Assessing Student Learning in Computer Science – A Case Study

K.B. Lakshmanan†, Sandeep R. Mitra†, and T.M. Rao†
†Department of Computer Science
The College at Brockport, State University of New York
Brockport, NY 14420, USA

Abstract – Student learning outcomes are statements that describe what students are expected to know and be able to do by the time of graduation. In order to assess the extent to which an outcome is met, it is necessary to define an outcome in terms of measurable performance indicators. Rubrics allow collection of relevant data and their consistent interpretation. Using just one outcome as an example, this paper presents the approach used by the College at Brockport, State University of New York to define appropriate performance indicators, construct a curriculum map, develop holistic rubrics, collect data, evaluate, and use the findings for continuous program improvement.

Keywords: ABET, accreditation, assessment, outcomes, performance indicators, rubrics

1 Introduction

Accreditation is a review process that educational institutions undergo periodically for improvement of academic quality and for public accountability [5]. ABET is a recognized accreditor of college and university programs in applied science, computing, engineering, and technology [1]. The Computing Accreditation Commission (CAC) of ABET accredits computing programs, in particular Computer Science, Information Systems, and Information Technology. The Engineering Accreditation Commission (EAC) of ABET accredits Computer Engineering and Software Engineering programs.

The College at Brockport, State University of New York is a co-educational, state assisted, comprehensive liberal arts college in Western New York, offering bachelor’s, master’s, and post-master’s certificate programs. The Department of Computer Science is located in the School of Science and Mathematics. The Advanced Computing Track of the Computer Science Major was first accredited by CSAB [4] in 1994. With the integration of CSAB as a member society of ABET, the program is now accredited by the Computing Accreditation Commission (CAC) of ABET. The undergraduate program is a small one, graduating about 10 students per year. Moreover, since other unaccredited computing programs exist, students pursuing this rigorous accredited program are highly motivated academic achievers. The department does not offer a graduate program in computer science.

The Department of Computer Science at The College at Brockport, State University of New York has a long tradition of continuous program improvement through assessment of student learning outcomes. The department undertook a major revision of the outcomes and the assessment process following the release of the outcomes based assessment criteria by the Computing Accreditation Commission of ABET in 2006. In preparation for this revision, almost all faculty members of the department attended ABET-sponsored workshops on assessment. In this paper, we focus on just one outcome as an example, and discuss the approach we used to develop an assessment methodology that has been used successfully for ABET accreditation since then.

2 Outcomes

ABET requires each computing program seeking accreditation to develop a clear set of student learning outcomes, collect assessment data, determine the extent to which the outcomes are attained, and use the results of the evaluation to improve the program. In ABET terminology the student learning outcomes were previously known as program outcomes. But they are now known simply as student outcomes. The ABET web site has extensive material related to assessment [2].

Criterion 3 (Student Outcomes) of the ABET Criteria for Accrediting Computing Programs lists a set of characteristics any computing program must enable a student to attain. The list is augmented by additional characteristics under specific program criteria for computer science, information systems and information technology. Specifically, for computer science, the criteria list eleven characteristics. An institution may
adopt this list (a)-(k) itself as the set of student learning outcomes for its computer science program. However, at Brockport, we chose to develop a smaller set of eight outcomes that incorporated the essence of these eleven characteristics:

A. An ability to apply fundamental principles of computing and mathematics as appropriate to the discipline of computer science.
B. An ability to analyze a problem and model it as a computing system using appropriate methodologies, and to identify the computing requirements necessary to meet the desired needs.
C. An ability to design, implement and test a computing system, and to evaluate and compare the efficiencies of alternative solutions.
D. An ability to use current techniques, skills, and tools appropriate for immediate employment in computing technology development fields.
E. An ability to function effectively on teams to accomplish a common goal.
F. An ability to communicate effectively, both orally and in writing, using accepted standards of the profession.
G. An ability to analyze the social and human context of computing as it impacts individuals, organizations, and society, including ethical, legal, security, and global policy issues.
H. An ability to work and learn independently and an appreciation of the importance of continuing education and professional growth over the course of a lifetime.

The eight outcomes relate to fundamentals of the discipline, problem analysis and logical design, physical design and implementation, technical skills, team work, communication skills, ethics and other issues, and continuing education. Our outcomes are well-documented and widely disseminated through print literature and on the web to a variety of stakeholders such as potential and current students, alumni, and employers. ABET criteria require that there be a documented and effective process for the periodic review and revision of these outcomes to ensure that they are consistent with the published program educational objectives, which are broad statements that describe what graduates are expected to attain within a few years of graduation.

3 Performance Indicators

ABET criteria provides considerable leeway as to how the assessment of student learning outcomes is carried out. The criteria merely states that “effective assessment uses relevant direct, indirect, quantitative and qualitative measures as appropriate to the outcome being measured.” However, with growing maturity in the assessment process, programs are expected to place greater emphasis on direct assessment of student learning, which is based on student demonstrated work, rather than indirect assessment, which is based on student, alumni, or employer surveys. In this paper, we will just focus on Outcome A, as an example, and describe our assessment methodology. We will also focus only on the direct assessment of the outcome through evaluation of student work. As can be seen, Outcome A is a minor restatement of the student characteristic (a) in the published ABET criteria. Clearly, this outcome is a fairly broad statement encompassing fundamentals of computer science and mathematics appropriate to computer science. Thus, the assessment of the same requires breaking it down to a number of simple measurable aspects that allow one to determine the extent to which the outcome is met. In ABET literature these measurable aspects are known as performance indicators or performance criteria. Performance indicators are written with action verbs such as define, demonstrate, discriminate, evaluate, and interpret. Performance indicators also spell out specific subject contents [3]. In our case, we chose the following six performance indicators to define the Outcome A:

A1. Demonstrate an understanding of basic data structures and algorithms
A2. Demonstrate an understanding of a high-level object-oriented programming language and software design
A3. Demonstrate an understanding of number systems and digital logic
A4. Demonstrate an understanding of computer organization and architecture
A5. Demonstrate an understanding of analysis of algorithms
A6. Demonstrate an understanding of models of computation

The above list was chosen after careful evaluation of several performance indicators proposed by faculty members. Note that we have chosen two performance indicators to assess each of the software, hardware and theory aspects of computer science. Also the six performance indicators together cover the five foundational areas of computer science mentioned in the criteria: algorithms, data structures, software design, concepts of programming languages, and computer organization and architecture. Thus, collectively, the student performances measured by these indicators provide a reasonable basis for stating claims on the extent to which the Outcome A is attained. Similar sets of performance indicators have been developed for all other outcomes, but will not be presented or discussed in this paper.
4 Curriculum Map

Once the performance indicators have been identified, the next step for us was to determine for each performance indicator a set of courses or educational practices in the program through which the relevant skills, knowledge, and behaviors are acquired. If carried out carefully, this step in itself can help an institution identify some weaknesses in the program. For each performance indicator, the direct assessment data may be gathered from course material and student performances in classes through a variety of sources such as assignments, laboratory and written examinations, portfolios, programming projects, and oral examinations [6]. It is important to note that ABET expects summative assessment data, because the student learning outcomes specify what students are expected know and be able to do at the time of graduation. In other words, it is not necessary to gather data regarding each performance indicator at various stages in the curriculum. An institution may choose to develop formative assessment data for better control of their program, but ABET criteria do not require this. Further, some institutions may require a senior capstone project course. Assessment data for several performance indicators may be gathered from student demonstrated work in this course alone. But, in order to even out the workload on the faculty, it may be desirable to spread the assessment data collection over several courses. At Brockport, we collect assessment data for performance indicators in upper-level classes, as close to the time of graduation as possible. Table 1 presents performance indicators developed for the Outcome A, where the associated knowledge and skills are developed, where they are assessed, and how the assessment data is collected.

The following is a partial list of required courses in the computer science curriculum that appear in Table 1:

- CSC 203: Fundamentals of Computer Science I (CS1)
- CSC 205: Fundamentals of Computer Science II (CS2)
- CSC 303: Digital Logic and Computer Design
- CSC 311: Comp. Organization and Software Interface
- CSC 401: Programming Languages
- CSC 406: Algorithms and Data Structures
- CSC 411: Computer Architecture
- CSC 483: Theory of Computation

The complete curriculum and detailed descriptions of these courses may be obtained from the departmental web site at http://www.brockport.edu/cs/. Notice that if the knowledge or skill relevant to a particular performance indicator is developed over a sequence of courses, we collect assessment data in the last course in the sequence, that is, as close to the time of graduation as possible.

5 Rubrics

For further discussion let us focus on the Performance Indicator A2, which requires a student to demonstrate an understanding of a high-level object-oriented programming language and software design. Our goal in assessment is to categorize a student based on his or her performance relative to this performance indicator in one of four levels – beginning, developing, competent, and accomplished. But, exactly what aspects of programming languages and software design should be considered and exactly what data should be collected from course material? Further, how can we assure that different instructors at different times will characterize student performances consistently? To answer these questions, we have developed a holistic rubric associated with this performance indicator. Table 2 presents our rubric. Similar rubrics have been developed for all performance indicators associated with all student learning outcomes.

6 Data Collection

Until recently, we assessed all outcomes on a two-year cycle. We have recently moved to a three-year cycle. In any case, all instructors are informed of the performance indicators for which they need to collect data at the beginning of each academic year. Thus, an instructor for CSC 205, who is required to collect assessment data regarding the performance indicator A2 will be able to use the rubric for guidance and embed appropriate questions in course examinations or embed specific components in course projects and laboratory exercises that map to the knowledge and skills articulated in the rubric. The instructor will then be able to extract performance data and use them to categorize students in one of four levels indicated before.

At the end of the data collection period, which could be one semester or one academic year depending on the course, the instructor responsible for data collection is expected to provide the assessment coordinator the following:

- head counts of students in each of the four levels of performance
- mapping of course assignments, examinations, projects, etc., to rubric topics
- specific course assignments, projects, and examinations used in the analysis, and
- a reflective statement that could explain the distribution of students in the various performance levels

Table 3 is an example of head counts and the mapping of course material to rubric topics reported by an instructor.
7 Evaluation and Reporting

At the end of an academic year, the assessment coordinator of the department will tabulate all the head counts provided by various instructors and present them to the assessment committee for evaluation. Table 4 presents the compilation of student performances in regard to Outcome A during the 2006-08 assessment cycle. Chart 1 provides a visual presentation of the same data. The assessment committee is responsible for evaluating the data and arriving at suitable recommendations. We had set an a priori target level for performance as follows: an outcome is considered achieved if the percentage of the students rated competent or accomplished is 70% or higher. However, the assessment committee will still consider each and every one of the performance indicators individually and recommend actions for improvement even if the outcome is considered achieved. In this context, the reflective statements submitted by instructors along with assessment data plays a significant role.

8 Program Improvement

Table 5 provides an example of a typical recommendation for program improvement based on the assessment of Outcome A. Observe that the Outcome A is considered achieved as the percentage of students rated competent or accomplished is above 70%. However, close inspection of the individual performance indicators show that for both A3 and A4, the percentages of students rated competent or accomplished is below 70%, more so in the case of A4. Both performance indicators relate to hardware aspects of computing covered in the sequence of courses CSC 303, 311, and 411. The recommendation in this case took into consideration the recent changes in the curriculum and suggested only minor fine-tuning of course contents. In subsequent cycles of assessment, particular attention will be paid to recommendations of previous cycles.

9 Concluding Remarks

Using just one outcome as an example in this paper, we have presented the approach used by the College at Brockport, State University of New York to define appropriate performance indicators, construct a curriculum map, develop holistic rubrics, collect data, evaluate, and use the findings for continuous program improvement. Some assessment data have been modified to protect the privacy of instructors and to make this brief presentation more illustrative. The assessment methodology has been used for more than two cycles. With that experience, we have following observations to make.

- Student learning outcomes: We chose not to use the CAC list of student characteristics (a)-(k) as our outcomes, under the assumption that having fewer outcomes would lead to less work on the faculty. But, we now believe that there is no particular advantage to have developed our own set of student learning outcomes, since assessment data collection workload is really dependent on the total number of performance indicators we have.
- Performance indicators: When we first designed our assessment scheme, we tried to keep the performance indicators collectively exhaustive of all aspects of the outcome assessed. As a result, in some cases, we ended up with as many as ten performance indicators, leading to considerable workload for our faculty. We now believe that about six performance indicators per outcome are appropriate. Having fewer than six performance indicators may not provide sufficient information for program improvement.
- Holistic rubrics: Our rubrics show varying levels of detail as several faculty members participated in the development. We now believe that we need to bring some uniformity as these rubrics play a significant role in data collection and reducing faculty bias in scoring.

10 Acknowledgment

The authors acknowledge the contributions of other faculty members in the Department of Computer Science, The College at Brockport of the State University of New York in the development of the student learning outcomes, the performance indicators, and the holistic rubrics.

11 References

Outcome A. An ability to apply fundamental principles of computing and mathematics as appropriate to the discipline of computer science.

Table 1. Performance indicators, curriculum map, and assessment method

<table>
<thead>
<tr>
<th>Performance Indicators</th>
<th>Curriculum Map (Where Developed)</th>
<th>Where Assessed</th>
<th>Assessment Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1. Demonstrate an understanding of basic data structures and algorithms</td>
<td>CSC 203, 205, 406</td>
<td>CSC 406</td>
<td>Selected questions extracted from course examinations</td>
</tr>
<tr>
<td>A2. Demonstrate an understanding of a high-level object-oriented programming language and software design</td>
<td>CSC 203, 205</td>
<td>CSC 205</td>
<td>Selected questions extracted from course examinations; selected components of course projects</td>
</tr>
<tr>
<td>A3. Demonstrate an understanding of number systems and digital logic</td>
<td>CSC 303</td>
<td>CSC 303</td>
<td>Selected questions extracted from course examinations; selected components of course projects</td>
</tr>
<tr>
<td>A4. Demonstrate an understanding of computer organization and architecture</td>
<td>CSC 303, 311, 411</td>
<td>CSC 411</td>
<td>Selected questions extracted from course examinations</td>
</tr>
<tr>
<td>A5. Demonstrate an understanding of analysis of algorithms</td>
<td>CSC 205, 406</td>
<td>CSC 406</td>
<td>Selected questions extracted from course examinations</td>
</tr>
<tr>
<td>A6. Demonstrate an understanding of models of computation</td>
<td>CSC 401, 483</td>
<td>CSC 483</td>
<td>Selected questions extracted from course examinations</td>
</tr>
</tbody>
</table>

Table 2. Rubric for Performance Indicator A2

Performance Indicator A2. Demonstrate an understanding of a high-level object-oriented programming language and software design.

<table>
<thead>
<tr>
<th>Beginning (Fails to meet)</th>
<th>Developing (Close to Meeting)</th>
<th>Competent (Meets, Satisfactory)</th>
<th>Accomplished (Exceeds, Exemplary)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cannot determine appropriate data representation or design algorithms. Cannot implement solutions using simple control structures in a high-level language. Cannot analyze problems, identify classes, assign responsibilities, or design solutions. Cannot create analysis/design artifacts such as use cases and class diagrams.</td>
<td>Can determine a data representation and design an algorithm for simple problems. Can implement solutions using basic control structures (if, switch, and looping) to solve simple problems in a high-level language. Can analyze problems, identify classes, assign responsibilities, and design solutions for simple cases. Can create some analysis/design artifacts such as use cases and class diagrams.</td>
<td>Can determine an appropriate data representation and design correct algorithms. Can implement solutions using control structures (if, switch, looping, and recursion) to solve reasonable sized problems in a high-level language. Can analyze problems, identify classes, assign responsibilities, and design solutions for reasonably complex cases. Can create analysis/design artifacts such as use cases and class diagrams for reasonably complex cases.</td>
<td>Can determine the most appropriate data representation and design efficient algorithms. Can implement solutions using control structures (if, switch, looping, and recursion) to solve large problems in a high-level language. Can analyze problems, identify classes, assign responsibilities, and design solutions for fairly complex cases. Can create analysis/design artifacts such as use cases and class diagrams for fairly complex cases.</td>
</tr>
</tbody>
</table>
Performance Indicator A2. Demonstrate an understanding of a high-level object-oriented programming language and software design.

<table>
<thead>
<tr>
<th>Rubric Topic</th>
<th>Course Topic</th>
<th>Assignment, Exam, and Projects</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data representation and design of algorithms</td>
<td>Collection classes: stacks, queues, linked lists, binary trees, etc.</td>
<td>Exam II: Question 2; Project 2</td>
</tr>
<tr>
<td>Implementation in a high-level language</td>
<td>Java implementation, manipulation of data structures, wrapper classes, recursion</td>
<td>Exam I: Question 3, 4; Laboratory Assignments1, 2</td>
</tr>
<tr>
<td>Problem analysis, identification of classes and assignment of responsibilities</td>
<td>Abstract classes, interfaces</td>
<td>Final examination Questions 7 and 8; Project 3</td>
</tr>
<tr>
<td>Creation of analysis/design artifacts</td>
<td>Use cases/CRC cards</td>
<td>Exam II: Question 6; Laboratory Assignment 4; Project 3</td>
</tr>
</tbody>
</table>

Table 4. Compilation of assessment data for Outcome A during 2006-08 cycle

Outcome A. An ability to apply fundamental principles of computing and mathematics as appropriate to the discipline of computer science.

<table>
<thead>
<tr>
<th>Performance Indicators</th>
<th>Where/When Assessed</th>
<th>Head Count of Students Rated</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Beginning</td>
<td>Developing</td>
</tr>
<tr>
<td>A1. Demonstrate an understanding of basic data structures and algorithms</td>
<td>CSC 406 Spring 2007</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>A2. Demonstrate an understanding of a high-level object-oriented programming language and software design</td>
<td>CSC 205 2006-07</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>A3. Demonstrate an understanding of number systems and digital logic</td>
<td>CSC 303 2006-07</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>A4. Demonstrate an understanding of computer organization and architecture</td>
<td>CSC 411 Spring 2007</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>A5. Demonstrate an understanding of analysis of algorithms</td>
<td>CSC 406 Spring 2007</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>A6. Demonstrate an understanding of models of computation</td>
<td>CSC 483 Spring 2007</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td></td>
<td>8</td>
<td>14</td>
</tr>
</tbody>
</table>

*Data presented here are for illustrative purposes only and are not actual
Outcome A. An ability to apply fundamental principles of computing and mathematics as appropriate to the discipline of computer science.

Percentage of students rated competent or accomplished: 52/74= 70.27%

Status of the outcome: Achieved. (The percentage of students rated competent or accomplished is above 70%)

Recommendation(s): Performances under indicators A3 and A4 are of concern. The percentages of students rated competent or accomplished are below 70% for both. We have revised the contents of CSC 311: Computer Organization and Software Interface effective Fall 2006. We have moved from Intel to MIPS architecture. We also require programming in “C” language in this course now. Hence, no immediate course correction is recommended. The instructors teaching hardware-related courses (CSC 303, 311 and 411) should continue to monitor the situation, with small fine-tuning of contents, if found necessary.

Time frame for implementation: Starting from the next offerings of CSC 303, 311, and 411.