AC 2007-2582: A PROCESS FOR THE DIRECT ASSESSMENT OF PROGRAM LEARNING OUTCOMES BASED ON THE PRINCIPLES AND PRACTICES OF SOFTWARE ENGINEERING

Robert Lingard, California State University-Northridge

A Process for the Direct Assessment of Program Learning Outcomes Based on the Principles and Practices of Software Engineering

Abstract

The Computer Science Department at California State University, Northridge (CSUN) has developed and is currently utilizing a process for the direct assessment of program learning outcomes that follows an approach similar to that used in the development of software. The software engineering steps have been applied in an iterative manner to the process of direct assessment. Before the development of this approach, the assessment of learning outcomes within the department was based primarily on indirect measures, such as student self-assessments or the subjective views of faculty, employers, and others. Additionally, when learning was assessed directly, it was usually in the context of specific courses. The needs to directly assess student learning and to assess how well it was retained over time were recognized.

The new assessment process was developed to meet the following goals. First, it needed to facilitate the direct assessment of student learning outcomes and to measure not just the skills and knowledge of students when they completed a specific course, but what they retained as they neared completion of the program. Second, it needed to be a continuous process that ensured the assessment of all program outcomes over a reasonable length of time. Third, the process should be able to be incorporated within the existing operations of the department and the activities of the individual instructors. That is, it needed to be efficient and not unduly burdensome for members of the faculty. Finally, it needed to be sufficient to satisfy the ABET requirements for assessment, as well as those of the University.

The four major steps in software development, *requirements analysis, design*, *implementation*, and *validation* have been applied to the assessment process. In the requirements analysis step, a determination is made of which learning outcomes to assess during the current cycle. The most important outcomes, possibly based on the results of indirect assessments or previously conducted direct assessments, are chosen. In the design step, for each selected outcome, a group of faculty consisting of all those teaching courses strongly related with respect to given outcome is charged with the responsibility of developing an assessment plan. This activity involves the selection or development of assessment instruments and rubrics. During implementation the assessment plan developed during design is carried out and the results are analyzed. Based on this analysis the validity of the results is determined. In the case of valid results, recommendations for program improvement are made as appropriate. For assessments considered invalid, recommendations for improved ways to assess the outcomes in question are made. This process then goes back to the first step for the next iteration.

The process described above has been operational for the last two years and has been proven to be very effective with respect to gathering assessment data. Both valid and invalid assessment results have been obtained using the process. Many of the successful assessments have led to important program improvements, and the unsuccessful attempts have resulted in improvements in assessment techniques.

Introduction

Assessment of learning outcomes has become increasingly important over the past several years. Accrediting institutions such as ABET have transitioned to a model based on learning outcomes, a model that heavily involves the assessment of student achievement with respect to these outcomes, requires feedback of assessment results, and demands their use for continuous improvement of the program. Previously, ABET evaluated programs based only whether they offered an appropriate number and variety of courses in computing, scientific, mathematical, and general education areas, had well qualified faculty, and provided the resources necessary to support the program. The new model requires that institutions take a greater responsibility for evaluating their own performance based on the assessment of student achievement as measured against their own criteria.[1] Assessment has become essential for accreditation in today's environment.

Accreditation aside, the need for program assessment has become important recently as more of the stakeholders in the educational process are demanding accountability from the education providers. Universities, especially public universities, are frequently being asked to account for how they spend the educational dollars they receive. Students, parents, governments, and employers all want to know that a quality education has been provided and that the money they provided to do so was well spent. This is not an unreasonable request. Assessment is the primary means by which evidence can be gathered to demonstrate that graduating students possess the appropriate knowledge and skills to allow them to be productive members of society in their chosen fields.

The third, and possibly most important, reason for assessment of student achievement within an educational program is that those directly involved in the learning process need to know how well they are doing and what can be done to improve learning. Most who are involved in teaching have a sincere desire to improve learning whenever and wherever possible. But how can improvements be made, or even the need to make changes be recognized, without understanding how well students are learning with present approaches? Assessment is a form of measuring and measuring is important for understanding. As Lord Kelvin said, "When you can measure what you are speaking about . . . you know something about it; but when you cannot measure . . . your knowledge is of a meager and unsatisfactory kind."[2]

Given the importance of assessment the question is how to accomplish it in a meaningful and useful way within the budget and time constraints of the organization? The specific approach used within the Computer Science Department at California State University, Northridge is described below. The process utilizes principles and practices fundamental to software development, namely understanding the needs (requirements analysis), planning the process (design), implementing the plan (implementation), and evaluating the results (validation). Furthermore, this approach is designed to involve all faculty in the department, be part of and not separate from the existing educational process, support the direct assessment of program learning outcomes, and minimize the need for additional resources.

Problems with Assessment

Because of the assessment requirements placed on organizations by accrediting bodies such as ABET and the stakeholders of the educational process, there is often a tendency to do assessment for assessment's sake. That is, the attitude can be to "just get some assessment done" in order to satisfy the requirement. Rather than attempting to determine what assessments are really most important, assessment is done in places where it is easiest to do. For example, if a standardized test exists that can be easily given to students and the results automatically tabulated, there might be a temptation to use it even it's applicability to the particular program is marginal or if it does not addresses areas of greatest need for assessment. In this "rush to assessment" the goal is to get something done quickly that minimizes the impact on faculty and other resources.

In a similar fashion, in order to "close the loop," program changes are made without even validating the results of the assessment. "Closing the loop" refers to the ABET requirement that not only are program learning outcomes assessed, but that the results of the assessments are used for making program improvements. The ABET guidelines do not explicitly address the issue of validating assessment results, and in a desire to demonstrate that the loop has been closed, there is often a premature attempt to make program changes. Blaha and Murphy make several recommendations for creating a computing assessment plan, and they caution against compressing the assessment cycle "in order to have obvious accomplishments to report to the accreditation body at the time of review."[3]

Another problem with assessment is that it is often used to "showcase" a program by assessing only the areas in which high levels of student achievement can be shown.[4] At the very least there might be a reluctance to assess the real areas of educational concern because the results might expose weaknesses in the program. When the accreditation visit occurs, the senior leaders of a program want it to be seen in the best possible light. Accrediting bodies requiring assessment must be careful not to penalize programs for weakness uncovered by the process. Otherwise, the most important assessments will not be done or, at least, not reported.

Approach to Assessment

According to Diane Halpern, writing on the science of learning, "the first and only goal [in education is to] teach for long-term retention and transfer."[5] She adds, "We only care about student performance in school because we believe that it predicts what students will remember and do when they are somewhere else at some other time." Although assessing student learning within a given class is important in order to understand instructional effectiveness with respect to course objectives, it doesn't tell us much about whether the material will be retained or whether students will be able to transfer what they have learned to new situations. Meaningful assessment of learning can only be done at some later time. In the approach discussed here, program learning outcomes are assessed as students near graduation in courses where they apply the knowledge and skills learned earlier. Although this doesn't guarantee that they will be able to use what they have learned in out-of-school contexts, it certainly provides a greater degree of assurance that they have retained what they have learned and will be able to apply it in work and life experiences.

Another aspect of this approach is that it relies primarily on the direct assessment of learning outcomes. Traditionally the assessment of program outcomes has relied primarily on indirect measures, in particular, surveys. Sanders and McCartney report the results of two surveys on the use of various assessment instruments which show that senior exit surveys and alumni surveys are the most used ways of obtaining assessment information.[6] Direct measures such as exams, either internally or externally developed, and portfolios are used much less frequently. Meaningful assessments of student retention and their ability to apply what they have learned can only be obtained by directly measuring student performance, not just by asking them if they can do it. Our approach emphasizes the measurement of student achievement with respect to program outcomes, not at the time they first acquire the knowledge and skills, but later in the program when they are required to apply what they have learned.

Applying the Software Engineering Paradigm

The four most fundamental activities in software engineering are:
1. Requirements Analysis (understanding the problem)
2. Software Design (planning a solution)
3. Implementation/Coding (carrying out the plan)
4. Testing/Validation (making sure the solution is correct)

An analogous set of activities can be applied to the process of assessing program outcomes. The first step is to decide what needs to be assessed. It makes no sense to try to assess everything at once. Likewise, randomly picking some set of outcomes to assess seems equally senseless. Some process is required to decide what is in most need of assessment. One approach is to have the faculty teaching courses related to a given program outcome informally assess how well this outcome is being met. This can be done in a meeting of appropriate faculty by examining and discussing student work and performance in the related courses. It is usually not difficult to determine whether or not students are learning as expected or whether problems may exist. Based on this informal assessment recommendations can be made on the need for formal assessment of a particular program outcome. The recommendations regarding each of the program outcomes can be compared and the outcomes where the greatest need for formal assessment can be selected for further study. Whether this approach is used or some other, the point is to carefully examine student achievement of all program outcomes and to select a subset for formal assessment based on where there is the greatest need for more information.

The second step is to develop a plan for assessment. For each program outcome selected for formal assessment, a plan needs to be developed identifying how, when, and by whom the assessment is to be accomplished. This step involves the identification or development of instruments for assessment, the design of rubrics for measuring the results, and the establishment of a schedule for the assessment activities that identifies who will perform the work items required and when each will be accomplished. In software engineering the highest percentage of errors in the final product are traced back to the design. Similarly, assessment results that fail to provide useful information are most likely caused by a poor design of the assessment activity. Frequently the instruments used do not really measure the intended student achievements or they are not given at the right time, to the right students, or to enough students for the results to be considered meaningful. It may be necessary to conduct pilot studies to validate a proposed

assessment instrument before using it. Assessment activities conducted without appropriate design and planning can merely consume lots of time and resources without producing any results of real value.

The third step is to implement the assessment plan. With a good plan carrying out the assessment activities should be fairly straightforward. However, a problem that often occurs is that the plan is not followed. As in software engineering, effort must be given to the task of monitoring progress with respect to the plan. Since many faculty members are likely involved in one way or another, periodic meetings may be necessary to examine the status of the plan. It is probably wise to have one individual in the department who has the designated responsibility of monitoring assessment plans. Only if the assessment activities are implemented as planned can there be assurance of achieving successful results.

Finally, the last step is validation of the results. This step is often omitted or given only cursory attention. However, this activity is of vital importance. Just as in software engineering, if the final result is not valid, the effort cannot be used as a platform to build upon. ABET and other accrediting organizations require that assessment results be used to make improvements to the program. However, care must be taken to make sure the assessment results are valid before considering changes to the program in the name of improvement. It is easy to jump to conclusions after examining assessment results, especially when there is a perceived need to "close the loop" by making "improvements" to the program. This often overlooked and frequently not mentioned part of the assessment process may be the most important of all. When inappropriate program changes are made, the waste of resources can be extensive. Blaha and Murphy describe a case where a new required course was added to the curriculum based on assessment data whose validity was later brought into question.[3] Attempts should be made to compare assessment results from a variety of approaches in order to be convinced of their validity. Only when program changes are made based on valid assessment results can there be any assurance that those changes will result in improvements to the program.

Like software engineering this process is iterative. That is, even when one set of assessment activities are completed and program changes have been made, the newly changed program must be assessed. One must go back to the first step, look at the program outcomes again, and determine the needs for assessment. Primary among those needs will be the need to assess the extent to which any program changes have improved student learning. This again is an often overlooked step in the assessment process. It is not acceptable to merely make program changes based on assessment results; one must assess the effectiveness of these changes. Only if the changes result in actual improvements to the program should they be continued.

The Assessment Process

Before discussing how the software engineering paradigm has been applied at CSUN, it will be helpful to understand the organizational environment in which assessment is accomplished. Within the Department the courses that are part of the Computer Science Program are divided into Program Areas of related courses as shown in Table 1. This approach was adapted from a similar approach used within the Computer Science Program at Ohio State University that established Course Groups.[7] Each Program Area is associated with one or more of the program

PROGRAM AREA	COURSES	PROGRAM OUTCOMES
Fundamental Concepts	• Intro. to Algorithms & Prog. • Data Structures & Prog. Design • Advanced Data Structures • Plus Electives	• Demonstrate an understanding of algorithms and data structures. • Demonstrate proficiency in using a high-level computer language. • Demonstrate a problem solving ability.
Systems	• Computer Architecture • Computer Organization • Operating Systems & Sys. Arch. • Plus Electives	• Demonstrate an understanding of computer organization and architecture.
Language/Theory	• Concepts of Prog. Languages • Automata, Languages & Comp. • Discrete Mathematics • Combinatorial Algorithms • Symbolic Logic • Plus Electives	• Demonstrate an understanding of programming language concepts and knowledge of a variety of programming language paradigms. • Demonstrate an ability to apply mathematical skills appropriate to the computer science discipline.
Software Engineering	• Intro. to Software Engineering • Plus Electives	• Demonstrate proficiency in collecting, analyzing, and interpreting data. • Demonstrate an understanding of emerging technologies and a working knowledge of currently available software tools. • Demonstrate an understanding of the principles and practices for software design and development. • Be able to apply the principles and practices for software design and development to real problems.
Societal Issues	• Societal Issues in Computing • Plus Electives	• Demonstrate an awareness of the evolution and dynamic nature of the foundational core of computer science. • Demonstrate knowledge of the social impact of computing. • Demonstrate an understanding of the professional and ethical considerations of computing.
Communications	• Intro. to Software Engineering • Societal Issues in Computing • Plus Electives	• Be able to effectively communicate orally. • Be able to effectively communicate in written form. • Be able to work effectively on a team.
Lifelong Learning	• Societal Issues in Computing • Plus Electives	• Demonstrate the knowledge and capabilities necessary for pursuing a professional career or graduate studies • Demonstrate the recognition of the need for, and ability for, continuing professional development.

Table 1
Program Areas

outcomes. Associated with each course is a set of course objectives which specify the competencies each student successfully completing the course should have. Each of these course objectives is related to one or more of the program outcomes. Courses are assigned to Program Areas based on the relationship between their course objectives and the program outcomes that are part of a Program Area. Each course is in at least one Program Area, but sometimes a course might be in multiple areas. The table above shows the mapping between Program Areas, courses, and program outcomes for the Computer Science Program at CSUN.

The Program Area coordinators comprise the Department's Assessment Committee. This committee is responsible for the outcomes selection, assessment planning, implementation, and validation of results for the assessment activities of the Department. The committee prepares recommendations for program improvements based on assessment results and presents them to the Department for discussion and to make decisions relative to implementation. The Department acts as a Curriculum Committee of the whole to approve any program changes after a thorough examination of the assessment results. The annual process for the formal assessment of program learning outcomes is shown in Figure 1 and described below.

1. At the third department meeting during the spring semester, area coordinators for each of our "program areas" are elected (Each Computer Science course in the program has been placed into one of seven program areas, and each area has a coordinator.) Current coordinators may be re-elected, and there are no term limits. This set of program area coordinators constitutes the Assessment Committee for the year.

2. During the first month of the spring semester the Assessment Committee identifies the program outcomes to be formally assessed during the current calendar year and presents this list for department approval no later that the first department meeting in March. Outcomes are selected using the following criteria:
    a. The primary outcomes selected for assessment are those identified as most in need of formal assessment based on informal assessments from the previous year.
    b. Any outcome not assessed during the last four years must be included (except for the first four years this process is used). This will insure each outcome is formally assessed at least once every five years. (Each outcome, if not formally assessed, is informally assessed annually.)
    c. Any outcome formally assessed in the previous four years for which the assessment results are deemed inadequate or for which some program change has been implemented to improve an uncovered problem or concern should be included for reassessment.
    d. The number of outcomes to be formally assessed should be at least one fifth of the total number of outcomes.

3. Upon department approval of the outcomes to be formally assessed, the Assessment Committee will assign assessment responsibility for each selected outcome to the appropriate program area coordinator. Prior to the first department meeting in April, the program area coordinators will prepare an assessment plan for each of the outcomes for which they are responsible. The emphasis is on direct assessment of the outcomes. These plans will be presented for departmental approval during the first Department meeting in May.

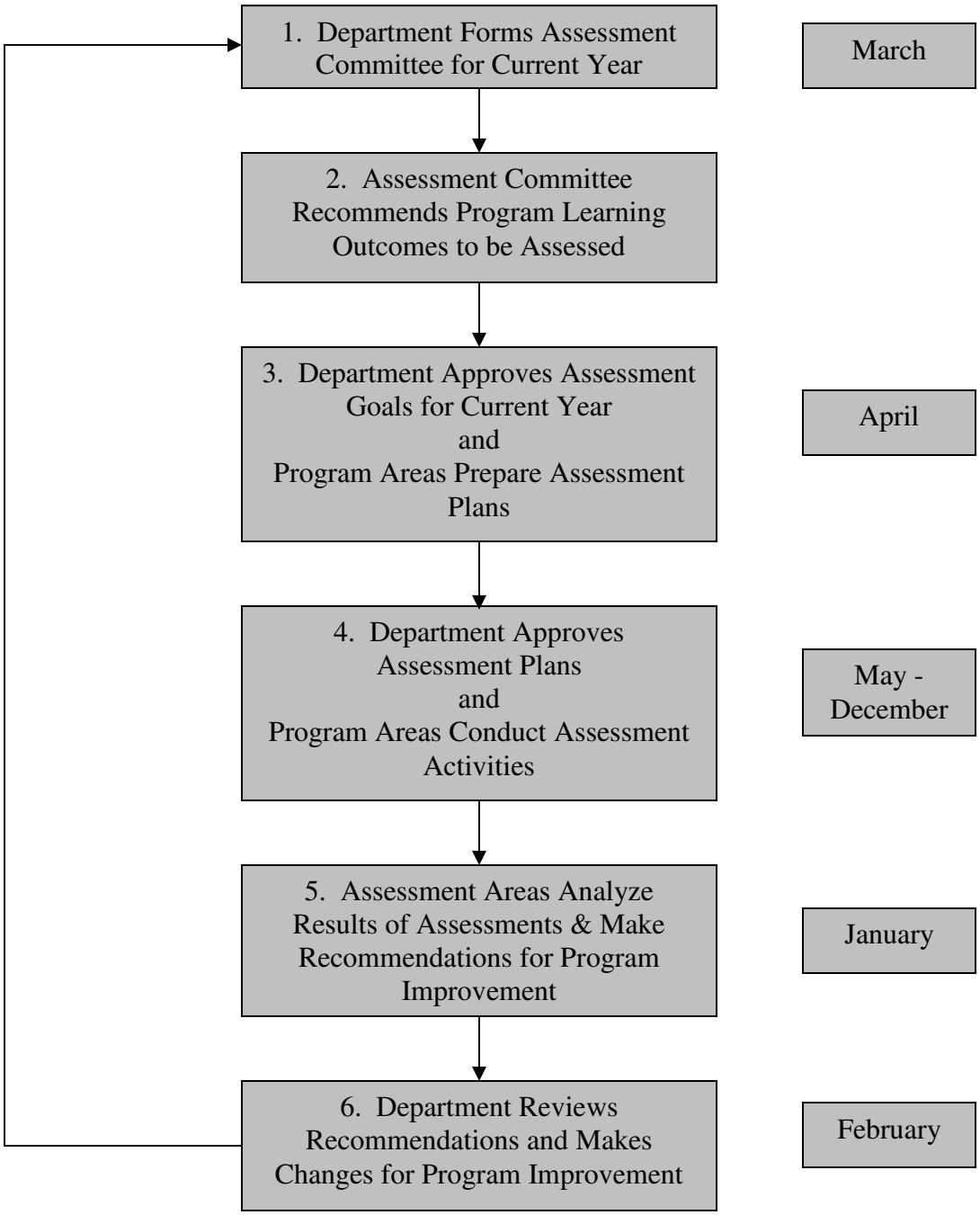


Figure 1
Assessment Process

4. Once the assessment plans are approved, the program area coordinators ensure that the planned and approved assessment activities are accomplished and prepare a final report, prior to the last Department meeting of the year, summarizing and evaluating the results of these activities

5. Prior to the third Department meeting of the spring semester the assessment reports from all Program Areas are analyzed by the Assessment Committee and recommendations for program improvements are made.

6. During the third Department meeting of the spring semester, the program improvement recommendations from the Assessment Committee are discussed, and the Department decides what actions are appropriate based on these recommendations. The process then loops back to step 1.

Example Results

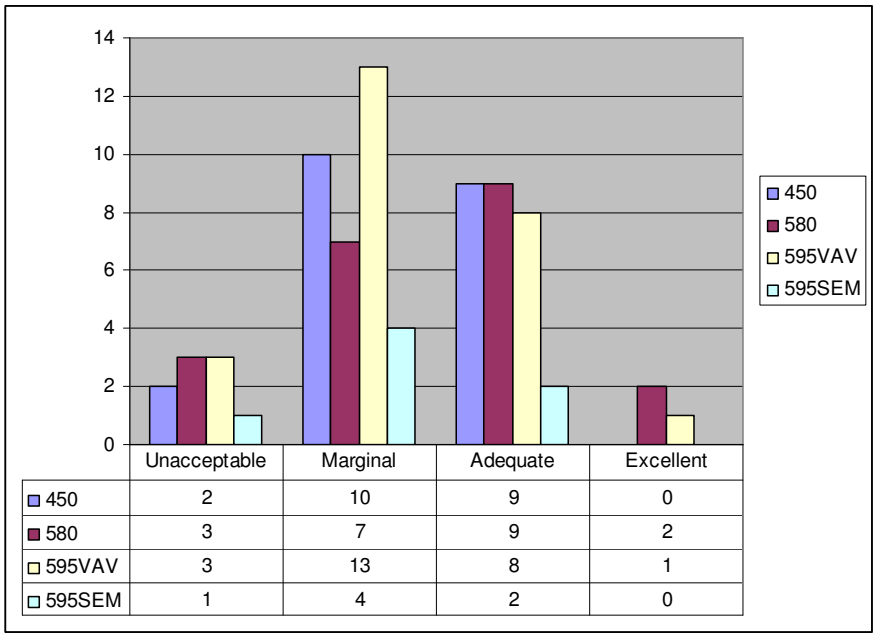
One case that probably best illustrates many of the points made about assessment in this paper is the assessment activity associated with one of our Program Learning Outcomes relating to software engineering. The outcome identified for assessment was that students should be able to "demonstrate an understanding of the principles and practices for software design and development." This outcome was selected for formal assessment based, in part, on the recommendations of the Software Engineering Program Area group. Informal discussions led to a concern that students in elective courses were not proficient in applying the software engineering principles that they supposedly learned in an earlier required course. The Assessment Committee decided to make assessment of this outcome part of the plan for 2005. The decision to conduct this assessment was based on a careful analysis of the outcomes most in need of assessment.

An assessment plan was then developed for this outcome which included the design of a test to be given to students in various elective courses that required students to complete team oriented software development projects. These courses all identified the introductory course in software engineering as a prerequisite. Several of the past and present instructors of the introductory course participated in developing the test. A rubric for evaluating student performance was developed as well. Although the process was followed, the design of the assessment instrument was flawed as was demonstrated later in the validation step.

The assessment test was administered during the fall of 2005 to students in four elective courses. This implementation effort involved not only the instructors who developed the test but the instructors of the four classes where it was given. These faculty members had a vested interest in the results since their courses were designed based on certain assumptions about what students knew about software engineering. This assessment approach has been successful in maximizing the number of faculty involved in the process. Unfortunately, the results from this particular assessment were disappointing. As shown in Figure 2 below a majority of students fell into the marginal or unacceptable categories. In this chart the bars show the different courses in which the assessment was done, and the y-axis indicates the number of students in each category.

The initial analysis suggested that there were significant problems with the program with respect to the teaching of software engineering. But according to the process the results needed to be validated before any definite conclusions could be drawn. In fact, in attempting to validate the results, serious flaws were uncovered in the testing instrument. In a statistical analysis of the

questions and the answers given, it was found that many of the questions failed to correlate with overall success on the test. Further examination showed that some of these questions were ambiguous or did not test important concepts. In the final analysis, it was felt that there were more problems with the assessment instrument than with the students who took it. This experience demonstrated the importance of the instruments used for assessment and the need to validate results before using them to make program changes. The assessment instrument has been revised and is currently being used to gather new assessment data. That data will be analyzed and new recommendations for program improvement will be presented as appropriate.



	Unacceptable	Marginal	Adequate	Excellent
450	2	10	9	0
580	3	7	9	2
595VAV	3	13	8	1
595SEM	1	4	2	0

Figure 2
Software Engineering Assessment Results

Even though the results of the original assessment were placed in question, some of the recommendations for program change seemed appropriate in any case. The recommendations were as follows:

1. Add an additional software engineering course to the list of required courses for computer science majors.
2. Modify the course objectives for introductory computer science courses to include an introduction to software engineering concepts.
3. Modify the course objectives for elective courses with software engineering projects to include the reinforcement of software engineering concepts.

The general conclusion from the analysis of the assessment results was that the program did not include enough reinforcement of software engineering concepts. Recommendations 2 and 3 seemed reasonable and fairly easy to implement, and a decision to adopt them was made even though the assessment results could not be validated. Implementing recommendation 1, however, would have required a major change to the program, and a decision regarding its implementation was postponed until valid assessment data could be gathered.

Assessment is, or should be, an iterative process. It is difficult to get it right the first time. New attempts at assessment are difficult to design, and design flaws are often difficult to see until the assessment is completed. An effective process will include not only an analysis of assessment results, but an analysis of the assessment activity that produced them. Program changes should be based only on valid results. It is frequently necessary to improve the way an assessment was done and repeat it in order to obtain better results. This is an acceptable part of the process.

Conclusions

Early experience with this new assessment process has been very positive. The process has been embedded into the normal operations of the Department and as such has achieved the kind of continuity required. Because assessment has become an annual department activity, members of the faculty have accepted it as a normal part of the educational process, assuring its continued use. Since the process makes sure all program outcomes are formally assessed at least once in a five year period, no outcome can be overlooked for long. Care was taken to minimize the additional work required by individual faculty members. By spreading the assessment work over everyone in the Department, no one has become over burdened. This simultaneously solves two problems. It assures participation by all members of the faculty and, at the same time, minimizes the work any one individual has to do. Acceptance and participation by the faculty are essential ingredients in any successful assessment program. Most importantly, the emphasis on the direct assessment of program outcomes as students near graduation gives the greatest possible indication that students graduating from the program have retained and will be able to transfer the knowledge and skills they have learned to post graduation situations. Finally, we believe that the process we have put in place will satisfy the requirements of ABET as well as those of the University.

Bibliography

1. Helps, C. R. G., Lunt, B. M. and Anthony, D. K. (2005) ABET Accreditation with IT Criteria. In *Proceedings of the Special Interest Group on IT Education*, pp. 353-359.
2. Thompson, W. (1968), Lord Kelvin, Popular Lectures and Addresses [1891-1894]. In *Bartlett's Familiar Quotations*, Fourteenth Edition, p. 723a.
3. Blaha, K. D. and Murphy, L. C. (2001). Targeting Assessment: How to Hit the Bull's Eye. In *Proceedings of the Consortium for Computing in Small Colleges: Northwestern Conference*, pp. 112-121.
4. Chewar, C. M., Huggins, K. L. and Blair, J. R. S. (2006). Avoiding the Pratfalls of Program Assessment. In *The SIGSCE Bulletin*, Vol. 38, No. 4, pp 29 – 33.
5. Halpern, D. F. and Hakel, M. D. (2003). Applying the Science of Learning to the University and Beyond. In *Change*, July/August, pp, 36-41.
6. Sanders, K. E. and McCartney, R. (2003). Program Assessment Tools in Computer Science: A Report from the Trenches. In *Proceedings of the Special Interest Group on Computer Science Education*, pp. 31-35.
7. Soundarajan, N. (2001). Objectives, Outcomes, and Assessment Mechanisms for CS Programs. In Proceedings of the 31st ASEE/IEEE Frontiers in Education Conference, pp. T2A17-T2A22.