among the advantages with CAA is the possibility to personalize assignments and to resubmit answers (which is important from a constructivist perspective), which improved grades greatly, but the share of failing students remain approximately the same (slightly under 20% in [12]).

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
Koli Calling'09, October 29–November 1, 2009, Koli, Finland.
Copyright 2009 ACM 978-1-60558-952-7/09…$5.00.

However, there are also disadvantages with CAA [24]. If CAA is used on the web, the problem of knowing who is answering the questions and also whether this person is receiving help or not [3] becomes difficult. One negative aspects of CAA not mentioned in the literature (maybe because it is too obvious) is that constructing problems and evaluations that are correct becomes even more essential, as slips, mistakes and errors cannot be handled as smoothly as on a written exam or a lab assignment where that teacher simply can admit the mistake and correct it immediately.

In order to improve learning, counter plagiarism and reduce the number of failing students (regardless whether they fail the course or not), we also need to improve assessment.

Lecturers often do not know how well students are doing until after the first assessment. At this point, it may be too late to prevent struggling students from falling [1]. We therefore need to assess students early and with problems suitable for their learning. We know that deep approach to learning not surprisingly leads to higher grades [20], but also that students' expectation of their own grade on the introductory course is the most important indicator of performance [18] and the students' comfort level is the best predictor of success [21], and the strongest relationship between fifteen factors and performance on a programming module was a student's perception of their understanding of the module [1]. One study suggest that weaker students should only be required to gain the ability to read and understand programs, and thereby demonstrating knowledge and comprehension (using Bloom's taxonomy) [9]. The initial assessment on these levels should be a part of the early identification of struggling students. Passing these simpler problems could strengthen their self-confidence and perception of the subject and thereby improve learning in the entire course.

There is at least one report of weekly tests [23], but this was made in labs, which of course reduce time for interaction. However, these weekly quizzes dramatically reduced failure rates [23] and lab exams are better assessors of programming ability than traditional methods such as written exams and programming assignments [5] but an examination of novice programmers and the SOLO (Structure of the Observed Learning Outcome) taxonomy ends with a recommendation to mix training and assessment of reading and writing tasks [10].

There are already online programming assessment tools [11, 17], but unlike the proposal in [17], we are only suggesting a pedagogical methodology, not a technical system. There is also an argument against combined development and assessment systems: the students are not learning to use the tools used in "real" development. Self-assessment has also been used successfully for terminology quizzes as a way to encourage reading lecture material before class [22], our proposal goes a little further.

3. PROPOSAL

A web-based system for small diagnostic tests would liberate students from coordinating assessments in time and place and the teacher could automatically be sent a summary by email. However, creating sufficient number of questions in such a system would be a very time-consuming endeavor, and if the number of questions is too few, there is always a risk that some students will copy answers from others.

A remedy to this problem is to use *generic* questions. A generic question is a question formulated in a way that makes it possible to construct a large (even infinite) number of questions from it. For example, as a first problem in CS1, the following code is provided:

```
a = 17

b = a

a = 42
```

and the question follows: what is the value of b? Depending on whether a and b are primitive or reference variables, the answer will be 17 or 42, respectively. Examining the question we can realize that a can be replaced with any valid variable name, as can b; and 17 and 42 can be replaced with any variable value.

Similar constructions can easily be transferred to mathematics (and are in use in web courses at our university) and possible to other science subjects as well. The foundation is that the question is *determinable* (that is, all answers can be classified as 100% right or 100% wrong) and *input dependent* (that is, there is input to the question and this input can be varied and effect output).

The web technology makes it possible to give students a small test every day (or before or after a lecture, a lab etc.). A student that fail the test may, thanks to the generic formula, be given a new test immediately in the spirit of constructivism (this idea was proposed by Keller in 1968 [6, 7]). This monitoring could improve the situation for students, teachers and the university.

From a student perspective this system could improve

- learning, as the tests will inspire some students to study first,
- clarifying whether the student has understood or not (it is easy to think you can because everything seems so simple when the teacher explains),
- teacher support, as failing the test repeatedly will give a clear signal that the student needs assistance, both to the student and the teacher.

From a teacher perspective the system can give information in several ways:

- Individual level: which students failed (more than once on a question) this can be used to approach these students to give them support
- Group level: Reports on how many (percentage) have failed (the first time) on each question and use that to repeat instructions during the course and improve the explanation to the next course
- With test results stored in a database it could also be used to detect negative trends (students that never use to fail suddenly fails)

From a university perspective the system could improve:

- Throughput of students as failures can be detected and corrected much earlier.
- Results in general as study habits improve.
- If the system is used in several courses, it could also be used to identify students that struggle in several subjects (many failures in several courses).

We have experience of similar attempts from the test system in a previous project [2]. A minor part of that project is still in use in

```
Introduction to Computer Science
Summary of test 1 September 9 2009

General
85% passed on their first attempt
10% passed on their second attempt
5% needed more than two attempts

Students with none or more than two attempts
Adam Sandler: 7 Failed
Britney Spears: 3 Passed

Questions
Attempts: 1  2  3  3+
Q1        45  3  2
Q2        48  2
Q3        45  3  1  1
```

Figure 1. Example of output from the system to the

on-campus courses in programming for the mid-term. The main difference between this project and previous is

- We now have an infrastructure for development and maintenance of the system.
- The focus on generic questions that has undisputedly right or wrong answers.
- The technical solution is far more evolved with a complete database, logging of web activities, etc.

Initially we will start with courses in computer science where we have most knowledge and experience, but we clearly see how these ideas can be extended into other sciences and for parts of social science and humanities.

4. STUDY DESIGN

There is a web-based system in use at our university, but today it is only used for distance education. This system will be a stable foundation for the experiments we intend to perform. There are generic questions on math and programming in the system, but these are ad hoc and there is no general way to introduce new questions, and no teacher interface for this.

In order to add functionality for generic questions and student monitoring we intend to:

- Observe how the ad-hoc solution to generic exam questions that are in use today can be used for diagnostic purposes, as proposed.
- Based on these observations, propose a design for the system so that any teacher can add new generic questions and use the system.
- Develop a model for organization and continuous improvement of a database with generic questions that all teachers have access to.

We will do this in two simultaneous pilot studies, one at an American college, and one at a Swedish university. At both sites the pilot will be run in an introductory course in computer science using Java or Python.

An example of output from the system to the teacher can be found in Figure 1. This could be sent via email to the teacher or be shown on a secure web page. The amount of information presented should be limited and the thresholds should be possible to configure. The information is divided in three sections. The general section is to get a sense for how the entire class has done on the test. The student section lists names of students with more than two attempts. The purpose is that the teacher should get information on which students that may need extra support. In this mock example, Britney Spears does not seem to need any assistance but Adam Sandler definitely does. The section with questions is not interesting at all in this example, but in case there is something wrong with one of the questions it will be clear which from this information. This may be attributed to a failure in formulating the question or if the question is correct, there might be something overseen in the teaching and course material.

5. DISCUSSION QUESTIONS

- Is this a good idea from a teacher's perspective?
- Is this a good idea from a student's perspective?
- How should the pilots be evaluated?
- Are there more efficient ways to achieve the same goals?

6. ACKNOWLEDGMENTS

Thanks to the project Virtual Campus at Resource Centre for Net-based Education, KTH Royal Institute of Technology for financing and STINT, Swedish Foundation for International Cooperation in Research and Higher Education for additional travel grants.

7. REFERENCES

[1] Bergin, S. and Reilly, R. 2005. Programming: factors that influence success. In *Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education* (St. Louis, Missouri, USA, February 23 - 27, 2005). SIGCSE '05. ACM, New York, NY, 411-415.

[2] Bälter, O. 2004. WIKKED (in Swedish) URL: www.nada.kth.se/utbildning/projekt/wikked Last Visited June 8, 2009.

[3] Carter, J., Ala-Mutka, K., Fuller, U., Dick, M., English, J., Fone, W., and Sheard, J. 2003. How shall we assess this?. In *Working Group Reports From ITiCSE on innovation and Technology in Computer Science Education* (Thessaloniki, Greece, June 30 - July 02, 2003). D. Finkel, Ed. ITiCSE-WGR '03. ACM, New York, NY, 107-123.

[4] Daly, C. and Horgan, J.M., (2001), Automatic Plagiarism Detection, Proceedings of the IASTED International

Conference Applied Informatics, pp.255-259. Innsbruck, Austria, Feb 2001.

[5] Daly, C. and Waldron, J. 2004. Assessing the assessment of programming ability. In *Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education* (Norfolk, Virginia, USA, March 03 - 07, 2004). SIGCSE '04. ACM, New York, NY, 210-213.

[6] Herzberg P. 2001. The Keller Plan: 25 Years of Personal Experience. In Positive Pedagogy – Successful and Innivative Strategies in Higher Education, vol. 1, #1. ISSN: 1496-8126.

[7] Keller F S. 1968. "Goodbye, teacher..." J. Appl. Behavioral Analysis. Vol 1(1), pp 79-89.

[8] Korhonen, A., Malmi, L., Myllyselkä, P., and Scheinin, P. 2002. Does it make a difference if students exercise on the web or in the classroom?. In *Proceedings of the 7th Annual Conference on innovation and Technology in Computer Science Education* (Aarhus, Denmark, June 24 - 28, 2002). ITiCSE '02. ACM, New York, NY, 121-124.

[9] Lister, R. and Leaney, J. 2003. Introductory programming, criterion-referencing, and bloom. In *Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education* (Reno, Navada, USA, February 19 - 23, 2003). SIGCSE '03. ACM, New York, NY, 143-147.

[10] Lister, R., Simon, Thompson, E., Whalley, J. L., and Prasad, C. 2006. Not seeing the forest for the trees: novice programmers and the SOLO taxonomy. In *Proceedings of the 11th Annual SIGCSE Conference on innovation and Technology in Computer Science Education* (Bologna, Italy, June 26 - 28, 2006). ITICSE '06. ACM, New York, NY, 118-122.

[11] Malmi, L., Karavirta, V., Korhonen, A. and Nikander, J. (2005): Experiences on automatically assessed algorithm simulation exercises with different resubmission policies. In ACM Journal of Educational Resources in Computing, 5 (3)

[12] Malmi, L., Korhonen, A., and Saikkonen, R. 2002. Experiences in automatic assessment on mass courses and issues for designing virtual courses. In *Proceedings of the 7th Annual Conference on innovation and Technology in Computer Science Education* (Aarhus, Denmark, June 24 - 28, 2002). ITiCSE '02. ACM, New York, NY, 55-59.

[13] McCracken, M., Almstrum, V., Diaz, D., Guzdial, M., Hagan, D., Kolikant, Y. B., Laxer, C., Thomas, L., Utting, I., and Wilusz, T. 2001. A multi-national, multi-institutional study of assessment of programming skills of first-year CS students. In *Working Group Reports From ITiCSE on innovation and Technology in Computer Science Education* (Canterbury, UK). ITiCSE-WGR '01. ACM, New York, NY, 125-180.

[14] Naps, T. L., Eagan, J. R., and Norton, L. L. 2000. JHAVÉ—an environment to actively engage students in Web-based algorithm visualizations. In Proceedings of the Thirty-First SIGCSE Technical Symposium on Computer Science Education (Austin, Texas, United States, March 07 - 12, 2000). S. Haller, Ed. SIGCSE '00. ACM, New York, NY, 109-113. DOI= http://doi.acm.org/10.1145/330908.331829

[15] Powers, K., Gross, P., Cooper, S., McNally, M., Goldman, K. J., Proulx, V., and Carlisle, M. 2006. Tools for teaching introductory programming: what works? In *Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education* (Houston, Texas, USA, March 03 - 05, 2006). SIGCSE '06. ACM, New York, NY, 560-561.

[16] Rajala, T., Laakso, M.-J., Kaila, E. and Salakoski, T. (2007). VILLE - a language-independent program visualization tool. In Proc. Seventh Baltic Sea Conference on Computing Education Research (Koli Calling 2007), Koli National Park, Finland. CRPIT, 88. Lister, R. and Simon, Eds. ACS. 151-159.

[17] Roberts, G. H. and Verbyla, J. L. 2003. An online programming assessment tool. In *Proceedings of the Fifth Australasian Conference on Computing Education - Volume 20* (Adelaide, Australia). T. Greening and R. Lister, Eds. Conferences in Research and Practice in Information Technology Series, vol. 140. Australian Computer Society, Darlinghurst, Australia, 69-75.

[18] Rountree, N., Rountree, J., and Robins, A. 2002. Predictors of success and failure in a CS1 course. *SIGCSE Bull.* 34, 4 (Dec. 2002), 121-124.

[19] Sheard, J., Dick, M., Markham, S., Macdonald, I., and Walsh, M. 2002. Cheating and plagiarism: perceptions and practices of first year IT students. In *Proceedings of the 7th Annual Conference on innovation and Technology in Computer Science Education* (Aarhus, Denmark, June 24 - 28, 2002). ITiCSE '02. ACM, New York, NY, 183-187.

[20] Simon, Fincher, S., Robins, A., Baker, B., Box, I., Cutts, Q., de Raadt, M., Haden, P., Hamer, J., Hamilton, M., Lister, R., Petre, M., Sutton, K., Tolhurst, D., and Tutty, J. 2006. Predictors of success in a first programming course. In *Proceedings of the 8th Austalian Conference on Computing Education - Volume 52* (Hobart, Australia, January 16 - 19, 2006). D. Tolhurst and S. Mann, Eds. ACM International Conference Proceeding Series, vol. 165. Australian Computer Society, Darlinghurst, Australia, 189-196.

[21] Wilson, B. C. and Shrock, S. 2001. Contributing to success in an introductory computer science course: a study of twelve factors. In *Proceedings of the Thirty-Second SIGCSE Technical Symposium on Computer Science Education* (Charlotte, North Carolina, United States). SIGCSE '01. ACM, New York, NY, 184-188.

[22] Williams, G. C., Bialac, R., and Liu, Y. 2006. Using online self-assessment in introductory programming classes. *J. Comput. Small Coll.* 22, 2 (Dec. 2006), 115-122.

[23] Woit, D. and Mason, D. 2003. Effectiveness of online assessment. In *Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education* (Reno, Navada, USA, February 19 - 23, 2003). SIGCSE '03. ACM, New York, NY, 137-141.

[24] Zhang, D., Zhao, J. L., Zhou, L., and Nunamaker, J. F. 2004. Can e-learning replace classroom learning?. *Commun. ACM* 47, 5 (May. 2004), 75-79.