Usage Simulation for Evaluating Educational Materials

Viera K. Proulx
Northeastern University, Boston, MA 02115
vkp@ccs.neu.edu

Joseph W. Proulx
University of Colorado - Boulder, Boulder, CO 08310
proulx@ucsu.colorado.edu

Abstract

In this paper we describe how usage simulation can be used to evaluate educational software. Usage simulation, typically conducted by an expert acting as a typical user, provides a detailed review of problems encountered together with suggestions for fixing them or for making additional improvements. We used this method to evaluate the laboratories for the first semester of introductory computer science course. To make sure the evaluator will identify typical problems that a novice programmer may encounter, our evaluation was done by a student familiar with the material evaluated, but otherwise having only a limited programming experience.
We describe how the evaluation was conducted, present the results we obtained, and identify the benefits of conducting usage simulation of educational software in general.

1. Introduction

After working for several years on educational software our Computer Science Education Research Group at Northeastern University (Harriet J. Fell, Viera K. Proulx, and Richard Rasala) has built a large collection of laboratory materials for teaching introductory computer science. The materials have been class tested and we received feedback from other instructors that used them. Still we felt a more systematic evaluation was needed before the project could be published (on the web or commercially). Motivated by the industry practice of user-testing software either by experts who act as naive users, or by users themselves, we decided on a similar method.

We start our discussion by identifying the need for this type of evaluation of educational materials. Next we describe the evaluation process. We show how this type of evaluation differs from either evaluation done by students taking the course, or from a peer review and finally, identify the benefits this kind of evaluation can provide.

2. The need for user testing of educational software

2.1 Type of evaluation typically conducted

Typical software is written for a particular type of user. The designers study the expected user behavior and design the interfaces, functionality and documentation to satisfy the intended user. Users are involved in testing the software written for them before the software is released. This testing may have several different goals, for example usage simulation [6], acceptance testing [1], or system testing [1]. Preece [6, p. 673] comments on the usage simulations:

[Often] 'these tests are done by experts, who simulate the behavior of less experienced users' ... 'The main reasons ... are to do with efficiency and prescriptive feedback' ... 'Often little prompting is needed to get the reviewers to suggest possible solutions to the problems identified' ... 'Generally, they provide detailed reports based on their use of the prototype or working system.

Software designed for classroom use is typically 'class tested'. This means that the instructor who created the software and the supporting materials uses it in teaching a class of students, and if it seems successful, offers the product for use to other instructors. The author may also collect data from the students when the materials are used in the class - e.g., by questionnaires, by observing student's reactions, or by requesting written comments on completion of each lab [4]. If the materials are collected into a published text or a CD-ROM, additional feedback is obtained from other instructors who 'test' the course materials in their classrooms.

In many cases a more structured evaluation of course materials is conducted. For example, the instructors collect end-of-semester evaluation forms from all students, get weekly feedback from other course instructors and teaching assistants, and from the faculty teaching subsequent courses to the same students [5]. More formal studies involve a comparison of two groups of students - a control group and the group that uses the course materials [3]. Boroni et. al. used high school students to test the effectiveness of their visually based DynaLab [2]. The result of this test made it clear that the intended educational outcomes were achieved.

The creator of materials may also write a paper for a professional journal or a conference describing the materials and their pedagogical value. Such papers are peer reviewed, providing an additional feedback to the author.

2.2 Problem with these types of evaluations

The peer reviewer of journal papers can comment on the perceived quality of a particular subset of the complete work - namely the part described in the paper. While it is possible to see that some ideas are new, pedagogically sound, exciting, and motivating, without seeing the actual details and observing students in the context of the class or lab, it is hard to evaluate the effectiveness of the actual presentation of the ideas to students.

Classroom testing may show that some of the materials are more accessible or exciting, or better written. However, if we make mistakes, omit something, do not explain something clearly enough, students struggling with the materials for the first time and fighting the time pressures of all their coursework often perceive it as their shortcomings. They may like one lab more than another one, but it is hard for them to see that our approach may have been at fault. Even if they think we may be wrong, they are not in a position to suggest a suitable correction or alternative. This is especially true in introductory courses where everything students see is very new and quite different from their previous experiences. Weekly feedbacks from other course instructors and teaching assistants will also reflect only the immediate reaction to the project of the week.

The testing by high school students [2] showed the effectiveness of the materials but did not provide a feedback on the writing style and presentation. It also focused on one topic, rather than a whole suite of laboratories.

2.3 Employing usage simulation for evaluation of curriculum materials

Over the past several years we have built a sizable collection of introductory computer science course materials - labs, toolkits, projects, and exercises. After converting to C++ from Pascal, introducing object oriented design concepts into the course, and upgrading several times to new versions of the compiler, we needed to make these materials ready for wider distribution. We felt, we needed to carefully review especially the first semester materials, to make sure all concepts are presented in an appropriate sequence, explained in sufficient detail, and follow a sensible path. We decided that we needed an evaluation by an outsider not closely familiar with the project. We also realized, that any evaluation by a highly experienced computer programmer will not catch the types of mistakes that are the biggest stumbling blocks for our freshmen. Experienced programmers would fill in the missing points without realizing it - just like we may have done when writing the materials.

We modeled the evaluation process on the usage simulation method, with the evaluator acting in the role of a novice student. Selecting a competent evaluator can be quite difficult Our selection was serendipitous. Joseph Proulx, son of one of the members of the research group had just completed his first year at the University of Colorado, Boulder and was home for the summer. He has taken a one semester course in C++ programming and so had fresh recollections of the difficulties he encountered in learning the subject. He has good analytical and writing skills and a passion for discussing technical and pedagogical ideas with others. We had a good rapport and knew he would not hesitate to offer valid criticism. Harriet Fell offered to use funds from her professional development fund to pay for Joe's work.

We now describe the evaluation process and follow by assessment of the results.

3. The evaluation process

We did not feel the evaluator needed any instruction or lectures in the concepts or skills needed to do the projects. As needed, we clarified technical details peculiar to our environment but otherwise he worked independently. To provide guidance, we divided the evaluation of each lab into two phases: working out the solution and reflecting on this experience.

For each lab the evaluator read all the lab materials and worked out the solutions, just as a student in class would. During this time, he kept records on the time spent and wrote comments both in a journal and on the margins of the lab handouts. A Lab Evaluation Checklist (see appendix A) provided a framework for this evaluation. This checklist was deliberately quite detailed. The goal was to point out the different types of problems - writing, goals not clearly defined, tutorial part unclear, missing or confusing instructions, instructions too detailed so student had little thinking to do, etc. The goal was to make sure the evaluator felt free (and responsible) to comment on all aspects of the laboratory materials.

Upon completion of each lab the student reported his findings, comments, and criticism by writing a Lab Evaluation Report (see appendix B). The first part of the report included the length of time the evaluator needed to read the lab assignment and to actually work on the required programming (design, implementation, and debugging), followed by a list of specific problems encountered while reading the lab or while working on the programming solution. The next part of the report contained free form comments or editorial suggestions most appropriate for the given situation. The Lab Evaluation Checklist was used as a guideline for the kinds of questions that needed to be addressed. The evaluator was allowed (and encouraged) to ask questions while working on the labs. The numerous discussions we had during the evaluation process supplemented the written reports in a significant way.

3.1 Sample comments and evaluation feedback

The results of this evaluation were quite impressive. Our evaluator had a number of substantial comment for the first few labs - the ones that come at the most critical stage. To understand the depth of the evaluation and the quality of the feedback we received, we list below a selection of comments from the written reports, and comment on the impact of some of them on our work.

- "while my impression of the ... code supplied to student is positive, there seems to be an awful lot of thinking-work to be done at such an early stage..."
  The lab was revised to contain a simple startup problem before more complex ideas were introduced.

- "...The last paragraph, on loops, never says what a loop is..."

- "... You mention a few key concepts without defining them..."

- "... I suggest that the explanation be linked with an introduction to what happens in the lab..."

- "... there are about five listed goals and this may be too many...there should be two types of goals, which I will call primary and secondary.

The primary goals are to learn for loops and do a simple animation. Clearly, loops are one of the most important concepts for an introductory computing course. The secondary goals are to learn several new coding tools. These include delay, SysBeep, and the SoundChannel class."
This lead to a complete revision of the way goals are listed in all labs.

- "The section on materials is okay, although at first it seems as if there are far too many files associated with this lab. However, the first real problems come in the next section, at the bottom of page 2. 'The essence of abstraction is to make manifest the crucial concepts at the highest level and to hide the details at the lower levels' is a problematic sentence; the sentence is the essence of abstraction, perhaps."

- "...I have one suggestion for the 'Laboratory Activities section. I found it much more useful to run the solution under the pause mode, so that I could make predictions based on my version of the algorithm and then test them (i.e., the scientific method)...."

- "... Currently the main goals of this lab are to introduce the type double and the concept of returning values to functions. I suggest that these be changed to more advanced concepts. ... As a replacement of these goals, I suggest you introduce reference parameters here. They are used widely in the [next] lab, but that lab introduces too many new concepts to allow beginning reference parameters there as well..."
  This was an extremely useful comment and lead to a complete revision of this lab and its goals.

This type of evaluation is quite time consuming. It requires that the evaluator not only has the time to read the lab materials and do the lab assignment, but that there is a time for reflection, asking questions, thinking about alternatives, questioning the authority. As can be seen from the selected comments, our evaluator had a number of very valuable suggestions - not only about the individual labs, but even about changing what topics were introduced in a particular lab. The corrections ranged from simple grammatical mistakes to suggestions for a complete reorganization of the code supplied to students. Our evaluator played the role of novice programmer very well - he would often comment - 'I would have liked to learn this topic your way' or, alternately, 'This is too much to do all at once'.

It is clear that such a thorough evaluation could not have been done by students currently enrolled in the class and filling out a post-mortem questionnaire. Teaching assistants burdened with the time pressure of regular teaching and grading together with their own coursework also could not provide such a detailed review. A novice that does not know what new concepts may appear in the next lab does not have the necessary perspective to comment on the overall flow of the whole course. The evaluation reports we received made it clear that a well-structured usage simulation by an advanced neophyte is an effective evaluation method for introductory laboratory materials.

We are now in the process of rewriting and editing these labs and expect that the students taking the course this year will benefit greatly from these improvements. We have revised several of the labs based on these evaluations. We also decided to separate a substantial amount of tutorial material from the laboratory instructions. We felt that the students in the course this Fall were more comfortable with the lab materials and focused better on the concepts presented in each lab. We expect to do the final careful revisions this summer.

4. Conclusion

A systematic evaluation of a collection of course materials is a difficult task. The authors of the labs are not well suited to do such evaluations alone. They are too close to the product, implicitly understanding points that may not be clear to students. They also may not think of alternate ways of presenting the goals, the tutorial, or the activities. Students taking the class also cannot see the big picture - being immersed each week in the most current activity and trying to absorb a wealth of new concepts. Peer evaluation could work to some extent, but it would require that the peer evaluators would also actually do all the student coding. The main concern here would be the amount of time required to do such a thorough evaluation.

We feel that usage simulation method for evaluation of curriculum materials by a student only slightly ahead of the targeted group provides a feedback not available from the traditional sources. It is important that such evaluator does not have difficulties understanding the concepts, is able to ask questions, is not afraid to offer criticism, and can reflect on the overall structure of each lab individually, as well as the structure of the whole course. Such work should always be rewarded - in salary or a course credit. It would be valuable to have several students conduct these evaluations, hopefully providing a perspective related to different backgrounds and learning styles. We highly recommend this form of evaluation to others involved in the curriculum material development.

References

1. Adams, D. R., Powers, M., and Owles, V. A., Computer Information Systems Development: Design and Implementation, South-Western Publishing Co., Cincinnati, OH, 1985.
2. Boroni, C. M., Eneboe, T. J., Goosey, F. W., Ross, J. A., Ross, R. J., "Dancing with DynaLab: Endearing the Science of Computing to Students", Proceedings of the 27th SIGCSE Technical Symposium on Computer Science Education, Philadelphia, PA, 1996, 135-139.
3. Calloni, B. A., and Bagert, D. J., "Iconic Programming Proves Effective for Teaching the First Year Programming Sequence", Proceedings of the 28th SIGCSE Technical Symposium on Computer Science Education, San Jose, CA, 1997, 262-266.
4. Fell, H. J., Proulx, V. K., and Casey, J., "Writing Across the Computer Science Curriculum", Proceedings of the 27th SIGCSE Technical Symposium on Computer Science Education, Philadelphia, PA, 1996, 204-209.

5. Parker, B. C. and McGregor, J. D., "A Goal Oriented Approach to Laboratory Development and Implementation", Proceedings of the 26th SIGCSE Technical Symposium on Computer Science Education, Nashville, TN, 1995, 92-96.
6. Preece, J., Rogers, Y., Sharp, H., Benyon, D., Holland, S., and Carey, T., Human-Computer Interaction, Addison-Wesley, 1994.

Partial support for this work has been provided by the National Science Foundation Leadership in Laboratory Development , award #DUE- 9650552

Appendix A: Lab Evaluation Checklist

Part I:

- Read the lab write-up.
- Record the amount of time you spent reading it.
- Mark all places where the explanation is not clear, suggest changes, write down the questions you have.
- Suggest corrections to grammar as needed, including spelling errors.

Part II:

- Do the lab.
- Record the length of time you needed to do the lab.
- You are allowed to ask questions of one of us while you do the lab (or before that). Record the questions you asked.
- Record the problems you encounter, (other than expected debugging errors)
- Complete the lab report if there is one

Part III:

Write a short summary of your overall impression of the lab focusing on the following questions:
- What do you think the lab was supposed to teach
- Did it do it 'the right way'
- Were the goals clear
- What was confusing
- What was missing
- What can be improved
- What else may students need to know before doing the lab
- Were you told too much?
- Was it too wordy? too brief?
- What do you think about the balance between the code supplied to the student and the work to be done - is it right, too much of our code, too much stuff for students to do
- Do we over-explain or overspecify the assignment - how can we improve that aspect
- Was the lab interesting - why or why not

You may also suggest changes and even to write down new prose as an alternative. In addition, you may suggest new labs based on gaps you discover, or based on other interesting problems that may occur to you

Appendix B: Lab Evaluation Report

Lab Title:
Evaluator:
Report Date:

Part I:

- Record the amount of time you spent reading the lab:
- Mark corrections and suggestions on the copy of the lab write-up, or create a new MS Word document

Part II:

- Record the length of time you needed to do the lab:
- Record the questions you asked while doing the lab:

- Record the problems you encounter, (other than expected debugging errors):

Part III:

Write a short summary of your overall impression
(following the suggested outline)