William Paterson University
College of Science and Health - Department of Computer Science

Fall 2013 – Spring 2015 Assessment Cycle
Analysis of the Program's Student Outcome Assessment Data

Program's Student Outcome: S6:

Demonstrate abilities to select appropriate data structures and to design algorithm to solve problems.

ABET's Related Student Outcomes: (b), (c), (i).

Curriculum Committee Subgroup: **Data Structures and Algorithms**

Members: John Najarian (Chair), Erh-Wen Hu, Gilbert Ndjatou

Date: February 22, 2016

Updated On: September 29, 2016

A. Analysis of the Assessment Data

For the assessment period Fall 2013 to Spring 2015, this student outcome was assessed in the CS 3420 Data Structures. First we observe that more than 50% of the students demonstrate More than Adequate Ability (both in the total statistic and in 4 of the 5 classes, the once exception being 46.7% (close to 50%)). So the majority is successful beyond adequate. On the lower end tail, the below adequate category, we have (in sorted order) 0%, 0%, 12.5%, 20%, 22%, which averages near 11%. That also indicates roughly a tenth fall behind performance-wise, a reasonable number in many respects considering the present cohort; these weakest students reach a stage of Computer Science where their prior problems become magnified and more evident.

B. Suggestions for Improvement

- More coordination of sections with tutoring support. To address the weakest students, perhaps some credit/weight can be assigned to students upon their receiving remedial help.
- We may need to review textbooks to find one that simultaneously meets all the criteria cited (good pedagogical diagrams/illustrations, projects which test students at several levels in each chapter, modern coding-standards (no more classics), elegant C++ OOP code (with well-tested code that actually works), not too long-winded in verbose exposition, concise examples, and sufficiently representative set of data structures (no reduced set or watered-down coverage).
- To contrast time-complexity of data structures, the use of bigger data sets is one way of meeting S6's selectivity criterion. Whether they work with real data or random-number syntheses, the net result is that students see the efficiency and efficacy of progressively faster data structures.

C. Improvement Implemented

The above suggestions have been implemented in Fall 2016.

D. List all the "performance level/frequency/percentage" tables and their sources.

a. Faculty Course Assessment Report: CS3420, Fall 2013

Data Collected: Each student's level of performance on the lab assignments and the questions on the tests and the final exam.

Method of Collection: lab assignments, questions on tests and the final exam that assess students ability to implement, manipulate and use linked list, stacks, queues, and trees to solve problems.

Performance Levels	Frequency	Percentage
Some Ability		
Adequate Ability	8	47.06%
More than Adequate Ability	2	11.76 %
High Ability	7	41.18%

Observations: This is a particularly strong class with hard working students who complete their homework and lab assignments. Nine out of the seventeen students did very well with all aspects of the course. The performance of the rest of the class (8 students) was adequate; however, most of them were weak in developing programming applications.

---

b. Faculty Course Assessment Report : CS3420, Spring 2014

Data Collected: Each student's level of performance on the lab assignments and the questions on the tests and the final exam.

Method of Collection: lab assignments, questions on tests and the final exam that assess students ability to implement, manipulate and use linked list, stacks, queues, and trees to solve problems.

Performance Levels	Frequency	Percentage
Some Ability	4	22%
Adequate Ability	3	16 %
More than Adequate Ability	8	44 %
High Ability	3	18 %

Observations: The stronger students could handle templates but preferred (were more comfortable with) classes and objects. The weakest students could read classes and objects but had difficulties implementing with them. To support this diversity, programs were developed and presented using three levels, specifically, struct/typedef, class/object, and template.

c. Faculty Course Assessment Report: CS3420, Fall 2014

Data Collected: Each student's level of performance on the lab assignments and the questions on the tests and the final exam.

Method of Collection: lab assignments, questions on tests and the final exam that assess students ability to implement, manipulate and use linked list, stacks, queues, and trees to solve problems.
.

Performance Levels	Frequency	Percentage
Some Ability	0	0.0%
Adequate Ability	3	21.43%
More than Adequate Ability	6	42.86 %
High Ability	5	35.71%

Observations: This is a particularly strong class with hard working students who complete their homework and lab assignments. Eleven out of the fourteen students did very well with all aspects of the course. The performance of the rest of the class (3 students) was adequate. Three students are weak in developing programming applications.

d. Faculty Course Assessment Report: CS3420, Spring 2015

Data Collected: Each student's level of performance on the lab assignments and the questions on the tests and the final exam.

Method of Collection: lab assignments, questions on tests and the final exam that assess students ability to implement, manipulate and use linked list, stacks, queues, and trees to solve problems.

Performance Levels	Frequency	Percentage
Some Ability	3	20.0%
Adequate Ability	5	33.3%
More than Adequate Ability	5	33.3%
High Ability	2	13.4%

Observations:

- Spent first three weeks on reviewing all OOP topics taught in the previous class and on dynamic memory, pointers, and arrays (static and dynamic; one and two dimensions) as most students had trouble with these topics.
- Some students had trouble handing in earlier programming assignments in a timely manner. To address the problem, the instructor organized students into small discussion groups and spent extra after-class hours with the groups turn the situation around.
- Responding to the requests from students, the instructor occasionally allowed individual students to present their code (often not working) for comments from the entire class. All students seemed to participated in the discussion went through critical thinking process. Most would tell me that it really helped them learn and request more such opportunities.
- Most students work very hard as evidenced by the extra hours many had chosen to stay in the classrooms after the class was over. Most students hand in nearly all programming assignments.

---

e. Faculty Course Assessment Report: CS3420, Spring 2015 (additional section)

Data Collected: Each student's level of performance on the lab assignments and the questions on the tests and the final exam.

Method of Collection: lab assignments, questions on tests and the final exam that assess students ability to implement, manipulate and use linked list, stacks, queues, and trees to solve problems.

Performance Levels	Frequency	Percentage
Some Ability	1	12.5%
Adequate Ability	2	25.0%
More than Adequate Ability	3	37.5%
High Ability	2	25.0%

Observations: The distribution is relatively bell-shaped but one student did not do sufficient homework, missed classes, and fell behind. Despite the one exception, we did well with Patil's strong pedagogy, my repeated recitations, and jointly doing most of the book's review questions with students in class in a lively partially competitive group review.

More than one third of the class did 10 or more of the 16 projects. That actually is very good since my projects have multiple parts and multiple programming exercises. However, it was disappointing to see two good students lose one whole letter grade due to incomplete homework.
Worse yet, one of them had the best exam grades; that is tragic. It makes no sense, as I kept encouraging that student to complete the homeworks.

I do not drop the lowest grade but if a student does poorly on one exam, I will talk to them, try to identify their problem, give some positive guidance but with somber note, and send them to the tutor. Their best grades were on the final exam. Their weakest point was with Linked Lists; the dynamic nature of the data structure caused them difficulties. Even with repeated requests to study harder and code-design/walk-throughs in class, they just did not practice enough or study enough. After the Linked Lists test, students recognized that they needed to study harder and practice more. My warnings were not sufficient; they needed to see a grade slump to appreciate the more complex programming context and develop dynamic structural reasoning, insight, and just general stronger

study habits as the subject matter advanced further. It was relieving to see they make the transition and progress beyond.

Students were more comfortable with classes and objects than templates. Many confronted their fears and were able to read templates. I chopped out any examples or usage of struct/typedef, strictly adhering to class/objects. We did a thorough review of classes and OOP at the start, so as to avoid struct and typedef.

Varsha Patil's text "*Data Structures using C++*" published by Oxford University Press in 2012 is a fine replacement for Nyhoff. The focus on pedagogy, good combination of easy review questions followed by homework questions, and numerous diagrams/illustrations/charts makes it a paragon, just what the students need. Given that other text's cost over a hundred and some two hundred, this text is very affordable, an important issue in light of the economic downturns many students are experiencing before they get jobs. Patil provides a fine exposition; some typos in the first printing are being corrected but students and I actually have fun catching and fixing them. For code, it is an excellent way of seeing who is alert and really understands. Likewise, I always help students debug their code by identifying problems and hinting or coaxing their reasoning along. We do recitation more than lecture and code walkthroughs are fun with modern C++ (as opposed to Nyhoff's typedef/struct approach). Drozdek and Weiss are also modern fine textbooks but students appreciate more the Patil support material. We were able to cover 14 of the 16 chapters. For a 900 page book, that's good. We completed every topic in the course outline with repeated reinforcement. I provided projects which went through most of the book and had several of my own to challenge them further. Most current data structures books each have several serious problems. There are no classics in this generation in data structures in C++.

We distribute material over more exams and increased the pace of the early part of the course, so graphs can get a separate exam. Someday, I will be able to reach ten exams with a separate one for hashing. This would refine the assessment process and pinpoint areas for improvement even more finely. Class cancellation due to snow storms were problematic too. We lost at least one week.