Just-in-Time Teaching for CS0

Tammy Bailey
Department of Computer Science
Duke University, Durham, NC 27708-0129
tammy@cs.duke.edu

Jeffrey Forbes
Department of Computer Science
Duke University, Durham, NC 27708-0129
forbes@cs.duke.edu

ABSTRACT

Just-in-Time Teaching (JiTT) is a teaching and learning strategy based on the interaction between web-based study assignments and an active learner classroom. The essence of JiTT is the *feedback loop* formed by the students' preparation outside the classroom that shapes their in-class experience. The goal of JiTT is to use feedback to guide teaching and to empower and motivate learners. This paper describes a successful implementation of the JiTT strategy for an introductory computer science course.

Categories and Subject Descriptors

K.3.2 [**Computers & Education**]: Computer & Information Science Education—*Computer Science Education*

General Terms

Human Factors, Design

Keywords

Active learning, CS0, JiTT, non-majors, pedagogy

1. INTRODUCTION

Students in an introductory computer science course are often not aspiring computer scientists. They may be motivated by curricular requirements or curiosity. Instructors must develop a curriculum of sufficient breadth, depth, and rigor along with innovative and creative teaching methods to engage a population of students with diverse knowledge and learning styles. Maintaining student interest while simultaneously providing understanding and appreciation of the course material is an ongoing challenge. Students unfamiliar with the concept of computing as a science often find the combination of theoretical and technical concepts difficult, uninteresting, or of little personal benefit. As such, non-majors tend to lack the motivation required for acquiring the knowledge and skills fundamental to the course material.

Many studies have posited that students are more likely to retain knowledge acquired via active learning strategies such as discussion and practice rather than passive learning strategies such as reading and lecture [5, 7, 11]. Just-in-Time Teaching (JiTT) is a teaching and learning strategy based on the interaction between web-based study assignments and an active learner classroom [10]. Students complete web-based assignments that are submitted online prior to the upcoming lecture. The instructor reviews the assignments "just-in-time" to structure the lecture in accordance with the level of understanding conveyed by the student responses. The subsequent web assignment is then motivated by the in-class discussion. The students' preparation outside the classroom, their experience inside the classroom and the feedback between the two form the *feedback loop* that is the essence of the JiTT strategy.

JiTT was developed in response to the observation that students are not learning as well as they could. In many cases, students focus on their final grade in the course rather than the acquisition of knowledge or skills. In our experience, such cases arise frequently when teaching at the introductory level. JiTT has been shown to be of benefit to courses that students consider to be of secondary importance to their lives or education [10]. The JiTT strategy was first implemented in introductory Physics courses and is now used with success in courses in Mathematics, Biology, Chemistry, and Engineering [8, 9]. JiTT is not widely used in computer science courses and is fairly uncommon at the introductory level, despite the fact that many courses in the area make extensive use of the Internet as a teaching, learning, or communication tool. The use of JiTT in non-introductory undergraduate computer science courses is discussed in [2, 6].

In this paper we present an implementation of the JiTT strategy as described in [9, 10] for introductory computer science courses. This strategy was developed over a number of semesters and implemented when teaching CS0 in Summer 2004. We provide several examples of web assignments and classroom activities for a variety of subject areas.

2. MOTIVATION

At Duke University, Principles of Computer Science (ACM CS0) is designated as a course fulfilling a curriculum requirement and is generally populated with students majoring or intending to major in non-scientific fields. The course curriculum is a result of an ongoing effort at Duke to teach computer science to a diverse audience, addressing a broad selection of topics while maintaining depth and rigor [4], in-

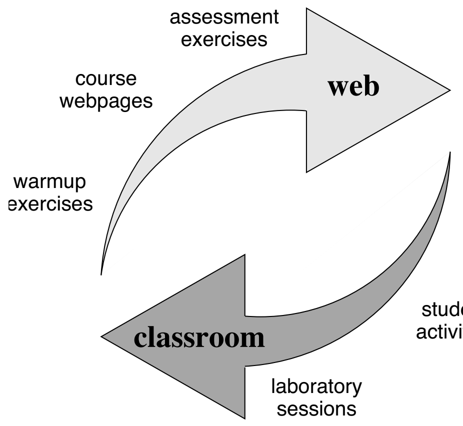

Figure 1: The JiTT feedback loop in CS0.

corporating active learning strategies in laboratory sessions to master technical and theoretical concepts [12], and integrating a series of classroom and web-based activities in which the students take an active role in researching and learning the societal impact of the current technical subject studied in class [3]. While our recent efforts [3, 12] were rewarded with increased student interest, participation, and satisfaction, our main goal as educators is making sure the students learn. Evaluation of students' written examinations show there are still many students who lack understanding of basic concepts or the ability to apply these concepts to unfamiliar problems.

3. METHODOLOGY

The essential element of the JiTT strategy is the feedback between the out of class course work – the *web component* – and the material presented in the classroom, the *classroom component*. In our course, the web component consists of warmup exercises (preparatory and prelab assignments), course web pages (informative, enrichment, communication), and assessment exercises (homework and laboratory assignments). The classroom component consists of lectures, laboratory sessions, and in-class student activities. Figure 1 illustrates the JiTT feedback loop in CS0.

3.1 Web component

Outside the classroom, the students' portal to the class is the web, providing round-the-clock access to course-related materials and a communication channel between the course participants.

3.1.1 Warmup exercises

Warmup exercises are characterized by a clear set of learning objectives, an explanation of unfamiliar terminology, and questions designed to make students confront their previously held notions, stimulate thought and interest, and to create a bridge to concepts introduced later in the course. The exercises ensure that students are familiar with the material prior to class and are aware of the important concepts. Warmup exercises are graded on effort, not necessarily correctness. Students are given credit for correct solutions, incorrect solutions based on correct reasoning, and for providing a clear explanation of where they reached an impasse. Collaboration is encouraged on these assignments.

Preparatory assignments are short, web-based assignments due a set amount of time before lecture. Links are provided to optional supplemental readings that either discuss the subject in greater detail (for students having difficulty with the assignment) or explore the topic in greater depth and context (for students who would like to learn more). The nature of the assignments varies. The assignment can be questions on the reading, solving a simple instance of a problem, or discussing their experimentation with an applet illustrating a particular concept. Whenever possible students are asked to discuss the relation of the preparatory assignment material to material previously learned in class and how they could use what they had already learned when solving the new problem. Students may also provide feedback on the difficulty or appropriateness of the assignment. Questions or comments on the assignment that would be of benefit to the class as a whole are posted on the discussion forums.

Prelab assignments are completed in preparation for the weekly laboratory sessions, and are due prior to the lab. As the laboratory sessions mainly concentrate on computer programming, these assignments generally ask students to formulate pseudocode algorithms for problems that have already been studied conceptually in lecture.

3.1.2 Course web pages

Three types of pages are accessible from the course web page and fit one of three categories. *Enrichment pages* are web sites and links, typically illustrating relevance of course material to everyday life or on topics related to the course material. The enrichment pages are updated throughout the semester and help students answer the questions: what is computer science good for and what is new and interesting in the field? *Information pages* are general course and instructor information, lecture notes, assignments, required reading, and syllabus. These pages also include useful links to resources, such as campus computer labs, instructions for software installation and accessing university network, academic calendars, codes of conduct, and acceptable use policies. *Communication pages* provide discussion forums for student-student and student-instructor conversations and anonymous feedback.

3.1.3 Assessment exercises

Homework assignments are available online and are either graded automatically or require written solutions that are turned in at class time. Homework assignments are strictly graded on correctness, and are thus only assigned after the material has been presented and discussed in class. Students are again encouraged to collaborate, but are told they must write and submit their solutions independently. These assignments contain 2-4 questions that are similar to what a student may expect to see on an exam. *Laboratory assignments*, assigned at each laboratory session, are discussed in Section 3.2.3.

3.2 Classroom component

Our course is split into three fifty minute lecture sessions and one seventy-five minute laboratory session per week. The lab session is designed to be a smaller hands-on experience and has an enrollment cap of thirty students. In both lecture and lab, active learning and peer instruction techniques are employed.

3.2.1 Lectures with PRS

The warmup exercises motivate and structure the classroom experience. By reviewing the warmups, the instructor can uncover misconceptions and determine what concepts need reinforcement. The key ingredient of an active classroom is preparation, both for the students and the instructors. After completing the warmups, students can better complete in-class exercises. The instructor is better able to assign appropriate work after reviewing those exercises.

An effective JiTT tool for the classroom is peer instruction. We use Personal Response System (PRS) quizzes along with mini-conferences to assess student mastery and participation. At various points in lecture, the instructor will pose a question along with possible answers. An example question on expression evaluation in Java is below.

Consider the following expression:

```
double C = (F-40.0)*(5/9);
```

If the value of `F` is -40.0, then what is the value of `C`?

a. -40.0
b. 0
c. 40.0
d. error
e. none of the above

The students answer the question by pressing the corresponding numbered button on their PRS transmitter. The student can also indicate his or her confidence (high, normal, low) in their response. We find that using PRS is a far more reliable measurement of overall class comprehension than asking students to raise their hands or volunteer answers. The PRS software records student responses and then graphs the distribution of answers immediately. If the vast majority of students selected the correct response, the instructor can demonstrate the expression evaluation in a program and simply ask the class for an explanation.

PRS is particularly useful when there is some question about the correct response. In that case, students are given time to confer with each other and justify their reasoning to their peers. In discussing the answer to the above question, students bring forth their knowledge of operator precedence, data types, automatic type conversion, and Java syntax. When students are asked to give their answers again, the answers usually converge towards one answer. Of course, that answer is not always the right one which can make for an even more enlightening discussion. In this case, we can demonstrate the expression evaluation in a program.

3.2.2 Student activities

Other forms of active classroom engagement also fit within the framework of the JiTT approach. Formal in-class debates also provide a forum for active participation and peer instruction [3]. Debates provide a context for the social aspects of computing discussed in the class. Contests, such as the Othello-playing agent tournament, provide an opportunity for students to demonstrate and describe their work to their peers.

3.2.3 Laboratory sessions

The laboratory sessions provide an opportunity for students to experiment with algorithms learned in class. Students either write short programs or participate in hands-on labs as described in [12]. Hands-on labs sometimes do not involve computers at all but rather involve physically manipulating objects while solving problems or simulating algorithms. One example is using posts and disks to solve the Towers of Hanoi problem. The students are guided primarily by undergraduate teaching assistants who are familiar with the misconceptions and difficulties novices face.

4. EXAMPLES

In this section, we provide examples of exercises and activities for a variety of topics discussed during the course.

4.1 Week one

To introduce students to the JiTT structure, a preparatory assignment is given at the first lecture. The students complete the assignment at the beginning of class. The instructor takes a short break to review the responses, and the remainder of lecture is spent discussing the responses.

1. Write down all interactions you have had with a computer today.
2. Would your life change drastically if all computers stopped working? Explain briefly.
3. Complete the following sentence: Computer science is the study of ________________.
4. To the best of your knowledge, what do computer scientists do?

These questions serve as a good way to encourage class participation from the start, illustrate the importance of computing and the effect is has on their everyday lives and address misconceptions about computer science. Invariably, half of the students will respond to the third and fourth questions with *"computers"* or *"programming"*. The instructor should not only clarify these misconceptions but use them to lead the class to a discussion of problem solving and algorithms, which will be the focus of the upcoming lectures.

In the next preparatory assignment, students are asked to read excerpts from George Polya's book on the concept of problem solving [13] and are given an assortment of classic puzzles such as this one:

Tom has three boxes of fruit in his barn: one box with apples, one box with oranges, and one box with both apples and oranges. The boxes have labels that describe the contents, but none of these labels is on the right box. How can Tom, by taking only one piece of fruit from one box, determine what each of the boxes contains?

In lecture, the instructor presents solutions to the puzzles, emphasizing problem solving strategies and addressing common mistakes. This introduces and motivates the upcoming lectures on algorithm design.

4.2 Encryption

A simple example of a substitution cipher is the *Caesar cipher*, which is named after Julius Caesar – not because he invented it, but because he is believed to have used it to communicate with his army. Caesar secured his messages by shifting each letter in his message three letters to the right. After informing his generals of the shift value, he was able to send them secured messages. The Caesar cipher is a a type of substitution cipher called a *shift cipher*, since the ciphertext alphabet is derived from the plaintext alphabet by shifting each letter a certain number of positions, called the *key*. Caesar's cipher uses the key 3. In general, shift cipher keys may be any positive or negative integer value.

> The following message is encrypted using a shift cipher with key 15:
> `CTKTG IGJHI P SDV LXIW DGPCVT TNTQGDLH`
> Decrypt the message.

> Suppose you received the following message:
> `XII VLRO YXPB XOB YBILKD QL RP`
> You were told the message is encrypted using a shift cipher but you forgot the key. Decrypt the message. What is the encryption key?

Most students will be able to successfully decrypt both messages. These preparatory exercises illustrate that if the encryption algorithm is known, then simple encryption algorithms are easy to break. As a class activity (or contest), students are given a message encrypted using a substitution cipher based on a random permutation of the alphabet. Students work together in groups to decrypt the message without knowing the substitution pattern. The objective is to realize that even though there are 26! possible substitution patterns, the cipher is still relatively easy to break using characteristics of the language such as letter and word frequencies and context to reduce the solution space.

The historical example of the Caesar cipher leads to a discussion of modern encryption techniques with particular emphasis on RSA cryptography. The warmup question posed to students is as follows.

> The ability to communicate securely with people all over the world has numerous applications. Is public key encryption the most important invention of the latter half of the twentieth century? Your response should discuss the effect of secure communication on society. In particular, you should consider how RSA along with the Internet has affected different aspects of civilization such as commerce, science, government, and education.

A more technical question used in a homework assignment illustrates the importance of choosing large primes.

> If we encrypt a message using the public key $(5, 299)$ and know that the RSA algorithm was used in generating the keys, what private key can be used for decryption?

4.3 Artificial Intelligence

Consider a scenario where a human judge engages in a natural language conversation with two other parties, one a human and the other a machine. If the judge cannot reliably tell which is which, then the machine is said to have passed the Turing test. A group of programs were developed in an attempt to "pass" Turing's test. The most famous such program, Eliza [14], was one of the first programs that attempted to communicate in a natural language by engaging humans in conversation with a simulated Rogerian psychotherapist.

An applet implementing the Eliza program is available to the students, who then answer the following questions.

> Take some time to chat with Eliza. Would you believe that you are conversing with a human rather than a machine? How long would it take you to be reasonably confident that you were not conversing with a human? Explain your answer.

This warmup can be used as both a preparatory and prelab assignment. In lecture, the student responses illustrate conversational patterns that "break" Eliza, providing an introduction to grammars, parsing, and natural language processing. In lab the students build upon the Eliza grammar, which is simply a file parsed by the Eliza applet.

Eliza was written in 1966 and is relatively simple, using a small database of words and phrases and applying a series of pattern matching rules to the human's statements when forming its replies. Since then, many more complex programs have been developed to simulate conversation. These programs are called *chatterbots*, and their goal is to make a human believe they are engaging in conversation with another human, at least temporarily. One such chatterbot is Alice [1].

> Take some time to chat with Alice. She will tell you she is a robot. How long would it take you to be reasonably confident that you were not conversing with a human pretending to be a robot when you chatted with Alice? Explain your answer.

The chatterbot Alice introduces the reverse Turing Test, that is, one in which the subjects attempt to appear to be a computer rather than a human. In addition, the assignment leads to a classroom discussion of modern artificial intelligence and applications to programs commonly in use including speech recognition, spam filtering, robots, software agents, and game playing.

5. ASSESSMENT

Our first efforts implementing JiTT have been at a small scale in a summer session course with no formal assessment. Our qualitative assessment is given below.

Positive

Students tended to best retain concepts related to assignments and activities that they preferred. The most popular assignments were related to artificial intelligence and game playing, and students performed well on exam questions related to parsing, the Turing test, search trees, and the minimax algorithm. The Towers of Hanoi activity was also noted as being very helpful in demonstrating the concept of recursion, and many students found that while programming the algorithm was difficult, once they discovered the correct solution it made formulating recursive problems much easier throughout the semester. Students generally had more success with concepts that could be illustrated with interactive applets, such as Towers of Hanoi, mazes, fractals, and sorting algorithms. Many students found the supplemental readings on ethical and controversial topics (such as file sharing, cracking and cyberstalking) interesting and read them on their own time, even though they were not required or discussed in class. Students were much in favor of receiving credit for effort on the warmup exercises. Students were also appreciative that failing to understand a particular warmup exercise despite a genuine effort to do so translated to an emphasis of the material in lecture rather than a poor grade on the assignment. Such students were more apt to ask for clarification during lecture and leave class with a better understanding of the material. In general, students felt the problem solving and logical reasoning skills they acquired were of significant benefit and would be useful in their future endeavors. The students were asked about the effectiveness of the course and a representative response is below:

"The web assignments were effective in making me read through the text beforehand so that I would better understand what would be taught in class the next day."

"They gave us a good idea of what to expect during class and of the concepts we were meant to grasp. Often, if I found a web assignment confusing, it let me know what I still needed to understand."

"I like the fact that working with others is encouraged, I believe that it helps a lot when you work with others. I like that we have web assignments and other assignments other than just the tests."

"This is an entirely new subject to me, and one I've never had much interest in, but I see now that it's definitely a creative and interesting field of study."

Negative

The main complaints were the amount of out-of-class work and the pace of the course, although this was noted by some students to be partly because this was a summer course. Providing too many links to supplemental and advanced readings was problematic for a small number of students who felt obligated to read them all on a regular basis. Allowing student collaboration fostered plagiarism in some cases. This abuse mainly occurred on problem solving questions rather than those that were conceptual in nature, although students said they benefited most from the former and least from the latter. As the majority of class assignments were only available on the web, there were recurring issues with Internet access, in particular for students living off-campus. Students also did not always see the connection between the web assignments and the course material, so care must be taken in writing the questions.

6. FUTURE WORK

While student response has been positive, a thorough assessment of the efficacy of JiTT in introductory computer science courses is still required. There are a variety of questions that need to be asked, including:

- Which parts of the model add the most value?
- Does this model encourage students to take future computer science courses?
- Do students from a course like this one actually exhibit better problem solving ability?

JiTT can be instructor and student labor intensive. The students have to regularly turn in assignments and the teaching staff has to review and grade them. Teaching a JiTT course requires a significant library of examples and problems. The web assignments and PRS quizzes have to be crafted very carefully to be effective. Also, the enrichment pages in the web component require continual attention. JiTT can be very time consuming for the instructor compared to a traditional lecture based class, but we hope that after a few iterations, we will have a reasonable array of materials for the web and classroom components. Our goal is to create a resource for other instructors, so that they can apply JiTT in their classes.

There are a number of logistical issues with JiTT that require attention. One issue is how well this system will scale to larger classes and different kinds of institutions with less technical and teaching staff support. Methods for allowing collaboration while minimizing the opportunity for plagiarism are discussed in [10]. In the warmup exercises, credit for effort must be well-defined. Some students tended to abuse this grading system. One idea is to provide several examples of "correct" wrong answers as opposed to "incorrect" ones.

7. CONCLUSIONS

Motivating and engaging students in introductory computer science courses is crucial, particularly in non-major survey courses. Just-in-Time Teaching provides a sensible framework for an active learning environment by blending out-of-class work with lectures and in-class activities. JiTT has been hailed in many of the sciences and is suited for computer science education as well.

8. REFERENCES

[1] The A.L.I.C.E. Artificial Intelligence Foundation. `http://www.alicebot.org`.

[2] O. L. Astrachan. Non-competitive programming contest problems as the basis for just-in-time teaching. In *Proceedings of the 34th Annual Frontiers in Education Conference*, 2004.

[3] T. Bailey and J. Forbes. Computers and society in CS0: An interactive approach. In *Proceedings of the 34th Annual Frontiers in Education Conference*, 2004.

[4] A. W. Biermann. Computer science for the many. *Computer*, 27(2):62–73, 1994.

[5] C. C. Bonwell and J. A. Eison. *Active Learning: Creating Excitement in the Classroom*. ASHE-ERIC Report Series. Jossey-Bass, 2000.

[6] R. Fleisher. Just-in-time: Better teaching in Hong Kong. In *Proceedings of the Second Teaching and Learning Symposium*, 2004.

[7] R. R. Hake. Interactive-engagement vs. traditional methods: A six-thousand-student survey of mechanics test data for introductory physics courses. *American Journal of Physics*, 66(1):64–74, 1998.

[8] J. Handelsman, D. Ebert-May, R. Beichner, P. Bruns, A. Chang, R. DeHaan, J. Gentile, S. Lauffer, J. Stewart, S. M. Tilghman, and W. B. Wood. Scientific teaching. *Science*, 304(5670):521–522, 2004.

[9] Just-in-Time-Teaching. `http://www.jitt.org`.

[10] G. M. Novak, A. D. Gavrin, W. Christian, and E. T. Patterson. *Just-in-Time Teaching: Blending Active Learning with Web Technology*. Prentice Hall, 1999.

[11] J. G. Penner. *Why many college teachers cannot lecture*. Charles C Thomas Publisher Ltd, 1984.

[12] S. Pollard and J. Forbes. Hands-on labs without computers. In *Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education*, 2003.

[13] G. Polya. *How to Solve It?: A New Aspect of Mathematical Method*. Princeton University Press, 1957.

[14] J. Weizenbaum. ELIZA–A computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(1):35–36, 1966.