# Infrastructure for continuous assessment of retained relevant knowledge

Travis Doom, Kathleen Timmerman, Michael Raymer
Wright State University, Dayton, OH 45431-0001
Email: travis.doom@wright.edu

**Abstract**
Over the past decade, engineering programs nation-wide have devoted significant effort towards measuring educational objectives in the style recommend by ABET Engineering Curriculum 2000 to monitor and improve program effectiveness. In many cases the collection and interpretation of this data has taken place in a labor intensive *ad hoc* fashion which limits utility of the collected data to drive curricular or pedagogic improvement. Herein, we present a data collection infrastructure designed to measure success in retaining specific knowledge area topics deemed critical by each engineering discipline's professional society. Where possible, assessment points are deployed at that start of courses that use knowledge topics developed in prerequisite core courses. This infrastructure allows evaluation of retention of expertise, allows assessment of differences in outcomes between learning pathways, and is less subject to instructor, course format, or other bias. Equally important, this infrastructure requires minimal resources post-deployment yet collects critical program data while also providing immediate feedback to students and course instructors regarding the preparation of students as they enter courses that allows for focused review and reinforcement of knowledge areas not retained due to variation in preparation.

## Introduction

Institutions of higher learning world-wide have embraced continuous improvement models to measure and increase effectiveness of student learning. This is particularly true in undergraduate engineering programs as the ABET 2000 criteria prompted engineering departments to adopt continuous program outcome assessment to satisfy basic level accreditation criteria. All ABET accredited engineering programs are now expected to have some model for continuous program outcome assessment in place. The key to continuous improvement is an effective assessment program. Without a solid measure of student learning, a cycle of improvement is driven by the variations and vagaries of the data and is less likely to result in meaningful positive change.

Learning is multidimensional and requires multiple methods of collection in order to produce meaningful data. Direct methods of assessment measure student performance against some rubric of success. Indirect methods of assessment more often measure the student's (or observer's) perception of attainment. While both methods of assessment have their place, direct measures of assessment have been used for decades to provide a means for quality assurance. Historically, direct examinations such as the ACT and SAT have been used to measure the educational achievement of high-school students applying to college. Similarly, examinations such as the GRE, subject GRE, and Fundamentals of Engineering (FE) examination have been used to measure student educational achievement in University and to partially gauge professional competency.

Examinations of this sort provide validation against a set of external criteria that demonstrate that the *retained knowledge* of each student is *relevant* to the current national standard. Unfortunately, end-of-program examinations of this sort make poor tools for continuous program improvement. It is difficult, if not impossible, to provide a linkage between overall examination performance and specific actions or pedagogies employed in the educational process that led to greater or lesser success.

Continuous periodic direct measurements provide the best opportunity for measuring the performance effects of specific changes to programs, courses, and pedagogies. However, such data collection efforts are practically limited due to the sometimes massive effort required from administration, faculty, and students.

We propose here an infrastructure to assess program effectiveness with the following goals:

1. The assessment provides continuous periodic direct measurements of retained relevant knowledge.
2. The assessment outcome is immediately valuable to the assessment participants (students and faculty) as well as the continuous improvement of the program.
3. The assessment is not unduly burdensome.

## Assessment knowledge topics

The goal of assessment is to provide data to measure (or illustrate a need for) improvement. The definition of the assessment standards then set a target goal towards which a program continuously strives to better meet. Although program objectives differ significantly among institutions, certain knowledge and skills are expected of graduates of engineering programs. We believe that the standard towards which programs should strive in Engineering is best communicated not only by the accreditation agencies but also by the appropriate discipline-specific international professional society. These societies maintain and regularly update the themes, knowledge areas, and professional practices expected of those entering their discipline.

For example, in computer science, the Joint Task Force on Computing Curricula between the Association for Computing Machinery (ACM) and IEEE-Computer Society provides regularly updated standards in curriculum, most recently in the volume Computer Science Curricula 2013 (CS2013) [1]. The CS2013 Body of Knowledge organizes the expectations of Computing graduates into 18 Knowledge Areas (KA) which are created, revised, and removed as the discipline changes over time (Figure 1, below). Each of these KAs is further specified as a set of Knowledge Units (Figure 2, below) each of which specifies a set of Knowledge Topics (Figure 3, below) expected at the time of graduation.

| | |
|---|---|
| AL | Algorithms and Complexity |
| AR | Architecture and Organization |
| CN | Computational Science |
| DS | Discrete Structures |
| GV | Graphics and Visualization |
| HC | Human Computer Interaction |
| IAS | Information Assurance |
| IM | Information Management |
| IS | Intelligent Systems |

| | |
|---|---|
| NC | Networking and Communication |
| OS | Operating Systems |
| PD | Parallel and Distributed Computing |
| PL | Programming Languages |
| SDF | Software Development Fundamentals |
| SE | Software Engineering |
| SF | System Fundamentals |
| SP | Social and Professional Practice |

**Figure 1: CS2013 Knowledge Areas [CS2013]**

| **Algorithms and Complexity (AL)** |
|---|
| AL/Basic Analysis |
| AL/Algorithmic Strategies |
| AL/Fundamental Data Structures and Algorithms |
| AL/Basic Automata Computability and Complexity |

**Figure 2: Sample Knowledge Units in the Algorithms and Complexity Knowledge Area [CS2013]**

**AL/Fundamental Data Structures and Algorithms**

- Simple numerical algorithms, such as computing the average of a list of numbers, finding the min, max, and mode in a list, approximating the square root of a number, or finding the greatest common divisor
- Sequential and binary search algorithms
- Worst case quadratic sorting algorithms (selection, insertion)
- Worst or average case O(N log N) sorting algorithms (quicksort, heapsort, mergesort)
- Hash tables, including strategies for avoiding and resolving collisions
- Binary search trees
- Common operations on binary search trees such as select min, max, insert, delete, iterate over tree
- Graphs and graph algorithms
- Representations of graphs (e.g., adjacency list, adjacency matrix)
- Depth- and breadth-first traversals
- Graphs and graph algorithms
- Shortest-path algorithms (Dijkstra's and Floyd's algorithms)
- Minimum spanning tree (Prim's and Kruskal's algorithms)
- Pattern matching and string/text algorithms (e.g., substring matching, regular expression matching, longest common subsequence algorithms)

**Figure 3: Sample Knowledge Topics in the Algorithms and Complexity: Fundamental Data Structures and Algorithms Knowledge Unit [CS2013]**

For computer science programs, CS2013 can serve as a "gold standard" for contemporary computing education. The professional societies of other engineering disciplines provide similar international curricular standards along with, in many cases, examinations which new graduates are expected to pass in order to be fully qualified to work in the discipline. In recognition that program objectives differ, CS2013 identifies topics as being either core tier-1 (required knowledge for every students in every program), core tier-2 (generally essential topic for which the vast majority should be covered but which may differ by student or program), or elective. CS2013 makes the categorizations by the process of "widespread consensus for inclusion" and further notes that "at least a preliminary treatment of most of these [core tier-1] topics typically comes in the first two years". The explicitly stated coverage target for core tier-2 topics is "90-100% for every student, with 80% [as measured in lecture hours] considered as a minimum".

CS2013 contains 163 lecture hours of Core Tier-1 material and 142 lecture hours of Core Tier-2 material. For comparison, assume that semester-based course has roughly 45 lecture contact hours. The CS2013 then consists of the equivalent of roughly four 3-credit hour semester courses worth of Tier-1 material and roughly three 3-credit hour semester courses worth of Tier-2 material, spread throughout the entire curriculum.

While acknowledging that every program has differing educational objectives, use of professional society standards provides metrics which can gauge the success of the program against a national model. Such metrics suggest an infrastructure for direct assessment that allows comparison against discipline-wide expectations and to allow reflection on the need, causes, and appropriateness of any major deviations from the widespread consensus proposed by the discipline's professional society.

**Continuous periodic direct measurements of retained relevant knowledge**

We have worked with our program faculty to produce a mapping of which CS2013 Knowledge Topics are prerequisite to or developed in each of the core/mandatory courses in our computer science program. Our initial assessment framework is limited to mandatory "core" courses. Students will gain additional experience in many core knowledge topics in their elective coursework. However, the topics and amount of coverage will necessarily vary based upon the selected electives. Thus, initial observations are limited only to core/mandatory courses.

For each course, the faculty has indicated what knowledge topics are developed or assumed (prerequisite) in the semester-based course offerings. We propose that an appropriate method to assess relevant retained knowledge is to perform a direct assessment of each knowledge topic not only in the course that develops that knowledge, but when possible, at the beginning of a subsequent course (or courses) that utilizes and builds on the topic.

Summative grading rubrics are, when possible, deployed at the start of the next course in the core course sequence (Figure 4, below). These assessment points allow better evaluation of the retention of expertise as measured prerequisite knowledge coming into each course. Assessment of prerequisite knowledge also allows assessment of differences among learning pathways, and are less subject to instructor- or course- related bias.

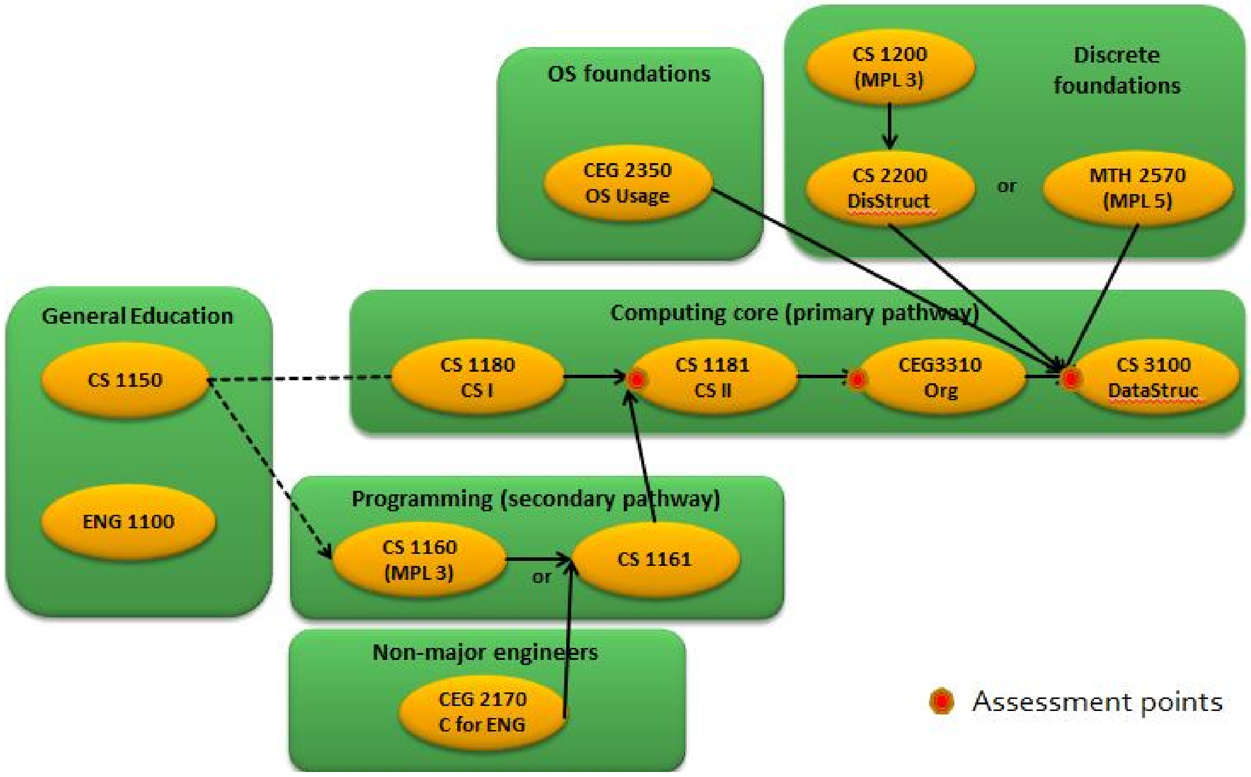

**Figure 4: Assessment points in the first two years of the core curriculum. Additional assessment points exist in advanced core courses including Software Engineering, Operating Systems, and the Capstone Design Sequence. These assessment points are not illustrated. The assessments point for Capstone Design takes places at the end of that course sequence as there is no subsequent required course in the program.**

## Immediate value to participants

We propose that the assessments be required of all students entering every core course. Furthermore, we propose that the results not affect their course grade. As this effort is not associated with a course grade there is no need to proctor or use valuable classroom time on the assessment. The assessments are simply delivered as on-line standardized quizzes (Figure 5, below). We have found that restricting access to on-line classroom materials until the assessment quiz for the course is completed gives complete class participation in the assessment.

In our experience, students are very open about their level of mastery of concepts assessed in not-for-credit surveys of prerequisite knowledge. The feedback from these assessments is immediately useful to students as it calls up old ideas (helping them to be ready for new related knowledge). This immediate feedback can also reduce anxiety regarding the sufficiency of their mastery of assumed prerequisite knowledge or identify specific areas where they can be coached to better prepare for succeed in a new course. As students find the feedback valuable to them personally, they are more likely to give significant and frank effort in the assessment process.

As a direct assessment of student preparedness, this data should be less biased than indirect assessments that ask students their opinion of their ability. Differences in self-expectation that may exist among students due to experience or demographic are removed. Thus, students/faculty get a more accurate measure of how well each student is prepared.

Consider the following segment of code in a java-like programming language. Assume that there are no syntax errors.

```
  int[] m = {2,3,4,5,6};
  int n = 0;
  int x = 0;
  for (int val = 0; val < m.length; val++)
  {
    if (val % 2 == 1)
    {
      n = n + val;
      x = x + 1;
     } // end-if
   } // end-for
```

What is the most likely use for the code segment above?

A) Calculating the total sum of the values held in array m.
B) Calculating the average of the values held in array m.
C) Calculating the number of even values held in array m.
D) Calculating the average of odd values held in array m.
E) Calculating the number of values held in array m.

**Figure 5: Sample Question to assess the Knowledge Topic AL/Fundamental Data Structures: Numerical Algorithms. This topic is developed in Computer Science I and built upon in Computer Science II. Thus, this question would appear in the assessment of prerequisite knowledge at the start of Computer Science II.**

Equally important, the results of these perquisite surveys can be made immediately available to the faculty teaching the course in which the examination is held. If the faculty member sees weakness in prerequisite knowledge then they are able act to help address the problem immediately. The assessment can help identify individual students that might require additional help as well as identify potential systemic deficiencies introduced by previous poor instruction, variation in schedule due to weather/emergency, differing pathways for preparation (such as transfer courses), or the like. Based upon assessed performance, the faculty can tailor any necessary review of prerequisite topics appropriately to the needs of each term's student preparation.

## Assessment overhead and administrative burden

Ease of assessment delivery allows the potential direct assessment of every student every term in every core course. As these assessments are delivered as on-line standardized examination, they require very little class time or faculty effort to administer. Each knowledge topic is mapped to relevant ABET engineering criteria 2000 CAC/EAC a-k criteria listed in ABET's Criterion 3: Student Outcomes [2,3]. This allows the data to be used by class or longitudinally by student to assess continuous improvement of the program overall against ABET Engineering criteria in a well-defined and straight-forward manner.

The most significant administrative burden is in the initial development, validation, and continuous improvement of the assessment questions. The initial burden of assessment development requires significant faculty involvement and may require multiple years of effort to construct assessment questions for every core course. The measurements for a knowledge area may be skewed by a set of poor assessment questions, thus continuous improvement of the questions in parallel with the improvement of curriculum remains an ongoing administrative effort.

## Conclusion

This assessment infrastructure allows for an assessment of retained knowledge, topic by topic, for each individual student, course, and term. When collected with appropriate demographic information, these assessments allow the differential measurements of knowledge retention under any number of pedagogical variables. The success of new instructional styles, laboratory techniques, or technologies for developing knowledge can be assessed against different approaches.

Every contemporary engineering discipline has a professional society that helps identify the core concepts of the discipline. Indeed, most engineering disciplines have standardized examinations of some sort that are used to demonstrate student proficiency for licensure or graduate studies. Questions of this sort can be used at the start of core courses or time points to assess student knowledge of prerequisite topics developed earlier in any program of study. These assessments can be delivered as online questions to minimize cost and maximize participation. When collected with appropriate demographic information, this rich set of data can guide program improvement more effectively than many existing program assessment plans. Although we present this infrastructure in the context of Computer Science, we believe that the approach can be applied to implement an infrastructure for effective assessment program for any engineering discipline.

## Bibliography

[1] ACM/IEEE-CS Joint Task Force. Computer Science Curricula 2013 (CS2013). http://ai.stanford.edu/users-/sahami/CS2013/. Strawman draft, Feb 2012.
[2] ABET. Computing Accreditation Commission (CAC) Criteria. http://www.abet.org/accreditation-criteria-policies-documents/
[3] ABET. Engineering Accreditation Commission (EAC) Criteria. http://www.abet.org/accreditation-criteria-policies-documents/