Pedagogical Transformations in the UBC CS Science Education Initiative

Donald Acton, Kimberly Voll, Steven Wolfman, Benjamin Yu
University of British Columbia
Department of Computer Science
2366 Main Mall
Vancouver BC V6T 1Z4
{acton, kvoll, wolf, benyu}@cs.ubc.ca

ABSTRACT
The UBC CS Science Education Initiative (CSSEI) has resulted in a number of research projects. New teaching methods and student assessment instruments are introduced to engage student learning and evaluations of their understanding. In this paper, we report four of these recent initiatives and their initial findings.

Categories and Subject Descriptors
K.3.2 [**Computers and Information Science Education**]: Pedagogy, education research – *just-in-time, BRACElet, in-class activities, attitudinal survey, problem-based learning, analysis.*

General Terms
Experimentation, Measurement.

Keywords
Computer education, transformation, code understanding, survey analysis, grade analysis.

1. Introduction
The UBC CS Science Education Initiative (CSSEI) is part of the five-year, $12 million project at the University of British Columbia under the Carl Wieman Science Education Initiative (CWSEI), which began in 2007. The project aims to dramatically improve undergraduate science education by taking a four-step, scientific approach to teaching: 1) establishing what students should learn through the identification of learning goals, 2) scientifically measuring what students are actually learning, 3) adapting instructional methods and curriculum, and incorporating effective use of technology and pedagogical research to achieve desired learning goals, and 4) disseminating and adopting what works for student learning. In the UBC CS department, we have developed learning goals for five of our core courses [3]. We have also identified some of the difficulties students face in achieving these learning goals. This paper describes some of our ongoing research projects to help students better achieve these learning goals, and our initial results.

© ACM, 2009. This is the author's version of the work. It is posted here by permission of ACM for your personal use. Not for redistribution. The definitive version was published in:
Proceedings of the 14th Western Canadian Conference on Computing Education, DOI:
http://doi.acm.org/10.1145/1536274.1536280

Just-in-Time Teaching
Previous studies of our CPSC 121: Models of Computation course have shown that students have trouble connecting topics and retaining core ideas from term to term, and from lectures to labs. Many students found the hands-on labs (often circuit-based) disconnected from the more theoretical material discussed in class. Downstream courses also tend to extensively review (re-teach) material that students learn in the course. These problems remain despite significant development efforts on the course by many faculty over several years.

In two sections of this course offered in the January to April 2009 term, we have shifted towards a heavily problem-based lecture style enabled by Just-in-Time-Teaching (JITT) [5].

The core tenets of JITT are two-fold. Before a class, instructors assign reading and other preparatory work to students and expect them to complete it. During the class, instructors avoid re-covering the textbook material and instead use exercises to engage students in higher-level discourse and learning of the material. The key mechanism that feeds both elements of this process is some kind of assessment (e.g. a quiz) just before class. This quiz motivates students to do their preparation and enables instructors to assess how well students learned the preparatory material and where to focus attention during the classroom work.

In CPSC 121, each (possibly multi-day) lecture ends with the preparatory assignment for the next lecture. This assignment comes in three parts: a list of "pre-class learning goals" that students should achieve before the next lecture, a list of readings that students can use to achieve those goals, and suggested exercises to assess their progress. In addition, before the next lecture, students are required to complete an online, low-credit quiz that assesses the pre-class learning goals. Ideally, these quizzes would be drawn randomly from an immense bank of questions all targeted at the relevant learning goals. For now, the quizzes are fixed. As a result, rather than resubmitting the quiz again and again with instant feedback, students are instead encouraged to look over the quiz as they begin their preparation (to focus their reading) and complete it at their leisure. Feedback comes only after the deadline, however. Beyond the automatically graded questions targeting pre-class learning goals, each quiz also includes 1-2 open-ended questions that get students thinking about the next lecture's in-class learning goals. These are graded purely for completeness.

Just before the lecture (i.e. "just in time"), the instructor aggregates the results on the automatically graded questions and summarizes results on a sample of the open-ended questions. Where students did poorly on the automatically graded questions,

the instructor dynamically introduces new discussion and problems to reinforce those concepts. The instructor usually summarizes students' responses on the open-ended question into a strategy guide that the class discusses before diving into closely related problems for in-class work.

In-class work is weighted heavily toward working and discussing practical problems, often with (anonymous, ungraded) clicker questions used to gauge students' progress and summarize their results. In the larger section (with ~90 students) two undergraduate TAs attend lecture to help the instructor facilitate problem solving sessions.

As an example, the pre-class learning goals for the first lecture on predicate logic are stated as follows:

> By the start of class, the students are expected to be able to:
>
> - Evaluate the truth of predicates applied to particular values.
> - Show predicate logic statements are true by enumerating examples (i.e., all examples in the domain for a universal or one for an existential).
> - Show predicate logic statements are false by enumerating counterexamples (i.e., one counterexample for universals or all in the domain for existentials).
> - Translate between statements in formal predicate logic notation and equivalent statements in closely matching informal language (i.e. informal statements with clear and explicitly stated quantifiers).

The online quiz asked 11 questions such as, "What can you say about the truth of the following statement: $\exists x \in Z, \exists y \in Z, \exists z \in Z, x^2 + y^2 = z^2$?" Students averaged over 80% on all but three questions; so, the instructor reviewed those three questions to begin the class.

The in-class learning goal was for students to be able to build statements about the relationships between properties of various objects using predicate logic. Problems may be drawn from real-world examples like "every candidate got votes from at least two people in every province" or computing examples like "on the ith repetition of this algorithm, the variable min contains the smallest element in the list between element 0 and element i". These will then segue to open-ended predicate logic representation problems, including one asked on the quiz (defining precisely what it means for a list to be sorted).

Since students already understood the basic terminology, as indicated by their quiz results, lecture focused on higher-level topics, including summarization and further development of the open-ended quiz question to precisely define what it means for a list to be sorted. Lecture culminated with students defining a predicate GenerallyFaster(a1, a2) that indicates when one algorithm is "generally faster" than another based on a predicate Faster(a1, a2, n) that compares the algorithms for particular problem sizes. Students invented several of their own definitions, including one that corresponds to the standard comparison of asymptotic performance for algorithms, and we discussed the strengths and weaknesses of each one.

Anecdotally, the instructor feels he has a clearer sense of when students are able to apply core course concepts successfully. In particular, rather than only discovering that students cannot solve real problems when they fail at assignments or exams, he can now discover and address students' difficulties early in the learning process. An anonymous clicker survey shows that students generally now see the relevance of labs to the course material, possibly due to the increased connection between hands-on problems and theory in lecture.

Finally, assessment results (e.g., the midterm) indicate that both sections that use a problem-based style in class are performing at least as well as the section using a more traditional lecture approach.

2. Problem-Based Peer Learning

In our Data Structures and Algorithms course (CPSC 221), we are faced with a wide range of topics that require students to transition smoothly between abstraction and implementation. This relationship can be difficult for students to understand without high-level problem solving practice paired with implementation practice. Traditionally, CPSC 221 has seen implementation techniques taught during labs in the form of an algorithm and a partially-implemented framework, which students complete under the guidance of a lab assistant. Higher level abstract concepts are then presented in a more traditional, lecture-style environment. Repeatedly, however, interactions with students (both exam-based and in-person) have shown a distinct lack of comprehension regarding the nature of abstraction versus implementation. In particular, students make inappropriate choices when deciding to address a problem at an abstract versus implementational level.

To address this deficiency, during our two-section January to April 2009 offering of CPSC 221, an experiment was conducted to test the efficacy of a non-standard lecture model. In our experiment, one section of the spring term was offered in the traditional format where classroom contact hours were spent presenting electronic slides (made available prior to the class), and clicker-style questions to help students review the material and assess their progress. The second section was delivered using an alternate model of presentation presenting new course concepts in two components: an "offline" component where students were expected to read and learn the prescribed concepts before class; and an "introduction via problem" component where students were expected to solve a moderately sized problem, or series of problems, in groups during class.

The offline-learning component was sometimes supplemented with a clicker-style quiz or short discussion administered at the start of the class to ensure students have done their preparatory readings and to identify any areas of confusion among the students (not unlike the JITT model described for the CPSC 121 above). After a brief discussion, the remaining class time was spent on the "introduction via problem" component using group exercises, which required the students to directly apply the concepts they were learning while the instructor monitored the progress of each group, offering hints and discussion points when appropriate. Some of the exercises were presented in the form of a series of tasks that progressively led the students from newly acquired foundations toward more in-depth applications, or a large-scale problem in which the students had to determine and apply the relevant concepts to the solution.

It should be noted that in the "introduction-via-problem" component, core concepts were introduced "cold", that was without any prerequisite reading and a minimal introduction by

the instructor. The students relied on their problem-solving skills and prior knowledge to solve the problem. As an example, students were given two unidentified sorting algorithms at the start of the complexity unit for which they were to answer the questions, "What do these algorithms do?" and "Which algorithm is better?" The ensuing group discussion served as a segue into complexity theory and how and why we wanted to compare algorithms.

Aside from different presentation styles, both sections shared the same instructor, course material, labs, assignments, exams, and learning goals. In both sections the course material was presented via three-hours of classroom-contact time, and two hours of lab time. Furthermore, both sections used learning goals as concept "book ends" presented at the start and close of each unit.

As of this writing, the final exam has not been administered yet. The final exam is composed of a question derived from a shared exercise that was run in both sections, and two questions for which the material explicitly appeared on an experimental exercise. Each of these questions had high-level and low-level sub-questions to test performance between the two sections given the differing delivery of content. Any significant differences between the sections will result in the lower section being scaled up to meet the higher section.

An end-of-term informal survey has now been completed in the experimental section, indicating a highly favourable response to the experimental classroom setting. While not conclusive, this nonetheless motivates this approach from the perspective of student engagement and enjoyment. And although the midterms have thus far been statistically insignificant in their differences (though the average was higher in the section delivered using the alternate learning model), the true efficacy of this approach is more likely to be felt in long-term retention. This effectiveness may already start to appear in the final exam as students are forced to revisit the material from the first half of the term. Plans are underway for a follow-up study in approximately five months to test retention between both the experimental and control sections.

The final outcome of this experiment will be analyzed on the basis of assessment outcomes between the two sections. Although the first midterm did not report a statistically significant result between the sections, the instructor reports that students did respond favourably to the learning environment, and there has been no push-back on the front-loading of course material. In addition, there are fewer students during office hours from the alternate learning model section.

3. BRACElet

BRACElet [1] is a multi-institutional investigation into the reading and comprehension skills of novice programmers. The basic idea is that as a student or beginning programmer transitions from novice to expert, their explanations of what a piece of code does also change.

For example when experts are presented with the following code sample:

```
bool bValid = true;
for (int i = 0; i < iMAX - 1; i++) {
  if (iNumbers[i] > iNumbers[i+1]) {
    bValid = false;
  }
}
```

and asked to explain what it does in plain English experts typically say the code checks to see if the numbers in an array are sorted [2]. Non-experts will typically engage in a line-by-line description of what the code does without relating what the body of code as a whole does.

Based on this idea we wanted to explore how student explanations changed as they progressed through our various first and second year programming courses. Assuming these questions, in some dimension, express a student's mastery of the concepts and techniques of a particular course, then these questions may serve as a baseline to measure the success, or lack thereof, of curriculum changes. In addition to tracking the progression of students from novice to expert, we also wanted to see if there was any relationship between a student's performance on these questions and their final course grade. As the first step in this study, we administered several "explain in plain English" questions to students at various levels of computing at UBC. Our initial data was gathered from:

1. Students who had just completed our primary introductory course in Computer Science (about 420 students).
2. Non computer science students who had just completed a second programming course, but whose first programming course was not as comprehensive as the course in (1) (about 140 students)
3. Students just starting the course in (2) (about 210 students)

In each case we provided our students with the same three code samples expressed in the primary programming language of that course. The samples were presented in order of increasing complexity. The first code sample swapped the value of two variables, the second computed the average of a collection of numbers in an array, and the third looked for the last occurrence of a character in a string. The first two questions were graded out of two. One mark was given for answers that provided a line-by-line description of the code with two marks awarded for answers that described what the code as a whole did. No half marks were awarded. Full marks were given for answers that weren't quite correct but were trying to explain what the code as a whole did as opposed to providing a line-by-line summary. The third question was graded out of three, with, again, one mark for a line-by-line description, two marks for a description of the code as a whole, and a third mark for indicating that the code was looking for the last occurrence of a character[1]. The questions were administered as part of the final exam for the first two student populations and as an anonymous quiz on the first day of class for the third group.

Although we have collected a substantial amount of data across the three identified student populations, the initial analysis has

[1] Students often realized the code was looking for a particular character but failed to realize it was the last occurrence being looked for.

focused on the students who had just completed their second programming course and whether or not these questions were good predictors of student course grade. Our hope was that the scores on the three "explain in plain English" (i.e. BRACElet) questions would either collectively, or individually, exhibit a strong correlation with either or both the final exam and/or course grade.

The approach we are using is to compute the correlation between the BRACElet question scores and the final grades in the course.

The data corresponding to questions B1, B2, B3 in Table 1 shows the results of this computation. The correlation for the three BRACElet questions ranges from 0.35 to 0.47. In addition the correlation of the sum of the three questions, 0.56, is provided. Although these questions all exhibit a positive correlation with the course grade they are only moderate predictors.

Since the BRACElet questions were not strong predictors (with correlation of 0.7 and greater) of a student's performance in this course we decided to expand our analysis to see if other questions on the final exam were better predictors. The rest of Table 1 lists all the questions on the final exam, their correlation to the student's final grade, and the number of marks the question was out of.

Question	Correlation to Final Grade	Maximum Score	Average Mark
Q1	0.64	10	7.16
Q2	0.7	20	10.52
B1	0.45	2	1.64
B2	0.35	2	1.76
B3	0.47	3	1.8
Q4	0.66	4	2.72
Q5	0.37	4	3.18
Q6	0.56	17	9.27
Q7	0.67	7	3.8
Q8	0.47	6	3.24
Q9	0.29	7	4.17
Q10	0.7	12	8.43
Q11	0.49	12	10.54
Q12	0.62	9	5.8
Q13	0.79	20	9.94
Sum of BRACElet Questions	0.56	7	5.2

(B1, B2, B3 are BRACElet questions.)

Table 1: Correlation between Questions on Final and Final Grade

3.1 Analysis of the Data

Our goal in looking at this data is to try to get a better understanding of the sorts of questions that correlate highly with a student's grade and the underlying skills required to be successful at these sorts of questions. The hope would be that if we can identify these skills we can then provide guidance to student's on how to improve these skills and hence their grade in the course. Similarly, if we find questions with a low correlation they need to be examined to see if there is something fundamentally wrong with the question, be it in the question's wording or in what we are testing for. In both scenarios the documented set of learning goals, which are given to students, are also used as part of the analysis. Students are also told that exam questions will always be mappable to one or more learning goals. In other words, the exam is in some sense a checklist of the level of knowledge and understanding of a subset of the course's learning goals. (There are usually too many learning goals to test all of them on a final exam.)

To illustrate the use of this data consider two scenarios:

1. Questions that have a high correlation to a student's final grade.
2. Questions that have negative or little correlation to the final grade.

Given these criteria three questions with a high correlation to the final grade (Q13, Q2, and Q10) and one with a low correlation (Q9) to the final grade are of particular interest. (The course instructor was pleased to see that no questions had a negative correlation.)

Question 13 had the highest correlation at 0.79. This question leads students through the process of developing a piece of code to add a new node to a linked list. The students are asked to write comments about what some code does, draw pictures of how the structure of the list changes, write a five-line function to add a new node, analyze the time complexity of the code they wrote, and finally compare this implementation to one that uses a different type of linked list. In essence this question touches on multiple aspects of the course material and different parts of the question require different levels of understanding of the material. It also has the property that later parts of the question can be completed without getting earlier parts correct.

In question 10, students are given snippets of C++ code and asked to label a picture of program memory. For example students might be expected to indicate where variable X is located and what value X has. If a pointer is involved, students would be expected to indicate the memory location the pointer points to. In total there were 6 snippets of code of varying degrees of complexity with respect to the resulting memory layout. Again like question 13 this question touches on multiple aspects of the course material. Some diagrams are relatively simple while others are fairly complex. To get full marks students would need to have a good understanding of the relationship between C++ code and how memory is used.

Finally, question 2 also shows strong correlation. Question 2 was comprised of a series of sub-questions requiring at most a short sentence to answer. The questions cover the breadth of the course material and typically focus on a key piece of information about a particular topic. An example of this would be a question that asks

under what conditions would Quicksort exhibit the same running time complexity as a simple bubble sort.

What do these three questions have in common? Not surprisingly they are all multi-part questions. In addition the sub-questions are not tightly tied to each other. That is, the inability to do one sub-question does not preclude being able to answer other sub-questions. Since both questions 10 and 13 explore one area at several levels of detail, one might refer to this as a breadth of understanding about a topic area. Question 2 has this same property of breadth but from the perspective of key ideas throughout the course. This observation raises the question as to the role of breadth play in a student's success and whether the ability to exhibit breadth of understanding in one area implies a breadth of understanding in another area? We might be able to gain some insights into this by looking for correlation between questions. For example is there a strong correlation between these three questions? We have not looked into this in detail, but our preliminary analysis suggests there is not. Taking this further, we also intend to take subsets of questions and explore the correlation between subsets of questions and final grades.

As indicated earlier, another area of interest is questions that have a low correlation to the course grade. From this exam, question 9 is a candidate. This question focuses on merge sort and asks a number of questions about how the sort works. Although this is a multipart question it has the property that if a student doesn't understand one part of the question it is likely that other parts of the question can't be completed. In some sense one can say that the question lacks breadth. Also, given the relatively low average mark for this question it is important to check the learning goals to ensure that the question is indeed testing students at the appropriate level for this material. In this case they were. Armed with this information the course instructor has a number of options:

1. Re-work the question so that it tests for a broader understanding of the topic.
2. Change the course learning goals to better reflect the expected learning outcomes and revise the question as necessary.
3. Change or augment the way the material is taught.
4. Do nothing and treat this as a question that only the top students in the class are expected to get.

Note that having a low correlation to a final grade is not necessarily a bad thing. For example Q5 has a low correlation (.37) but the average is relatively high (3.18/4). In examining the question and the learning goals it was determined that this was an important question and the material was not tested anywhere else. It just happened that most students did quite well on this question, even if they didn't have a good course grade.

Going forward, the questions with high correlation to final course grade (Q2, Q10, and Q13), or very similar ones, will be administered on the final exam for this course in April. From this we hope to see if these questions continue to be strong predictors given the difference student backgrounds in this offering of the course. Although the BRACElet questions were not strong predictors of a student's performance in this course, the questions provide useful metric for ranking students across the different courses. As a result, we will again be administering the BRACElet questions and we will also have the opportunity to compare the class's results with the same questions administered at the start of term.

As we gather and analyze more of the data we will be able to make changes to our teaching and course delivery methods in an effort to improve outcomes. The work we have described here will provide a way to measure our progress. In addition, we can also provide guidance to our students with respect to the things they can do to aid their learning. For example in this course we exhort our students to always draw pictures of data structures and their in-memory representations. Based on the results of our analysis (Q10) we can tell students in this class that the ability to draw out memory representations seems to be an important skill for course success. We know that students who can't draw memory representations do poorly. Unfortunately we don't know if drawing memory pictures causes students to have a better grasp of the course material or if it is the case that a better understanding of the material makes picture drawing easier. Knowing the answer to this and other similar questions would allow us to tell students how to be successful in this course and we hope to perform additional studies to answer these sorts of questions.

4. Attitudinal Surveys

Attitudinal surveys provide valuable information on the student perception of the course and their learning. This includes their perception of the level of difficulty of the course, their progress in the course, the way the material has been presented, the forms of assessment, their expectations, their self-efficacy, etc. While attitudes about the course can change in a very short time, attitudes about the students learning and expectations can provide useful insights to the instructors. We have conducted attitudinal surveys of students in each of the four levels of the CS program at UBC. Some of the questions asked include:

- What are your three biggest expectations from this course?
- How do you study for this course?
- How have you been using the learning goals for this course?
- Which of the following do you feel are most important to you when you work on a particular difficult assignment:
  - Regular meetings with TA / Prof to discuss about the assignment.
  - Bouncing off ideas with friends.
  - Sample code examples.
  - Regular feedback on progress.
  - Access to web resources.
  - Lecture / Assignment material.
  - Advice from students who have completed the course.
  - Help in the lab while working on the assignment.
  - Drop-in help at TA office hours.
- How do you know when you have learned something?

Responses from these questions allow instructors to have greater insights in the corresponding areas:

- Whether students are intrinsically (e.g. want to learn something) or extrinsically (e.g. taking a course primarily to satisfy program requirement) motivated in this course.
- Whether the students are spending appropriate and reasonable amount of time for this course.
- Whether the learning goals are well defined to guide the students in their learning.

- Whether appropriate support is made available to the students in their learning.
- Whether the students are developing appropriate meta-cognitive skills.

As we have developed learning goals for a number of courses in our program, we were interested in finding out whether the students are using these learning goals in their study, and if so, whether this has any correlation with their course grades. We were also interested whether students who are intrinsically motivated score higher in midterms / final than those who are extrinsically motivated. In both cases, there is a slight correlation. Students, however, are not usually good predictors of their own performance. When asked what final grade they expected to get, almost none indicated that they would fail the course, although, many of them had failed the first midterm. Also, when students were asked how they studied for their course, most answered that by reading the textbook or notes. This may not be the most effective ways in studying, especially for long term retention, where research shows that repeat testing is more effective [4].

While attitudinal surveys provide valuable insights on student attitudes and perceptions in a number of areas, insights into their unique background and circumstances can be gained through follow-up one-on-one interviews. These interviews are best conducted by an education researcher rather than by the instructor so that the students may freely express their opinions and point of views without concern of any impact on their grades. One of the questions that can provide particular insights for the instructors is how the students approach each question on the midterm or exam. The responses may reveal what strategies or the sources of material they use and this may provide insights for the instructors on what components of the course the students deemed to be useful for their learning. As an example, in one of the courses, students find that the labs are helpful but not so much for the assignments. A follow up investigation reveals that the assignments are too big and are worth too little towards the final course grade. In another course, clicker questions are found to be helpful in particular types of questions on the midterm. Such information is useful for the instructors for future offerings of the same course.

5. Conclusion and Future Work

In this paper, we have described a number of education research activities in UBC CS courses to improve student engagement and learning. We have collected student performance and attitudinal data in these courses and made some initial conclusions. We plan to correlate this data with similar data to be collected in the future terms for a longtitudinal study, especially in the area of long term retention. We will also correlate the data with students' prior education background and experience to ensure students success in our CS program.

6. REFERENCES

[1] BRACElet. 2006. BRACElet. Retrieved on February 19, 2009 from http://online.aut.ac.nz/Bracelet/repository2.nsf/HomePage?OpenPage.

[2] Lister, R., Simon, B., Thompson, E., Whalley, J., 2006. Novice Programmers and the SOLO Taxonomy. 11th Conference on Innovation and Technology in Computer Science Education (ITiCSE).

[3] Simon, B. 2008. Computer Science Learning Goals. Carl Wieman Science Education Initiative at the University of Britist Columbia. Retrieved on February 19, 2009 from http://www.cwsei.ubc.ca/departments/computer_learning_goals.htm.

[4] Roediger, H.. Karpicke, J. 2006. Test-Enhanced Learning, Taking Memory Tests Improves Long-Term Retention. *Psychological Science*. 17(3), pp249 – 255

[5] Novak, G., Gavrin, A., Wolfgang, C., Patterson, E. 1999. *Just-in-Time Teaching: Blending Active Learning with Web Technology*. Upper Saddle river, NJ: Prentice Hall.