A NOVEL, SUSTAINABLE MODEL OF ASSESSMENT OF PROGRAM OUTCOMES FOR A CSE PROGRAM

Neelam Soundarajan
Computer Science and Engineering
Ohio State University

Abstract

Assessment of program outcomes and the use of the results of those assessments to effect program improvements as well as documentation of the assessment processes and the resulting program improvements are key requirements of Engineering Accreditation Criteria. Over the last several years, programs have struggled to come up with direct assessment mechanisms that are not resource-intensive (in terms of faculty effort and time) and provide useful results that lead to specific, documentable program improvements. In this paper, we report on a novel approach that is both powerful in terms of its ability to identify specific program improvements and, at the same time, requires minimal resources to administer and hence is sustainable on a long term basis.

Introduction

Prados, Peterson and Lattuca [1] trace the history and evolution of engineering education and accreditation criteria through the twentieth century, culminating in the development of the Engineering Criteria 2000 (henceforth, EC [2]). The main motivation behind the criteria was to significantly reduce the specification of curricular content. Instead, each program was required to identify a set of program objectives, tailored to the program, and a corresponding set of outcomes, including the eleven outcomes (3.a) through (3.k) specified as part of Criterion 3 of EC. A key requirement of EC was a continuous improvement process based on assessing the degree to which graduates of the program achieved the program's outcomes and using the results of the assessments to effect program improvements. EC also required clear documentation of the assessment processes, the assessment results, and the improvements based on an evaluation of these results. While the curricular flexibility was widely welcomed, many programs have struggled to meet the requirements related to assessments of the extent to which the outcomes were attained, evaluation of the assessment results, use of the evaluation to identify possible improvements in the program, and documentation of all of this.

While a number of different assessment approaches have been proposed, some of which we review in the next section, most of them are very resource-intensive, especially in terms of faculty time and effort, both to administer and to document on a long-term basis. Moreover, there seems little evidence that programs have been able to effect improvements that can be directly or primarily attributed to evaluation of the results of these assessments. In this paper we present an approach that requires only modest resources to administer, is easy to sustain over the long term, is easy to document, and allows us to identify specific problems in the program and possible improvements to address them.

A key issue related to assessments has been the question of direct versus indirect assessments. A direct assessment is one that is based on evaluation of actual student work by someone who is qualified to perform the evaluation such as a faculty member or an internship supervisor. This may be contrasted with indirect assessments such as opinion surveys completed by students. It is important to note, contrary to what is occasionally claimed, that direct/indirect-ness have no relation to whether the assessment is quantitative/ qualitative. A direct assessment, for example, an assessment by an intenship supervisor of an intern, may well be qualitative; conversely, an indirect assessment, for example an exit-survey by a graduating student, may evaluate, on a quantitative scale, the extent to which the student attained the program's outcomes. The key point of direct assessments is that they are based on assessing actual student work by people qualified to do so, not whether the assessments are qualitative or quantitative.

Direct assessment of outcomes is not an explicitly specified requirement of EC and, indeed, in the early years following the establishment of EC, most programs relied heavily or entirely on indirect assessments. However, it was soon recognized that direct assessments provide the most reliable assessment of

the extent to which various outcomes are achieved since they are based on evaluation of actual student performance in relevant activities. Moreover, direct assessments are more likely to identify specific problems in, say, the curriculum of the program, and hence provide precise ideas for program improvements. Thus, many program evaluators expect to see at least some use of such assessments. Programs have, however, struggled to come up with direct assessment mechanisms that provide useful results leading to specific, documentable program improvements and, at the same time, are not resource-intensive. Of special concern have been faculty time and other resources involved in administering the assessments, collecting and collating the data, etc. POCAT (for program outcomes achievement test), the direct assessment approach that we present is, as we will see, both powerful in terms of its ability to identify specific program improvements and, at the same time, requires only modest resources to administer and sustain on a long term basis. Moreover, documenting the assessments, its results, and the program improvements based on the results also require only modest amount of effort and resources. It is also worth noting that while POCAT was developed for use in the author's Computer Science and Engineering program, the approach can be equally used in other engineering programs.

The eleven outcomes, 3.a through 3.k, included in Criterion 3 of EC may be classified into two groups, the technical outcomes group and the professional outcomes group. The former group contains such outcomes as, 3.a: an ability to apply knowledge of mathematics, science, and engineering; and 3.k: an ability to use the techniques, skills, and modern engineering tools necessary for engineering practice. The latter group contains such outcomes as, 3.g: an ability to communicate effectively. POCAT is intended only for the outcomes in the technical group. Assessing the outcomes in the professional group will require other methods. This is to be expected given the very different nature of the two groups of outcomes.

One of the key considerations behind the development of POCAT was to take account of recent developments in how individuals learn and how assessment results can be used to improve student learning. As Pellegrino [3] argues, "... our assessment system is seriously flawed and broken. Given the amount that we currently spend on assessment, we get very little in the way of positive return on investment. Many believe the return is actually negative with respect to valued educational outcomes. ... our approach to asssessment [should be] changed subtantially so that it can support processes of teaching and learning focused on deep learning and understanding". The How People Learn (HPL) framework [4] provides key insights into how students learn and the main impediments to improving their learning. A main point of HPL is that, depending on the field, students tend to harbor certain common misconceptions, and these tend to be among the most important impediments to learning. As we will see, POCAT has been very effective in helping identify misconceptions and other essential difficulties related to specific technical topics and ideas that students in our program share, allowing us to develop specific ways to address them, thereby improving the program.

A second consideration in the development of POCAT was that it is intended to be an assessment of the program, not of individual students. One of the common observations of educators at all levels is that students "cram" for final examinations in courses and once the examinations are complete, essentially forget most of the material. Moreover, depending on the particular course, there may be a considerable period of time between when a student takes the course and when he/she graduates from the program. This means, given that program outcomes are intended to be an indication of the knowledge and skills that graduates of the program are expected to possess, using student performance on specific questions in final examinations in particular courses to assess the extent to which these outcomes are achieved, as many programs report doing, seems rather questionable. As we will see, POCAT is designed to avoid these problems.

The main requirements that we imposed on POCAT may be summarized as follows:

1. It must provide a direct assessment of the technical outcomes in EC Criterion 3;
2. It must be an assessment of the program rather than of individual students;
3. Evaluation of the assessment results must help identify the kinds of problems that learning frameworks suggest are the main impediments to students' deep learning; and help us arrive at specific changes in our courses to address the problems, thereby improving the program;
4. The assessments, its results, the evaluation of the results, and the program improvements based on the evaluation should be easy to doc-

ument both in order to meet the ABET requirements as well as to help the program faculty keep track of the evolution of the program and the rationale behind it;
5. And, perhaps most importantly, that the process be sustainable over the long term with modest resources.

The rest of the paper is organized as follows. In the next section, we review related work. In particular, we consider several of the approaches that other programs have developed to meet the EC requirements with respect to assessment and improvement and some of the problems with these approaches. Next, we turn to POCAT. We detail its design, summarize the results we have obtained thus far, and consider the resources needed to implement it in the long term. Finally, we summarize how POCAT meets the above requirements and consider a possible improvement in POCAT.

Related Work

From the earliest days of EC , one of the key questions that programs have struggled with was finding suitable ways to meet EC's requirements regarding suitable assessment processes and documented improvements based on the results of the assessment of their outcomes. Evidence for this may be seen in the many papers in ASEE and FIE Annual Conferences, indeed in the number of sessions at these conferences devoted to discussions of ways to meet the outcomes assessment requirements of EC; the annual Best Assessment Processes (BAP) Symposium devoted to the topic; numerous papers in several volumes of the Journal of Engineering Education, IEEE Trans. on Education as well as special issues exploring the topic; etc. In a recent paper, Shaeiwitz and Briedis, both with considerable experience as program evaluators for engineering accreditation evaluations and as team chairs for these evaluations, note, "it appears that many programs are struggling to identify valid measures for their program outcomes . . . This is substantiated by evidence of the relatively large number of citations [following evaluations] for shortcomings relative to some aspect of this criterion" [5]. The statistics reported by ABET on the specific criteria that programs have difficulties with confirms this.

Shaeiwitz and Briedis go on to argue that a major reason underlying these problems is that many programs have thus far focused on indirect assessments of program outcomes using such techniques as exit surveys of graduating students and faculty opinions; and that direct assessments are necessary to provide objective measures of achievement of the program's outcomes and must be an essential part of every program's suite of assessments. Others have also made a strong case for relying on direct, rather than indirect, assessments.

Unfortunately, in the experience of many engineering programs, many direct assessment tools that programs have attempted to use have been, on the one hand, very resource intensive in terms of the amount of faculty effort required to use them; and, on the other hand, proved to be of limited value in assessing the extent to which the program outcomes are achieved by the students and in identifying possible improvements. For example, one commonly suggested approach is to use portfolios of student work [6, 7]. However, especially for large engineering programs that graduate more than a handful of students each year, the sheer volume of data collected via portfolios can be enormous. While e-portfolios might simplify the task of storing large volumes of data, since electronic storage space continues to become cheaper, and software can help with the organization of the materials, the task of evaluating all the collected information and arriving at possible improvements in the program can be overwhelming. Indeed, much of the literature on the topic does not even bother to address this question, stopping instead at the stage of how to collect and organize the materials included in these portfolios. But the ultimate purpose of the EC requirement is not collection of data, nor even assessment, but rather using the results of the assessment to arrive at program improvement [8, 9].

Another approach to direct assessment has been the idea of using targeted questions in examinations in particular courses in the curriculum, see, for example, [10, 11]. The common idea in this approach is that particular courses in the (core) curriculum are identified, and particular topics in those courses are associated with specific program outcomes. The faculty teaching the course are then required to ensure that the examinations (or quizzes etc.) in each section of such a course that they teach includes questions specifically targeted to those topics. The faculty are then required to provide a summary of the student performance in those questions; this summary is considered as providing the assessment data with respect to the particular outcome. While the data collected and stored using this approach is more manageable than in the case of portfolios, it

does require conscientious participation of the involved faculty. More importantly, the question of arriving at program improvements based on evaluation of the assessment data also remains.

Mak and Freza [12] present a method following this approach. Specific assignments in specific courses are tagged as the ones that measure particular outcomes; students are then required to achieve minimum specified performance in those specific assignments, else they cannot graduate. This is high-stakes testing and seems unfair to students who may have performed well in other assignments in those and in other courses. Danielson and Rogers [13], Howard and Musto [14], and Harvey et al. [15] present somewhat similar approaches. In each of these, a set of exam problems or other graded work in individual courses is related to specific program outcomes and the performance of students in these problems is assessed and used as an assessment of the particular outcome. For the student, these approaches are not high-stakes in the same manner as that of [12]. Nevertheless, as Helps, Anthony and Lunt [16] point out, such approaches are very expensive in terms of faculty time and effort. Further, as noted in Section 1, the performance of a student in course examinations is not necessarily a good indicator of the knowledge and skills that graduates of the program are likely to possess. Each of these approaches, like many of the others, is really an assessment of individual students at certain points in the program rather than an assessment of the program. Moreover, the emphasis in each case seems to be on assessment for the sake of assessment rather than for the sake of program improvement.

Before concluding this section, it would be useful to mention the idea of concept inventories [17, 18, 19] which, while unrelated to outcomes assessment as in EC, has some similarities with the POCAT approach. The original concept inventory was created by Haloun and Hestenes. The inventory was for Newtonian Mechanics. It was intended to assess students conceptual understanding of key principles of the subject such as force and momentum. The inventory was a multiple choice test with each question containing distractors that were designed based on common misunderstandings that students have with respect to that concept. Although a multiple choice test would seem to be incapable of assessing deep understanding of the subject, experience with the inventory showed that a well-designed one can be remarkably effective. Since the original work, a number of inventories have been developed to assess student understanding of a range of subjects from strength of materials to electromagnetics to fluid mechanics etc. As we will see, while there are some similarities between concept inventories and the POCAT approach, there are also crucial differences.

POCAT Model/Approach

Background: Students in the CSE program at Ohio State typically take, during their sophomore year, fairly standard courses on programming and software engineering, introduction to systems, and discrete mathematics. In the junior year and early senior year, they take a team-project course and complete a number of courses on core computing topics such as automata theory, concepts of programming languages (including language implementations), computer architecture, databases, operating systems and algorithms. Each of these courses builds on the earlier courses in the program; for example, the architecture course builds on the introduction to systems courses and heavily uses ideas from the discrete math course. In some cases, these courses build each other as well; for example, the programming language course borrows ideas such as formal grammars from the automata theory course. In the late junior year and the senior year, students take a number of elective courses on a range of topics such as AI, computer graphics, networking, information security etc. These courses also build on concepts and technical details that students learn in the beginning courses as well as in the more advanced core courses. The students also complete, typically in the final quarter —Ohio State is on the quarter, not semester, system— of the senior year, their capstone design course.

Let us now turn to the program outcomes. As noted in Section 1, POCAT is designed to assess student achievement of Criterion 3 outcomes that fall in the technical group. These outcomes are:

3.a an ability to apply knowledge of computing, mathematics including discrete mathematics as well as probability and statistics, science, and engineering;
3.b an ability to design and conduct experiments, as well as to analyze and interpret data;
3.c an ability to design, implement, and evaluate a software or a software/hardware system, com ponent, or process to meet desired needs within realistic constraints such as memory, runtime efficiency, as well as appropriate constraints related to economic, environmen-

tal, social, political, ethical, health and safety, manufacturability, and sustainability considerations;

3.e an ability to identify, formulate, and solve engineering problems;

3.k an ability to use the techniques, skills, and modern engineering tools necessary for practice as a CSE professional.

Each of these outcomes relates to a number of the courses listed above. Thus, for example, (3.a) relates to almost every one of the courses; this is not surprising given the breadth of (3.a). (3.c) is related most directly to the junior-level project course whose main component requires the students to design and implement a number of important system-level software pieces and integrate them, and the capstone design course which engages students in an intense quarter-long design and implementation of a system that meets specified requirements and constraints. It is also related to such courses as the operating systems course and the course on concepts of programming languages since these courses address, in various contexts, questions related to memory efficiency and runtime questions; and often include suitable programming projects although of a somewhat smaller scope than in the capstone design course.

(3.e), like (3.a), is related to most of the courses including even the discrete mathematics course since part of solving certain CSE problems consists of precise characterization, using formal logic notations introduced in that course, of various aspects of the problems' requirements as well as the solutions. Other courses are more directly related to solving CSE problems of various kinds; for example, the programming languages course familiarizes students with functional programming languages which are better suited for developing programs to deal with certain kinds of computing tasks than are standard imperative languages.

One reasonable interpretation, the one we have adopted in our program, of (3.b) is that the task of designing test suites to test the functioning of complex software systems and analyzing and interpreting the results of the test runs to arrive at proper conclusions about the system under test are analogous to designing and conducting experiments and analyzing/ interpreting the resulting data. Thus, the junior project course as well as the sophomore software-engineering sequence both contribute to (3.b). In addition, several of the other courses which require the students to suitably test the software they design and implement also contribute to this outcome.

(3.k) too relates to a number of courses. For example, even the the course on automata theory which is occasionally dismissed as "just theory" contributes to it since notions such as regular expressions that are central to that course are among the most powerful tools in a CSE professionals' arsenal. In other courses such as on networking and information security, students study other tools such as communication protocols and ones for ensuring security of online transmissions.

Some of the details below are somewhat specific to Computer Science and Engineering (CSE). Indeed, several of the outcomes listed above, in particular (3.a), (3.c) and (3.k), are slightly specialized versions of the corresponding EC Criterion 3 outcomes, specialized to be more directly related to CSE. Nevertheless, the underlying approach is applicable to all engineering programs.

POCAT Details

The Program OutComes Achievement Test (POCAT) is designed to test student achievement of the outcomes (3.a), (3.b), (3.c), (3.e), and (3.k). POCAT is a test that students are required to take near the time of their graduation from the CSE program. The questions in the test are based on topics from nine required high-level courses in the program including the ones mentioned in Section 3.1 as well as a number of popular elective courses. As mentioned earlier, these courses range over key topics such as automata theory, databases, programming languages, computer architecture, algorithm analysis, AI, networking, information security etc.

All the questions on the test are multiple-choice questions with, typically, two or three questions in each topic area. But they are not the typical questions one might find in, say, the final exams of these courses. Instead, they are more conceptual and are designed to test how well students understand key concepts from across the curriculum. The distractors in each question, i.e., the wrong answers, are carefully chosen based on the misconceptions that students typicallly harbor concerning the particular topic. Each question is often the result of sometimes quite extended discussions among faculty involved with the courses in question. The discussions usually focus on identifying the common student misconceptions about a given concept and on the best ways

to capture these misconceptions in suitable distractors. The questions on the test are also chosen in such a way that there is at least one —and often more than one— question directly related to each of the outcomes in the technical skills group. This is straightforward because, as explained in Section 3.1, the nature of most of these outcomes is such that many courses are related to each one; thus most questions based on these courses typically correspond to one or more of these outcomes. A sample of the POCAT test is available on our web site [20].

All students in the program are required to take the test one to two months prior to graduation from the program. But the students' performance on the test do not affect the grades of individual students in any courses, nor indeed are any records retained on how individual students performed on the test. When a group of students takes the POCAT, each student in the group receives a unique code that appears on that student's test but only the individual student knows his or her code. Once the tests have been graded, summary results, organized by this code, are posted on electronic bulletin boards so an interested student can see how well he or she did and how his or her performance compared with that of others who took the test; no one else, whether faculty, staff, or other students, have any way of knowing how a given student performed on the test. A sample set of results is available on our web site [21]. Making the test results anonymous in this manner was a deliberate decision. We did not want students to spend a lot of time preparing ("cramming") for the test since the goal was to assess the extent to which they have acquired and internalized the knowledge and skills associated with the various outcomes since this is what they will take away with them as graduates of the program. The test results are a true measure of the program's outcomes.

Initially, there was a concern that if individual students' performance on the test did not affect them in any tangible way, they would not take the test seriously. Our experience with the many administrations of the test have completely eliminated this concern. Students, released from anxiety about a high-stakes test, seem to enjoy taking the test and try to do their best. Following the test --which is typically held on a Tuesday evening in the middle of each quarter-- it is not uncommon to see students who had just completed the test having long and heated discussions about some of the questions on the test; it is also possible that the pizza and soft drinks that the students are served immediately after the test serve as good lubricants! The contrast from what one typically sees following a midterm or final exam in a course could not be more stark. It is almost as if the test has the effect of transforming many students whose main interest in any course is to do just well enough in the exams so that they receive a satisfactory grade and are able to graduate, into budding CSE professionals who bring all of their knowledge and skills to tackle challenging problems.

There is one other feature of the POCAT questions that is worth remarking on. Each question has, as one of the choices (typically the last one), an answer along the lines of "I don't know". The instructions for the test suggest that the student should pick that answer if he or she has no idea what the correct answer is. Since their performance on the test will have no impact on their record, students who do not know the answer to the question and know that they do not know pick this answer. This means we do not have to worry about the student trying to make guesses and confounding our attempt to pin down misconceptions that he or she may have. Interestingly, students choosing this answer deliberately also represents their evolution from being students to becoming CSE professionals. As students, their main goal tends to be to get the best possible scores in the tests; hence they make even wild guesses if there is a possibility that doing so would improve their scores. As professionals, their goal should be to solve problems; and the first step in successfully doing so is recognizing, where appropriate, that they do not know the answer to some question, so they can seek suitable assistance.

The faculty members responsible for each question provide an estimate of the percentage of students who ought to be able to answer the question correctly as well as the particular outcomes that the question is related to. This information is also included in the summary results. The final aspect of POCAT is the evaluation of the results and arriving at ideas for program improvement. The initial discussion of the results takes place in the program's Undergraduate Studies Committee. The committee consists of several faculty including some who regularly teach the high-level courses included in POCAT; student representatives; and the staff adviser (who also takes care of administering the test). The committee considers such issues as: a. Are there any questions for which the percentage of students who got the correct answer differs substantially from the figure that the faculty involved with the corresponding course(s) expected? b. Are there any questions for which par-

ticular incorrect answers, i.e., distractors that represent particular misconceptions, especially more popular than other incorrect answers? c. Are there any longer term trends with respect to questions related to particular concepts? Etc. The student members on the committee often provide insight into particular misconceptions that students might have by noting, for example, that a course taught by a particular instructor takes a particular approach to an idea or a topic and that that might lead to certain specific misconceptions with respect, perhaps, to a related concept. The staff adviser might occasionally note that, in the pizza session following the test, a particular question seemed to provoke the most intense debate. And faculty who have taught the particular courses or related courses bring important insights into analyzing and understanding the results.

Most commonly, the results of the POCAT do not offer many surprises. But, occasionally, there might be a question in which a substantially smaller percentage of students than expected get the correct answer. In other cases, distractors that the faculty might consider obviously incorrect might be chosen by a significant number of students even if the percentage of students who got the correct answer is in line with expectations. In yet other cases, a substantially larger percentage of students than expected might get the correct answer. In the first case, an appropriate course of action, in the form of suitable changes in the course in question or possibly in prerequisite courses, may need to be identified. But such changes are not determined in this meeting of the Undergraduate Committee. Instead, the faculty involved with the courses in question (not all, sometimes not even any, of whom might be on the committee) are informed about the anomaly. Those faculty might then decide to further investigate the problem by introducing new activities into their courses; or they might decide that the problem might be with the precise wording of the POCAT question and offer a revised version of the question for use in future POCATs; or this result might provide added confirmation for what they had already concluded on the basis of observations in their course and start work on designing appropriate changes in the course.

The course of action in the second case in which an unexpected number of students chose a particular distractor might be similar. The question in this case would be, why did a large number of students find the particular distractor appealing? In this case, though, the issue often tends to be poor wording of the question; but there have also been cases where such a result has helped identify certain misconceptions prevalent among students that the faculty had not thought about. The third case is also rather interesting. It may suggest one of two possibilities. Either the distractors had not been sufficiently well designed so that students were able to arrive at the correct answer by eliminating all or most of the distractors. Or, in fact, students do have a better understanding of the idea or concept in question than faculty had given them credit for; so the faculty may decide to revise the course to increase the depth of the discussion with respect to that concept. Later, we will present some specific examples.

We conclude this section by comparing the POCAT-approach with concept inventories [18, 19]. There are some obvious similarities, for example, in the types of questions used; but there are also some important differences. For one, concept inventories are intended to test the conceptual misperceptions of specific, individual students. Thus, it would not make sense to make these tests anonymous as is the POCAT. Second, questions in concept inventories do not include "I don't know" as a choice because students would always be expected to make a best guess. These are typically students entering their first courses in college, not students about to graduate and become professionals. Third, security is an important consideration with concept inventories because students' performance on the test can have a significant impact on which courses they are placed in, how fast they can progress through the curriculum, etc. By contrast, there is no motivation for individual students to try to "cheat" on POCAT since their performance does not become part of their record, indeed is not even known to anyone else. Perhaps the most important difference is that while concept inventories are intended to be common to all programs in a particular discipline, POCAT is very much tailored to our program. The questions are designed by the faculty who teach the particular courses and are intended to help identify problems in our particular courses as well as ideas for improvement in our particular program. Thus a specific POCAT test we use would not be appropriate for use in another program. However, the approach certainly is usable by any CSE program, indeed by any engineering program. What is required is for the program faculty to identify key high-level courses and design suitable multiple-choice questions that probe for conceptual (mis)understandings that may be common among students. Next, use the results to fine-tune the questions until the problems are clear. And,

finally, develop suitable revisions in the course(s) to address the problems to effect improvements in the program.

Results

In this section, we consider some recent specific improvements in our program based on the POCAT assessments. We start with a somewhat detailed description of our first improvement and follow that with much briefer descriptions of the rest. One of the courses included in the POCAT is CSE 655, a required late-junior/early-senior-level course on concepts of programming languages. One of the most common problems that programs, especially large ones, exhibit has to do with uninitialized variables; i.e., using a program variable without having assigned it a suitable value. There are various possible ways to address this problem. First, we could design our programming language in such a way that whenever a variable is defined, it is automatically assigned some appropriate default value. A second would be to change the syntax of the language so that a programmer cannot introduce a new variable without explicitly specifying an initial value for it. A third would be to have the compiler analyze any program it compiles to check that each variable has been initialized before it is used. A fourth would be to have the compiler insert, into the compiled code, additional checks to make sure that each variable that is used has a value that was actually assigned to it. The fifth and final approach would be to do nothing and expect the programmer not to make the mistake of using a variable without first initializing it; in this case though, if the program does have an uninitialized variable, the program will probably crash when the compiled code is actually executed because the system will use whatever random bit pattern happens to be in the memory location assigned to that variable.

Each of these approaches has advantages and disadvantages. For example, the first approach, assigning a default value, does eliminate the problem but it may be masking a real bug in the program; i.e., the programmer truly forgot to assign a specific value to a particular variable and then used that variable, assuming that she had previously assigned the correct specific value to the variable. In this case, the program will run (using the provided default value) but the results will probably not match what the programmer expected and debugging this can be a problem. By contrast, if the program had crashed (as in the last approach), the programmer might have reexecuted the program after inserting some "breakpoints" (at which points the program will stop before waiting to be told to continue) and quickly localize the problem. Of course, if the program doesn't actually crash because the random bit pattern at the memory location in question happens to be a legitimate value, it would be as difficult to find the bug as in the first approach.

What about the third approach where the compiler analyzes the program to check that each variable has been initialized before it is used (and issues a warning if it finds variables that are used before being initialized)? While this would be ideal, it doesn't always work. The problem is that because of complex conditional and looping structures in the program, the compiler cannot tell exactly which parts of the program will be executed before which other parts. It can do an approximate analysis; and arrive at a conservative evaluation that would flag some uses of certain variables as questionable because it is not able to conclusively establish that, in all cases, during program execution, that the variable in question will be initialized before being used. Java uses this approach. C++ uses the fifth approach (leave it to the programmer); Resolve-C++, a local dialect of C++ that is used in our beginning CSE sequence, uses the first approach.

This topic is discussed in some depth in CSE 655. Students who have internship or other work experience often bring up other languages (such as Perl) that they may have encountered in their work places and talk about how they seem to handle the problem. At the same time, at least for some students the essential conceptual nature of the problem and its possible solutions tend to remain unclear. Here is a POCAT question designed by faculty who teach the course to identify problems related to this concept:

> One common problem in programs is that of uninitialized variables, i.e., using a variable without having initialized it. This is commonly a run-time error but Java flags this error at compile time. How does it do this?

1. Java uses a special technology that converts run-time errors into compile-time errors;
2. Java uses a "conservative" approach, sometimes flagging situations which are not actually erroneous;
3. Java does automatic initialization of all variables so the problem of uninitialized variables cannot arise in Java programs;

4. Java is an interpreted language, so this question is meaningless;
5. I have no idea.

The correct answer is, of course, (2). But many students, perhaps because of the "buzz" around Java, seem to pick (1). The third choice is more involved. It turns out that Java, in fact, does automatic initialization but not of all variables; that is what makes this a wrong choice. That means, a student who actually understands the concept may still pick this wrong answer. Thus there is a key difference between this student and one who picks answer (1). Answer (4) is another interesting distractor. Languages may be implemented using either compilers or interpreters (with apologies to readers with background in Computer Science & Eng.!). Interpreters don't actually translate the given program; they instead execute it more or less as given. This means that if they encounter this situation, they can easily identify it during execution of the program and print a suitable warning message making the job of the programmer very easy. Although Java does use interpretation, that part of it is not relevant to this discussion; hence the correct answer is indeed (2). However, again a student who understands the concepts well may choose (4) as her answer because she just does not know (or did not remember) some details of how Java works. The last answer, "I have no idea", as discussed earlier, is important. Students who do not know the answer to the question, and know that they do not know, will pick this answer.

When the question was tried a couple of years ago, the faculty in question expected 70% or so of the students to get the correct answer. In fact, the number of students who picked the right answer was substantially less. While some of the students seem to have chosen an answer (such as (3)) that would indicate not having knowledge of some Java details, many others chose answers (such as (1)) that indicated failure to have a sufficiently good grasp of this important concept. Indeed, someone with a good understanding of the concept should, even if she had not heard of Java before, be able to choose (2) as the most likely answer. Based on this, the faculty revised the discussion in CSE 655 to include a more detailed discussion of the topic. The performance of students in recent offerings of POCAT in this (and similar) questions has been substantially better. In terms of program outcomes, this question is related to (3.c), (3.e), and (3.k).

Our second example is related to CSE 680, a junior/senior-level course on algorithms and analysis of algorithms. One fairly standard topic in this area is solving what are known as recurrence relations. These relations can be used to express the running time of certain algorithms; in effect the running time of the algorithm for input of a certain size is related to the running time for input of a smaller size; which, in turn is related to the running time for input of still smaller size; etc. But getting a good feel for the actual running time of such an algorithm for large inputs requires us to "solve" the relation to obtain the asymptotic behavior of the algorithm. In general, this can be a fairly difficult task but if certain conditions are satisfied, a result known as the Master theorem can simplify the task considerably. This can be important in certain situations such as when dealing with algorithms designed to search through very large volumes of data since the difference in running time between different algorithms for the task can be very substantial. Hence it is important to be able to solve the corresponding recurrence relations so that one can make an informed choice among the algorithms in question.

Hence the faculty involved with CSE 680 designed a POCAT question intended to see if students are able to solve (reasonably simple) recurrence relations. The performance of the students who took the test was unexpectedly poor. In the evaluation discussion analyzing the test results, one explanation that was offered was that students were, in fact, capable of using the Master theorem to solve the relation — the relation in the POCAT question being one that satisfied the conditions that allow the Master theorem to be applied— but that, because of the complex nature of those conditions, students could not be expected to remember them when taking the POCAT. Indeed, this seemed to be confirmed when the CSE 680 instructor asked a similar question as part of his final examination for the course. A large majority of the students in the course answered the final exam question correctly.

The faculty could, at this point, have accepted the explanation above as accurate but they decided to test it further. In the next offering of POCAT, they revised the question to include a statement of the Master theorem. Given this revision, these faculty felt that students should indeed be able to check that its conditions were satisfied in the given scenario and solve the specified recurrence relation. To their

surprise, student performance on the question on this test was as poor as it had been in the earlier test! This is indeed a puzzle and one that has not yet been resolved. Why did the students in the course final examination do so well when students taking the POCAT did so poorly even when they were provided an explanation of the Master theorem? Further fine-tuning of the question in future offerings of the POCAT will help address the question and tell us whether and what changes in the course are needed.

It may be worth noting that, in practice, one would expect a CSE professional to look up, perhaps online, the details of the theorem rather than necessarily remember them. Thus if the faculty's original explanation that students taking the POCAT simply did not remember the theorem had turned out to be correct, we would have concluded that no change in the course was called for. But it didn't.

Our final example involves two courses, the discrete math course and CSE 321, the final course in our sophomore-level software engineering (SE) sequence. One important thread in the SE sequence is to help students see the importance of precise specifications (rather than informal explanations using a few example test cases) of the behaviors of programs and to have students work with the specifications for a number of simple systems. These specifications, partly in order to be easily machine readable, use a "plain text" set of notations such as "union", "there exists", etc., rather than the traditional mathematical symbols, such as "$\cup$" and "$\exists$", etc. In the discrete math course, which is taught by the mathematics department and is taken by students at about the same time as CSE 321, students learn basic mathematical logic using the traditional mathematical symbols.

Several quarters ago, faculty involved with CSE 321 proposed a question involving a simple specification for inclusion in the POCAT. The faculty expected most students to be able to answer the question correctly; their main purpose in asking the question was to see whether the students had retained the ideas from 321 to the time of their graduation. As it turned out, when the question was typeset for POCAT, words such as "union" were replaced, without knowledge of the 321 faculty, by the corresponding traditional mathematical symbols such as "$\cup$". When the results became available, the student performance was much poorer than the 321-faculty had expected. During the discussion that followed, one of them noticed the change in the notation and conjectured that it was that change that was primarily responsible for the poor performance. So, in the next offering of the POCAT, the question was retypeset using the notation used in 321 rather than the traditional notation and, indeed, students performed much better!

While this seemed to resolve the matter, some of the faculty, including those not directly involved with the SE sequence, were puzzled and concerned. Surely, our students, by the time they graduate from the program, should be able to easily see that, for example, "there exists" and "$\exists$" mean the same thing? And if they are not able to do so, as the results of the two POCATs seemed to indicate, didn't we need to revise some part of the program to ensure that they do so? Given that feedback, the CSE 321 faculty took the following actions. First, they assigned both versions of the question in a final examination of the course (with half the students getting each version of the question) to check whether the students in the course exhibit the same difference in performance when presented the two versions of the question. It turned out that indeed they did. Next, the instructor for the course introduced, in the next offering of the course, a 20-minute explanation/discussion of the essential equivalence of the two notations near the end of the quarter. Again the two versions of the question were asked in the final examination and this time students did equally well with both versions of the question. The conclusion was that, contrary to what faculty expected, students do not intuitively and on their own see the equivalence of, for example, "there exists" and "$\exists$"; these are somewhat involved ideas and need to be presented explicitly and clearly. Hence this explanation is now a permanent part of CSE 321.

In terms of the program outcomes, the 680-based question is related to (3.a), (3.b), (3.c), and (3.e). The 321/discrete math-based question is related to (3.a), (3.e), and (3.k).

Documentation, Sustainability

There are two distinct components to the documentation of POCAT. The first is the documentation of the test results. As noted earlier, this is mostly mechanical and performed by an automated script. The input to the script consists of the following information: for each student code, the answer choices (i.e., one of (a), (b), etc.) made by that student for

each question on the test; the correct choice for each question; the number(s) of the course(s) most directly related to each question; and the numbers of the program outcomes related to the question. Given this information, the script creates the results page such as the one at [21]. This process could be further mechanized by maintaining a database of possible POCAT questions which includes such information as the correct answer for each question, the related program outcomes, etc. Another possibility that would enable essentially complete mechanization would be to have students take the test on-line. We discussed this in a meeting of the Undergraduate Committee. The student representatives on the committee were opposed to this since they felt that students taking the test on-line would be unsure that their anonymity would be preserved. Given the essential importance not just of anonymity but the students' perception of anonymity in POCAT, we have abandoned this idea.

The second component of the documentation is a summary of the evaluation of the assessment results. For each question for which the results were unexpected or otherwise led to discussions in the Undergraduate Committee (and beyond), summaries of the discussion are written up. The summaries are similar to the examples in Section 3.3 except that, since it is intended for our faculty who are, of course, quite familiar with the details of the courses in the program, they tend to be much briefer. These summaries are maintained in a single web page (accessible to faculty in the department) in chronological order. In effect, over time, the page provides a historical view of the changes that were made to the program and the rationale, in terms of the assessment results that triggered them and the summary evaluations of the results, behind the changes. Thus, for example, a new faculty member to the department can read through this page and get an excellent view of the evolution of the program and the reasons behind important changes in the program.

One of the major difficulties that programs attempting to meet EC requirements concerning outcomes assessments, program improvements based on evaluation of the assessment results, and documentation of all of these, has been the amount of resources needed to do so. Indeed, as noted in Section 2, the resources needed in some cases are so onerous that programs have had to abandon them after a year or two. In other programs, one or two individual faculty have, at enormous personal sacrifice, attempted to keep the process going. Perhaps most importantly, almost every one of these cases has been an exercise in collecting assessment data and organizing it and creating tables summarizing the percentage of students who completed some course activity satisfactorily, etc. Almost never is there any evaluation or detailed analysis of the results, let alone ideas for program improvements that might be suggested by such analysis.

By contrast, as the examples in Section 3.3 demonstrate, developing ideas for program improvements based on careful analysis of the assessment results is, in effect, the driving force behind POCAT. As an added and important bonus, the two documentation components of our approach described above perfectly meet the letter and the spirit of the EC requirements. The summary results pages such as the ones at [21] document the assessment results. The evaluation page documents the evaluation of the assessment results and ties program improvements to that evaluation. The resources needed to to sustain the approach over the long term (about four years now) have been relatively modest. There are two aspects of the approach that require significant faculty effort and time. The first has to do with designing suitable questions for POCAT. This task, especially coming up with suitable distractors in each question that correspond to common student misunderstandings and difficulties related to the concept being addressed, can be a challenging task and requires several iterations. But it is precisely the sort of challenge that faculty engaged with the course are glad to take on. This is not the sort of mind-numbing assessment activity performed simply for the sake of meeting EC requirements that faculty rightfully resent. This challenge requires faculty to think deeply about what the central concept in question is, what are the additional concepts and ideas that might be related to it, perhaps peripherally, which might confuse students, how best to capture these potential confusions in a few carefully worded distractors, etc. The other aspect that requires significant faculty effort is the evaluation of the results and determining what changes, if any, are needed in our courses, based on the evaluation. But the task of determining what changes are needed in their courses is something that conscientious faculty engage in routinely and constantly. Our approach, in effect, makes the activity more productive by relating it to results of POCAT and by producing documentation that will be valuable when the same or another faculty member tries to understand why the program evolved in certain ways.

Conclusion

The main requirements that we imposed on POCAT were that it provide a direct assessment of the technical outcomes, (3.a), (3.b), (3.c), (3.e), and (3.k); that it assess the program rather than individual students; that the evaluation of the assessment results help us arrive at changes in our courses that will help address specific problems, thereby improving the program; that the assessments results, the evaluation and the program improvements be easy to document; and that it be sustainable with modest resources. Each of these requirements has been fully satisfied. First, it is clearly a direct assessment since it is based on student performance on the questions in the test. Moreover, just the three samples we discussed in Section 3.3 covered all of the above outcomes, some more than once. Second, POCAT clearly assesses the program not individual students since the performance of an individual student on the test is not known to anyone (other than the particular student).

Third, as the examples in 3.3 showed, evaluation of the test results helps identify specific problems that students have with particular concepts and the resulting discussion among faculty leads to specific changes in the program to help address the problems. Fourth, the web pages documenting the POCAT results and the page summarizing the evaluation of the results and the changes to the program provide the documentation needed to meet EC requirements and to help faculty track the program's evolution. And, fifth, the resources needed for administering the test and producing the summary results are modest. Faculty time and effort are indeed needed for creating the questions on the tests; and for evaluating the test results, identifying problems, and coming up with possible changes in the courses to address the problems. With respect to the former, since questions can be freely reused without concern about the test's security, faculty in each area need only produce a handful of questions every year. With respect to the latter, this is a normal activity that faculty engage in routinely. The only difference is that POCAT provides them test results that can serve as a basis (in addition, for example, to performance of students in final exams in particular courses) for this activity. Thus POCAT does indeed satisfy each of our requirements. Moreover, the approach, although not our actual tests, are very much usable by other CSE programs as well as by all engineering programs.

We conclude with an idea for a change that would make the POCAT approach even more effective. The EC technical outcomes are extremely broad. Indeed, it would probably be quite difficult to come up with a POCAT question that does not relate to several of them. As a result, the fact that for each technical outcome, there are one or more questions on a given POCAT that relate to it is no assurance that the test deals with most of the key technical topics that the program faculty consider as important. A better alternative would be to identify, for each (especially high-level) core course in the curriculum, its key outcomes; and ensure that the POCAT contains questions related to several key outcomes from several of these courses; and that over a period of a year or two, the set of POCATs administered include questions related to each key outcome of each of these courses. That will ensure that faculty are able to identify weaknesses in all important components of the program. We are currently working on implementing this change in our program.

References

1. J.W. Prados, G.D. Peterson, and L.R. Lattuca, "Quality assurance of engineering education through accreditation: The impact of Engineering Criteria 2000" *Journal of Engineering Education*, vol. 94, no. 12, 2005.

2. *Engineering Accreditation Criteria,* ABET website at: http://www.abet.org, 2007.

3. J. Pellegrino, "Rethinking and redesigning curriculum, instruction, and assessment", Paper commissioned by the National Center on Education and the Economy, 2006. available at: www.skillscommission.org.

4. J. Bransford, A. Brown, and R. Cocking. *How people learn: Brain, mind, experience, and school,* National Academy Press, 2000.

5. J. Shaeiwitz and D. Briedis, "Direct assessment measures", in Proc. of ASEE Annual Conference, ASEE, 2007.

6. G. Rogers, "*Got portfolios?*", Technical Report 07-04-CM, ABET Newsletter, 2007.

7. J. Mcgourty, L. Shuman, M. Besterfield-Sacre, C. Atman, R. Miller, B. Olds, G. Rogers, and H. Wolfe, "Preparing for ABET EC 2000: Research-based assessment methods and processes", Int. J. of Eng. Ed., vol. 18, no. 2, 2002.

8. G. Rogers, "Death by assessment: How much data are too much". Technical Report Spring 2002, ABET Communications Link, 2002.

9. C. Chewar, K. Huggins, and J. Blair, "Avoiding the pratfalls of program assessment", SIGCSE Bulletin, vol. 38, no. 4, 2006.

10. D. Ahlgren and J. Palladino, "Developing assessment tools for ABET EC. in Proc. of Frontiers in Education, pages T1A-17. ASEE/IEEE, 2000.

11. R. Felder and R. Brent, "Designing and teaching courses to satisfy ABET EC", *Journal of Eng. Education*, vol. 92, no. 1, 2003.

12. F. Mak and S. Freza, "Using student learning outcomes assessment to assure EC 2000 program effectiveness", In Proc. of the ASEE Annual Conf. ASEE, 2005.

13. S. Danielson and B. Rogers, "A methodology for direct assessment of student attainment of program outcomes", in Proc. of the ASEE Annual Conf. ASEE, 2007.

14. W. Howard and J. Musto, "Assessment workshop: a tool for promoting faculty involvement", In Proc. of the ASEE Annual Conf. ASEE, 2006.

15. H. Harvey, M. Krudysz, and A. Walser, "Direct assessment of engineering programs at the City College of New York", in Proc. of Frontiers in Edu., ASEE/IEEE, 2010.

16. C.R. Helps, D. Anthony, and B. Lunt., "Outcomes oriented ABET accreditation: Mechanisms for review and feedback", in Proc. of the ASEE Annual Conf. ASEE, 2005.

17. A. Savinainen and P. Scott, "Force concept inventory: a tool for monitoring student learning", *Physics Education,* vol. 37, no. 1, 2002.

18. I. Halloun and D. Hestenes, "Common sense concepts about motion". *American Journal of Physics*, vol. 53, no. 10, 1985.

19. D. Hestenes, M. Wells, and G. Swackhamer, "Force concept inventory.", *Physics Teacher*, vol. 307, no. 2, 1992.

20. CSEDept., Sample POCAT., http://www.cse. ohiostate.edu/ugrad/ets.pdf, 2010.

21. CSE Dept., Sample POCAT results, http://www.cse.ohiostate.edu/ugrad/etsr.html 2010.

Biographical Information

Dr. Neelam Soundarajan is an Associate Professor in the Computer Science and Engineering Department at the Ohio State University. His research interests include Software Engineering, engineering pedagogy including assessment and evaluation.