

Direct Assessment Methodology for the IIT Computer Science Department - August 21, 2008

Introduction

Currently, our department has numerous indirect assessment measures. These include course assessment surveys by course manager, faculty surveys, student surveys, and alumni surveys. We have some direct evaluation in that we study one course each semester in detail and we look carefully at each part of the course. However, we will augment our existing assessment measures with additional direct evaluation. The motivation for this is to improve upon our existing assessment measures and to work toward more concrete statistics that can help us continue to improve our curriculum. This document describes our direct measurement methodology.

Methodology

To obtain these direct measures we will use the following process.

Step 1: Identify Program Outcomes

We begin with our existing program outcomes. These are listed on our Departmental Web site at <http://www.iit.edu/csl/cs/about/mission.shtml> and they are listed here for convenience.

- a. An ability to apply knowledge of computing and mathematics appropriate to the discipline
- b. An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution
- c. An ability to design, implement and evaluate a computer-based system, process, component, or program to meet desired needs
- d. An ability to function effectively on teams to accomplish a common goal
- e. An understanding of professional, ethical, legal, security, and social issues and responsibilities
- f. An ability to communicate effectively with a range of audiences
- g. An ability to analyze the local and global impact of computing on individuals, organizations and society
- h. Recognition of the need for, and an ability to engage in, continuing professional development
- i. An ability to use current techniques, skills, and tools necessary for computing practices.
- j. An ability to apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehension of the tradeoffs involved in design choices
- k. An ability to apply design and development principles in the construction of software systems of varying complexity.
- l. Be prepared to enter a top-ranked graduate program in Computer Science.

Step 2: Map courses to outcomes

The mapping of courses to outcomes is given below. This is directly copied from our self-study assessment with the following addition. We note that although CS 425 Database Systems, CS 422 Data Mining, and CS 429 Information Retrieval are not listed in our self-study since they are not required courses. Because the vast majority of undergraduates take these courses as electives, we have added them to our course mapping. Additionally, CS 422 and CS 429 have been assessed rigorously for the past several years due to NSF grants, so we will be able to leverage work done on these courses by including them

	CS100	CS115 CS116 CS201	CS330	CS331	CS350	CS351	CS430	CS440	CS450	CS485	CS487	CS422	CS425	CS429	I PRO
a	X	X	X	X	X	X	X	X	X			X	X	X	X
b	X	X	X	X	X	X	X				X	X	X	X	X
c	X	X	X	X	X	X	X	X			X	X	X	X	X
d	X									X	X				X
e	X									X	X				X
f	X				X		X			X	X	X	X	X	X
g	X									X					X
h							X	X	X		X	X	X	X	X
i		X		X	X	X	X	X			X	X	X	X	
j		X	X	X	X	X	X	X	X						
k		X		X							X	X	X	X	
l							X	X	X		X	X	X	X	

CS100 Introduction to the Profession	CS115/CS116 Object-Oriented Programming I & II
CS201 Accelerated Introduction to CS	CS330 Discrete Structures
CS331 Data Structures and Algorithms	CS350 Computer Organization
CS351 Systems Programming	CS430 Introduction to Algorithms
CS440 Programming Languages and Trans.	CS450 Introduction to Operating Systems
CS485 Computers and Society	CS487 Software Engineering
CS422 Introduction to Data Mining	CS425 Database Organization
CS429 Introduction to Info. Retrieval Sys.	I PRO497 Interprofessional Projects(2)

Step 3: Select courses to focus on for the year

Before each academic year begins, our undergraduate studies committee will choose a representative set of courses to use to drive our direct evaluations. Using every course would simply overwhelm our committee with assessment activities. We will choose a set of courses that covers all of the outcomes. For this academic year, we have selected the following courses. The table below shows that these courses cover all of our outcomes.

Fall 2008
 CS 201 Accelerated Introduction to CS
 CS 331 Data Structures and Algorithms
 CS 485 Computers and Society

CS 425 Database Organization
CS 429 Introduction to Information Retrieval Systems

Spring 2009

CS 201 Accelerated Introduction to CS
CS 331 Data Structures and Algorithms
CS 485 Computers and Society
CS 425 Database Organization
CS 422 Introduction to Data Mining

We anticipate that future semesters we may vary this list of courses in order to make sure that our assessment methodology touches on all outcomes. In all cases, we will ensure that at least five courses will be used. We select a different subset of courses each year (mostly required, some elective) that covers (or almost covers) the program outcomes. The list of courses to be studied for a given semester will be identified as early as possible before a semester begins so that instructors of these courses will be aware of the need to submit additional information for their course. We have notified the faculty whom will be teaching the courses listed above.

Step 4: Map Course Outcomes to Direct Assessment

Starting with the outcomes defined for a course, identify direct assessment measures that will be used to assess each outcome.

For each course, course outcomes have been identified, listed on the course overview for each course (<http://www.cs.iit.edu/~abet/CourseOverviews.html>), and are part of the syllabus/Web site for each course each semester. As an example, CS201 outcomes (see <http://www.cs.iit.edu/~cs201/>) are given below:

Students who succeed in the course will be able to:

1. Analyze and explain the behavior of simple programs involving the following fundamental programming constructs: assignment, I/O (including file I/O), selection, iteration, methods
2. Write a program that uses each of the following fundamental programming constructs: assignment, I/O (including file I/O), selection, iteration, methods
3. Break a problem into logical pieces that can be solved (programmed) independently.
4. Develop, and analyze, algorithms for solving simple problems.
5. Use a suitable programming language, and development environment, to implement, test, and debug algorithms for solving simple problems.
6. Write programs that use each of the following data structures (and describe how they are represented in memory): strings, arrays
7. Explain the basics of the concept of recursion.
8. Write, test, and debug simple recursive functions and procedures.
9. Explain and apply object-oriented design and testing involving the following concepts: data abstraction, encapsulation, information hiding, inheritance, polymorphism

10. Use a development environment to design, code, test, and debug simple programs, including multi-file source projects, in an object-oriented programming language.
11. Implement basic error handling
12. Solve problems by creating and using sequential search, binary search, and quadratic sorting algorithms (selection, insertion)
13. Determine the time complexity of simple algorithms.
14. Apply appropriate problem-solving strategies
15. Use APIs (Application Programmer Interfaces) and design/program APIs

The Course Manager for each course will identify an assessment measure used for each course outcome. For example, to test “Write, test, and debug simple recursive functions and procedures.”; performance on test #3 might be used as the key assessment measure.

All assessment measures will use a graded test, graded homework assignment, or a graded presentation/project. If a course manager wishes to drill down further into just a given question on an exam, this is acceptable, but we are not requiring this at this time. The grade for an entire course is simply too broad to accurately assess a given outcome. Assessment measures will be submitted to the undergraduate studies committee prior to the semester for the review of the committee. We will require assessment measures for at least half of the published outcomes for a given course. We realize that one test may cover many assessment measures, but we will strongly encourage faculty to use different graded materials to cover different assessment measures.

Faculty will submit a table of the form below that maps assessment tools to course outcomes. Here is an example for CS201

		Course Outcomes															
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
Assessment Tools	Lab 1					X											
	Exam 1	X	X														
	Lab 8					X				X							
	Exam 2						X			X	X						
	Lab 11											X					
	Lab 12												X	X			
	Project			X	X	X										X	X
	Final				X			X	X								

Step 5: Map Direct Assessments to Program Outcomes

In order to compute objective measures for program outcomes, the items assessed will be tied to specific program outcomes. This may ultimately reduce the number of items assessed because each course may only need to assess one or two program outcomes. The focus of the assessments may shift more toward junior and senior level courses so that we can ensure that the program outcomes occur prior to the time a student graduates. For example, it may not affect a

program outcome if a Freshman is unable to perform well on a recursion question on an exam, but it would be far more relevant if a Junior or Senior is unable to perform the same task.

Step 6: Agree on Goals for Assessment Measures

In the future, our goals may well be tailored to each course. Since we are just starting our direct assessment work, we will use a sliding scale based on the seniority of our students in a given course. The goals are:

- For all CS 1xx or 2xx courses all assessments will have a goal of an A, B, or C for 80% of the students. We note that this rule may well be adjusted over time, but it is a starting point.
- For all CS 3xx courses, assessments will have a goal of an A, B, or C for 75% of the students.
- For all CS 4xx courses, assessments will have a goal of an A, B or C for 70% of the students.

We use this sliding scale as it enables us to work toward ensuring that our first and second-year students master the introductory material. Computer Science is one field where, if mastery of the fundamentals does not happen in the early years, it is extremely hard to recover. Hence, we will work to ensure that more students master the fundamentals in their early years.

Step 7: Collect Data

At the end of the semester, the course instructor will submit the grade distribution for each direct assessment measure to the undergraduate studies committee. We note that we have started a small software application to allow instructors to submit their assessments on-line. In addition to the grade distribution, instructors will submit an electronic version of the test, homework assignment, or project description used for the assessment.

Step 8: Analyze Data

The undergraduate studies committee will analyze the data that is collected at the end of each semester and will adjust any goals or assessment measures prior to the start of a new semester. Clearly, a report of which assessment measures met the goals and which ones did not meet the goals will be considered as part of this review.

Summary

We have described a methodology that sounds straightforward, but as we use it, we are certain to encounter difficulties. We will clearly summarize the efforts of using this process shortly after the Fall 2008 semester and will develop a yearly summary with results after the Spring 2009 semester. Our summary will include results from doing a full year of direct assessment and we will recommend any needed improvements to our process.