AC 2009-400: DIRECT ASSESSMENT OF PROGRAM OUTCOMES IN A COMPUTER SCIENCE AND ENGINEERING PROGRAM

Neelam Soundarajan, Ohio State University

Neelam Soundarajan is an Associate Professor in the CSE Dept. at the Ohio State University. His technical interests are in Software Engineering, Programming Languages, and in issues related to engineering education, including program assessment and improvement.

Direct Assessment of Program Outcomes in a Computer Science and Engineering Program

Abstract

Although *direct assessment* of program outcomes is not an explicitly specified requirement of *Engineering Accreditation Criteria* (EC), most program evaluators expect to see some use of such assessments. This is not surprising since direct assessments provide the most reliable evaluations of actual achievement, by the students, of the various outcomes. At the same time, programs have struggled to come up with direct assessment mechanisms that are not resource-intensive (in terms of faculty and administration time) and provide useful results that lead to specific, documentable program improvements. In this paper, we report on such a mechanism that is both powerful in terms of its ability to identify specific program improvements and, at the same time, requires only minimal resources to administer and sustain on a long term basis.

1. Introduction

Prados, Peterson and Lattuca, in their article[15] tracing the history and evolution of engineering education and accreditation criteria through the twentieth century, write: "By the late 1980s, ... engineering practice was changing dramatically and irreversibly ... [existing programs] produced graduates with strong technical skills, but these graduates were not nearly so well prepared in other skills needed to develop and manage innovative technology ... engineering accreditation had become an impediment to reform ... criteria were increasingly prescriptive ... institutions that attempted flexible and innovative programs were increasingly harassed in accreditation reviews and were forced to make their curricular requirements more restrictive."

Based on these considerations, Prados *et al.* note, a new set of criteria for evaluating engineering programs, *Engineering Criteria 2000 (EC)*[1], was created. The specification of curricular content was significantly reduced in the new criteria. Instead, each program was required to identify a set of *program objectives*, tailored to the individual program, and a corresponding set of *outcomes*, including the set of twelve outcomes (3.a) through (3.k) specified as part of Criterion 3 of EC. At the *core* of EC is the requirement of a continuous improvement process based on assessing the degree to which graduates of the program achieved the program's outcomes and using the assessment results to drive program improvements. EC also requires clear documentation of the assessment processes used, the assessment results, and the improvements based on these results.

While the curricular flexibility provided by EC has been widely welcomed, many programs have struggled to meet the requirements regarding suitable assessment processes and documented improvements based on the results of the assessment of their outcomes. One key problem has been the question of developing suitable mechanisms for direct assessment of program outcomes. 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. By contrast, *indirect assessment*s are things such as opinion surveys completed by students. One occasional distinction that is made between direct and indirect assessments is that the former is quantitative and the latter is qualitative. That is incorrect; a direct assessment may well be qualitative and an indirect one may be quantitative. The key requirement for 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.

Although direct assessment of outcomes is not an explicitly specified requirement of EC, most program evaluators expect to see some use of such assessments. This is not surprising since direct assessments provide the most reliable assessment of the extent to which various outcomes are achieved since, as noted above, they are based on evaluation of actual student performance in relevant activities, rather than opinion surveys and the like. 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 has been faculty time and other resources involved in administering the assessments, collecting and collating the data, etc. In this paper, we report on a set of mechanisms for assessing program outcomes that is both powerful in terms of its ability to identify specific program improvements and, at the same time, requires only minimal resources to administer and sustain on a long term basis. Moreover, only a modest amont of effort and resources are needed to document the assessments, its results, and the program improvements based on the results.

Criterion 3 of EC lists a set of eleven outcomes (3.a through 3.k) that all engineering programs are required to show that their graduates achieve. Individual programs may include, in their set of outcomes, additional ones. Typically, these additional outcomes are specific to the particular engineering discipline and, in some cases, are dictated by the additional criteria that programs in the particular discipline may be required to meet. In this paper, we will focus attention on the common (3.a) through (3.k) outcomes since these apply to all engineering programs. These eleven outcomes may be classified into two groups. The first group, consisting of (3.a), (3.b), (3.c), (3.e), and (3.k) are *technical* outcomes; for example, outcome (3.a) is *an ability to apply knowledge of mathematics, science, and engineering*. The second group consists of the remaining outcomes, (3.d), (3.f), (3.g), (3.h), (3.i), and (3.j), are related to what might be called *professional skills*[20] (also occasionally referred to as *soft skills*), as well as those related to *societal* issues. Thus outcome (3.d), related to a professional skill, is *an ability to function on multi-disciplinary teams*; outcome (3.h), related to societal issues, reads, *the broad education necessary to understand the impact of engineering solutions in a global, economic, environmental, and societal context*.

This separation into two distinct groups is related to the fact that almost every one of the courses in a typical engineering curriculum would contribute to developing the outcomes in the first group. By contrast, the outcomes in the second group would mainly be *developed* in other courses such as *general education courses* and *applied* in specific high-level engineering courses such as the capstone design course. Moreover, and more importantly, by the very nature of the outcomes in

the two groups, methods used to assess outcomes in the first group are not likely to be appropriate for assessing outcomes in the second group and vice-versa. Therefore, in the our program at Ohio State, we use two distinct ways for assessing the outcomes in the two groups.

The key part of our approach to direct assessment of the technical outcomes, which is the focus of this paper, may be summarized as follows. We should note that our program is a *Computer Science and Engineering* (CSE) program. As a result, some of the details here and in later sections will be somewhat CSE-specific. However, the underlying approach is applicable to all engineering programs. The essential component of the approach is *POCAT* (for *program outcomes achievement test*), a multiple-choice test that students are required to take near the time of their graduation from the program. The questions on POCAT are based on topics from various high-level, required courses in the program and are chosen to ensure broad coverage of the program's technical outcomes. The questions on POCAT are *not* similar to the detailed ones one might expect in the final examinations of the particular courses. Rather, they are intended to evaluate how well students understand *essential* concepts of the course, how well they apply them to specific problem situations, and how well they connect them to appropriate ideas from other related courses in the curriculum. In Section 2 and beyond, we will provide more complete details of our approach. As we will see, there are some similarities between the POCAT-approach and the idea of *concept inventories*[18,8,10].

For the second group of outcomes, the approach we adopted was to introduce a carefully designed set of activities in our capstone design course(s) and in the required course on social and ethical issues in computing. These activities were directly related to the various items of knowledge and skills in the various outcomes of this group. These activities are assessed by a carefully designed set of *rubrics*. These activities and assessments are somewhat similar to approaches adopted by many other engineering programs (see, for example,[14,20]). Because of this, we will not consider these outcomes further in this paper, focusing instead on POCAT and how it enables us to assess the first group of outcomes and arrive at improvements related to those outcomes.

The rest of the paper is organized as follows. In Section 2, we review background material. In particular, we review the problems that various programs have encountered in implementing direct assessments of EC outcomes. We also briefly review the background on concept inventories. In Section 3, we present the details of POCAT. In Section 4, we consider the resources, especially faculty time and effort, needed to implement and sustain the POCAT-based approach to assessing the technical outcomes and the effort needed to evaluate the assessment results and identify possible program improvements. In Section 5, we present some of the results, in the form of ideas for program improvement, obtained by using this approach over the last two years in our program. In Section 6, we summarize our work and consider future plans.

2. Background

From the earliest days of EC, one of the key questions that programs have struggled with is 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"[19].

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[17,13]. 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 assessing the information collected *and* arriving at possible improvements in the program based on the assessment results can be overwhelming. Indeed, some 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 the e-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[16,3]. As we will see in the rest of the paper, our approach attempts to minimize the volume of data collected, summarizing information at each stage, rather than saving up all of the student work.

Another approach to direct assessment has been the idea of using *targeted* questions in examinations in particular courses in the curriculum, see, for example,[2,7]. 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. 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. Also, the emphasis seems to be on assessment for the sake of assessment rather than for the sake of program improvement. Danielson and Rogers[4] and Howard and Musto[11] both present a somewhat similar approach. In both cases, 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, this approach is not high-stakes in the same manner as that of[12]. Nevertheless, Helps, Anthony and Lunt[9] point out, such approaches are very expensive in terms of faculty time and effort.

Perhaps more importantly, such approaches tends to take a *course-level* view rather than a *program-level* view. By contrast, the POCAT approach is likely to provide better results (in terms of ideas for program improvement), provide historical data in a more comprehensible manner, take a program-level view, and require fewer resources to administer and sustain.

The idea of *concept inventories*[18,8,10] has some resemblance to the POCAT approach. The idea of the concept inventory was created by Haloun and Hestenes. The original inventory was for Newtonian Mechanics. It was intended to assess students *conceptual* understanding of key principles of the subject. The exam was a multiple choice test with each question containing distractors that are 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 has shown that a well-designed 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. The Foundation Coalition's site (www.foundationcoalition.org) provides many resources on the topic.

3. The POCAT Approach

Before considering the details of our approach to direct assessment in this and in the next section, it may be useful to summarize the goals we tried to achieve:

1. Satisfactorily meet both the letter and the spirit of the accreditation criteria, meeting or exceeding the expectations of accreditation evaluators;
2. Ensure that the results of the assessments produce actual ideas for improvements in the program, not just be performed for the sake of meeting accreditation requirements; and

3. Ensure that the faculty effort and other resources required to perform, document, and continue the assessment activities are moderate and sustainable over a long time.

The last goal would seem to be necessarily in conflict with the other two goals. Nevertheless, our approach has been able to achieve a fair degree of balance between them.

The *Program Outcomes Achievement Test* (POCAT) is designed to test student achievement of the first group of program outcomes, i.e., (3.a), (3.b), (3.c), (3.e), and (3.k). It may seem that some of the details below are somewhat discipline-specific, i.e., Computer Science and Engineering (CSE). In fact, however, the underlying approach is applicable to all engineering programs. The POCAT is a test that students are required to take near the time of their graduation from the program. The questions on POCAT are based on topics from nine required high-level, courses in the program. These courses relate to a variety of key CSE topics such as software engineering, formal languages and automata theory, databases, programming languages, computer architecture, algorithm analysis 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*. In other words, the questions attempt to evaluate not only how well the student understands key concepts from the individual courses but also how well she is able to relate concepts presented in one course to problems and concepts presented in a later, related course. For example, a question may probe whether a student is able to apply the concept of *finite state machines* (from the course on automata theory) to the problem of designing a *tokenizer* for a compiler for a programming language; another may probe whether a student is able to apply ideas introduced in the algorithm analysis course to evaluate alternative ways in which a database might be used to solve a particular problem. Each question is typically the result of sometimes quite extended discussions among faculty involved with a number of these courses. 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. Indeed, because of the very nature of the questions and because of the broad scope of many of the outcomes in this group, many of the questions tend be related to more than one outcome in the group. A sample of the POCAT test is available on our web site[6].

All students in the program are required to take the POCAT two or three months prior to graduation from the program. But a student's performance on the test does *not* affect his or her grades 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. A sample set of results is available on our web site[5]. The idea of not publishing or even collecting information about the performance of individ-

ual students on the test was a deliberate decision by the faculty in the department since we did *not want* the students to spend a lot of time preparing for the test. The key purpose of the test was to help us assess the extent to which the students had acquired and internalized the knowledge and skills associated with the various outcomes, not how well they could prepare for tests.

Initially, there was a concern that if the individual student's performance on the test did not affect them in any tangible way, they would not take the test seriously. Our experiences with the seven or eight sets of POCATs that have been administered since we created the test have completely eliminated that 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.

Given the nature of the questions, the grading of the tests is essentially mechanical. The faculty members who were responsible for each question also 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. All of this information is automatically included in the summary results that are produced (as may be seen in the sample set of results). This allows the department's Undergraduate Studies Committee to have a well-informed discussion about the extent to which the particular group of students has achieved these outcomes and identify potential problem spots in particular courses, indeed in particular topics, and bring them to the attention of the appropriate group of faculty. This enables the relevant faculty to consider possible changes in various courses to address the problem. Over the last two years, a number of such improvements have been made in a number of courses in the program. In the next section, we will consider some details of the resources required to administer and sustain POCAT and to maintain documentation of the assessments. In the following section, we will consider some of the specific program improvements that POCAT has so far enabled us to identify.

We conclude this section by comparing the POCAT-approach with concept inventories. There are some obvious similarities, for example, in the types of questions used; but there are also some important differences. Perhaps the most important one is that while concept inventories are intended to be common to all programs in a particular discipline, the POCAT approach 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. All that is required is for 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.

4. Resource Requirements

Since the time EC was introduced, one of the key problems that engineering programs have had to face is that of resources for meeting EC's requirements. In particular, the resources needed to perform outcomes assessment on a continuing basis, document the results of the assessments, identify program improvements based on an evaluation of the assessment results, document these improvements, and sustain all this over the long term. As noted earlier, approaches such as collecting *portfolios* of student work are especially problematic since the volume of material that gets collected over time can become overwhelming. Thus, this was a key consideration in designing the POCAT-approach.

There are several different aspects of the POCAT-approach that require faculty effort and time as well other resources to deal with them. We consider each of these aspects in turn. The first has to do with actually designing candidate questions for POCAT. As noted earlier, for each of the (high-level) courses that are included in the "POCAT-courses", faculty who regularly teach the particular course identify important, central concepts of the course, common student misunderstandings and difficulties related to each of these concepts. They then design suitable multiple-choice questions related to each concept. A key consideration in designing the questions is coming up with suitable *distractors* that correspond to common student misunderstandings. This can be quite a challenging task and it is often not possible to come up with the right set of distractors in the first attempt; or perhaps the actual wordings of the various choices included in the question are such that they include critical clues to the correct answer that students can use to eliminate one or more of the distractors.

An example might help illustrate the issues. One of the courses included in the POCAT is CSE 655, a required senior-level course on concepts of programming languages including, as a key component of the course, standard implementation techniques used for standard languages such as *Java* and *C++* and issues that must be addressed in such implementations.

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 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 forget 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 re-executed 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 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 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* (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", is important. As we noted earlier, in POCAT, how well or poorly a student does has no impact on his or her academic standing or records. Thus, for students who really do not know the answer to the question, and *know that they do not know*, the last answer is the one to pick. 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.

Designing such questions, as noted earlier, can be quite challenging. 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. As noted earlier, the fact that student performance in the POCAT does not affect their academic record means that the test helps us assess the program rather than how well students can prepare for the test. In the current context, this same fact provides additional help. If a student's performance on the test affected his or her academic record, we have to be concerned about the *security* of the test; and would probably have to design new questions each time the test is offered. Given the challenges involved in designing high-quality questions of this kind, this would be quite difficult. But since there is no incentive for students to cheat and try to get advance access to the POCAT questions, we can freely reuse questions from one administration of the POCAT to the next.

The second aspect of POCAT to consider, from the point of view of resource requirements, is the actual administration of the test, the grading of the tests, and collating the results in an appropriate manner. Students in the CSE program who are near graduation are required to sign up to take the next POCAT. The test is administered on a weekday evening, typically starting at 5:30 pm.

Individual copies of the test have a unique (one- or two-letter) code on them and students pick one up as they come in; the students are specifically instructed not to write their names on the test and that they should remember the code assigned to their test sheet if they want to be able to see how they performed. Students typically take about 25-30 minutes to complete the test (consisting of about twenty questions similar to the one discussed earlier in the section. Following the test, pizza and soft drinks are served next door; students typically stay on for another 30-45 minutes, discussing the test as well as each others' plans following graduation. A staff adviser is present during the test and takes care of these details. If there are technical issues about any of the questions on the test, students are asked to make any needed reasonable assumptions and answer the question as best as they can. Occasionally, a student will be curious about the purpose of the test; the staff adviser who understands POCAT quite well is able to explain this.

The grading of the tests and the generation of statistical data such as what percentage of students answered a given question correctly and what percentage of students chose a particular distractor for the question are automated. The *Perl* script that does this produces a web page (see[5] for a sample) that summarizes the information. This entire aspect of POCAT takes minimal resources, typically about four to six hours of the staff advisor's time. Note also that these web pages, along with the actual tests which are added to an internal web site when each test is created, can serve as the documentation of the assessment required by EC with no additional effort being needed.

The final aspect of POCAT is the evaluation of the results and arriving at ideas of program improvement based on the results. The initial discussion of the POCAT results takes place in the program's Undergraduate Studies Committee. The committee consists of range of faculty including several who regularly teach the high-level courses included in POCAT. There are also student representatives on the committee. In addition, the staff adviser (who takes care of administering the test) is also a member of the committee. 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 particular 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 or the other seemed to provoke the most intense debate among students. And, of course, 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, the appropriate course of action might be to make suitable changes in the particular course or, possibly, in prerequisite courses. 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 provided 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 do 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.

The evaluation of POCAT results and determining appropriate courses of action clearly requires considerable faculty effort and time. But that is the point of assessment – using the results to identify possible improvements in the program, as well as improvements in the assessment itself. And, as noted before, far from resenting this work, faculty enjoy this activity since it helps them identify particular conceptual problems that students taking their courses commonly encounter and challenges them to revise the courses to address these problems. What does not require more than a very modest extent of resources is administering the tests, grading them, arriving at suitable statistical analysis of them, and documenting them.

5. Results

In this section, we will briefly summarize some of the results, in the form of program improvements, that have been arrived at on the basis of the POCAT results. For each improvement, we also specify which particular EC outcome(s) it is related to. Other programs that adopt the POCAT approach can expect the results of their assessments to suggest other ideas for improvement in their programs, ideas that will depend on the particular courses that they choose to include in their POCAT etc. That, of course, is expected and, indeed, desirable since ideas for program improvements should depend on the current state of the program, the details of the courses, the particular strengths that the program's faculty want students to acquire, etc.

1. Outcome 3.c: In some ways, this outcome which reads, "an ability to design a system, component, or process to meet desired needs within realistic constraints such as economic, environmental, social, political, ethical, health and safety, manufacturability, and sustainability", captures the essence of engineering. Naturally, almost everyone of our technical courses contributes toward developing this outcome and a number of questions on the POCAT are related to it. One of these questions asked students, given a problem scenario involving the design of an intricate (but small) piece of software, to choose from among a specified set of data structures with an eye toward maximizing performance. The Undergraduate Committee concluded, based on student performance in this question that although they certainly understood the importance of the issue and approached the problem logically, the students were not sufficiently clear about how to choose among the various options. Following this, the course instructors for the related courses have introduced a new set of activities to help students develop a better understanding of the issues.

2. Outcomes 3.e, 3.k: Outcome 3.e reads, "an ability to identify, formulate, and solve engineering problems. Outcome 3.k reads, "an ability to use the techniques, skills, and engineering tools necessary for engineering practice". The POCAT question described in Section 3, given the problem of uninitialized variables that is so prevalent in large pieces of software and given the importance of Java in professional CSE practice, is clearly concerned with these outcomes. In a recent POCAT, this question produced somewhat unexpected results. More students than we had expected chose answer (3), that *Java* does automatic initialization of all variables, rather than the correct answer, that the Java compiler uses a "conservative" approach to dealing with uninitialized variables. Following discussions in the Undergraduate Committee and discussions among faculty involved with the related courses, we are considering two changes; the first is to revise the CSE 655 course to have a slightly more detailed discussion of Java, given the practical importance of Java; second, the faculty also decided that it may be useful to rewrite the question without much reference to Java so that we can determine whether students are having conceptual difficulties with respect to the notion of uninitialized variables in programs and how systems can deal with it.

3. Criterion 3, 4: A key part of these criteria is the requirement of suitable assessment and evaluation processes whose results are used to improve the program. One of the natural and frequent results of the discussion and evaluation of POCAT results is ideas for improvements and revisions in the questions included in the test, i.e., in the assessment process itself. We mentioned above this with respect to the POCAT question related to uninitialized variables. As another example, POCAT includes a question concerning the concept of *relocation*. This has to do with the possibility of loading a piece of compiled/assembled code into a different part of the memory than it was originally intended for; or producing code in such a way that it can be loaded in any part of the memory. The question has to do with what changes have to be made in order for such relocation to work correctly. There are a number of complications here although the underlying concept is fairly straightforward. Thus designing a multiple-choice question with good distractors that helps distinguish between the various possible misconceptions that students may have, is a challenge. Based on student performance in several of the POCATs, the faculty involved with the related courses have gone through

two or three revisions of this; the most recent revision one will be used in the next POCAT and is expected to bring out student misconceptions rather than problems with the question.

One other point is worth noting. Often, a POCAT question relates to several different courses and concepts from those courses. For example, a question on *binary search trees* that is included in the test relates to the beginning software engineering sequence (where many different data structures are discussed) as well as in the algorithm analysis course where performance of various algorithms are analyzed. As a result, when student performance in this question was poorer than expected, faculty involved with both groups of courses consulted with each other before deciding on suitable changes in each group. In other words, the approach allows us to identify improvements not just in individual courses but also across the program curriculum.

6. Discussion

The main goal of this paper was to present a powerful approach to direct assessment of (many) program outcomes using POCAT, a carefully designed multiple choice test that students take when they are close to graduation. The test allows us to identify commonly held misconceptions about key principles and concepts from different, essential parts of the curriculum and directly leads to improvements in the program. Moreover, the resources required to administer the test, to grade them, obtain statistical information about student performance in various questions on the test, etc., are minimal. The main resource requirement, in terms of faculty time and effort, goes into designing suitable questions for the test, for analyzing the results of the test, and arriving at corresponding improvements in the program. On the student side, rather than resent having to take yet another test, students seem to enjoy it. It gives them a way to assess themselves and compare where they stand with their fellow students *without* having to worry about how it would impact their academic records and hence, possibly, their employment opportunities. Moreover, we do not have to worry about such issues as security of the POCAT questions since there is no incentive for the student to cheat! The approach is easily usable by other CSE, as well as non-CSE, programs.

We conclude with a comment about future work. A recent development with respect to assessment is the notion of *performance criteria* for various outcomes. That is, rather than assessing outcomes directly, programs are expected to list specific performance criteria for each outcome as well as expected levels of student performance for each criterion. It is the extent to which these criteria are achieved that the assessments should measure, rather than the outcomes directly.

This is a natural change to accommodate in POCAT. Each of our courses specify a set of intended learning outcomes which are essentially performance criteria for students taking the course. Thus it would be straightforward to map these performance criteria to the appropriate Criterion 3 outcomes. The real change that we would have to make would be to substantially increase the on-line bank of questions from which each POCAT is constructed to ensure that there are one or more questions for each of these performance criteria. Once this is done, as long as the bank contains information about which question relates to which performance criterion/criteria, very little additional work would be needed in summarizing the results by performance criteria.

References

[1] ABET. Engineering Criteria, www.abet.org, 2008.

[2] D Ahlgren and J Palladino. Developing assessment tools for ABET Engineering Criteria. In *Frontiers in Education*, pages T1A–17. ASEE/IEEE, 2000.

[3] C Chewar, K Huggins, and J Blair. Avoiding the pratfalls of program assessment. *SIGCSE Bulletin*, 38(4):29–33, 2006.

[4] 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.

[5] CSE Dept. Sample POCAT results. url: www.cse.ohio-state.edu/ñeelam/abet/DIRASSMNT/POCATRESULTS/autumn06Results.html, 2006.

[6] CSE Dept. Sample POCAT. url: //www.cse.ohio-state.edu/ñeelam/abet/DIRASSMNT/pocatSampleTest.pdf, 2006.

[7] R Felder and R Brent. Designing and teaching courses to satisfy ABET EC. *J. of Eng. Education*, 92(1):7–25, 2003.

[8] I Halloun and D Hestenes. Common sense concepts about motion. *American Journal of Physics*, 53:1056–1065, 1985.

[9] CR Helps, D Anthony, and B Lunt. Outcomes oriented ABET accreditation: Mechanisms for review and feedback. In *Proc. of the ASEE Annual Conf.* ASEE, 2005.

[10] D Hestenes, M Wells, and G Swackhamer. Force concept inventory. *Physics Teacher*, 307: 141–151, 1992.

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

[12] F Mak and S Freza. Using student learning outcomes assessment to assure EC2000 program effectiveness. In *Proc. of the ASEE Annual Conf.* ASEE, 2005.

[13] J Mcgourty, L Shuman, M Besterfield-Sacre, C Atman, R Miller, B olds, G Rogers, and H Wolfe. Preparing for ABET EC2000: Research-based assessment methods and processes. *Int. J. of Eng. Ed.*, 18(2):157–167, 2002.

[14] B Olds, B Moskal, and R Miller. Assessment in engineering education: Evolution, approaches and future collaborations. *Journal of Engineering Education*, 94(1):13–25, 2005.

[15] JW Prados, GD Peterson, and LR Lattuca. Quality assurance of engineering education through accreditation: The impact of Engineering Criteria 2000. *Journal of Engineering Education*, 94(1):165–184, 2005.

[16] G Rogers. Death by assessment: How much data are too much. Technical Report Spring 2002, Communications Link, 2002.

[17] G Rogers. Got portfolios? Technical Report 07-04-CM, ABET Newsletter, 2007.

[18] A Savinainen and P Scott. Force concept inventory: a tool for monitoring student learning. *Physics Education*, 37:45–52, 2002.

[19] J Shaeiwitz and D Briedis. Direct assessment measures. In *Proc. of ASEE Annual Conference*. ASEE, 2007.

[20] L Shuman, M Besterfield-Sacre, and J McGourty. The ABET "professional skills" - can they be taught? can they be assessed? *Journal of Engineering Education*, 94(1):41–55, 2005.