

COMPASS

A CS Program Assessment Project

Adel Abunawass, Will Lloyd, Edwin Rudolph
Department of Computer Science
University of West Georgia
Carrollton, GA 30118
000-770-836-6485

adel,wlloyd,erudolph@westga.edu

ABSTRACT

In this paper, we describe our Computer Science Program Assessment (COMPASS) project. COMPASS uses open-source software tools to support the development and analysis of course portfolios. We use the portfolios internally to improve the quality of our undergraduate computer science curriculum, and externally to satisfy the requirements of program and university accrediting agencies. COMPASS makes it easy for instructors to build good portfolios and for the department to review and analyze them.

Categories and Subject Descriptors

K.3.2 [Computers and Education]: Computer and Information Science Education – *accreditation, self-assessment*.

General Terms

Management, Measurement, Documentation, Performance.

Keywords

Assessment, accreditation, outcome-based learning.

1. INTRODUCTION

A significant context for computer science programs today is the demand for accountability from our institutions and from external agencies. Our department relies heavily on program assessment to satisfy these demands, to guide improvement of our curriculum and to ensure quality graduates. Like many, we previously relied largely on informal processes and anecdotal data. In some years, the inadequacy of some of these informal practices made self-assessment less than effective. Though we have a well-thought-out curriculum, we could not be sure what was actually being taught and, more importantly, learned, in each class.

To be able to present a more complete and accurate picture of our program, we have now incorporated into our assessment process a

more formal procedure for gathering and documenting course portfolio data we need to monitor our work and implement program improvements. We have named this procedure COMPASS (Computer Science Program Assessment).

Accrediting criteria for both university and professional accreditation agencies require that programs define measurable learning objectives. Departments must periodically collect data that indicate the extent to which the learning objectives are being met, and must use the analysis of that data to identify ways the programs can be improved. We designed COMPASS to help us answer four questions concerning the learning objectives for our courses:

- Are faculty members assessing students' attainment of the objectives defined for each course?
- Are students mastering the objectives?
- Is each course learning objective at the appropriate intellectual level for that course?
- Across the curriculum as a whole, do the objectives work together to offer our graduates a solid grounding in computer science?

In section 2, we will describe the context in which COMPASS operates: our curriculum and our other assessment procedures. In section 3, we will describe COMPASS itself. Section 4 presents our plans for future work, and section 5 our conclusions.

2. ASSESMENT CONTEXT

2.1 Our curriculum

We redesigned our computer science curriculum several years ago to make it outcomes-driven rather than centered on topics. In our new design, we hewed closely to the knowledge areas, units, topics, and learning objectives defined in CC2001, the Computing Curricula 2001 report [4]. A team of faculty members reviewed the report unit by unit, assigning topics to courses and identifying which learning objectives were linked with each topic. Our decisions were influenced by our departmental and university mission statements, by ABET criteria for computer science program accreditation, and by the nature of our particular student population.

We developed a coordinated network of fifteen computer science courses, supplemented by required classes in mathematics, science, and technical writing. Mindful of our outcomes-based

approach, we tried to ensure that students in each course had had the opportunity to learn the prerequisite material in previous courses.

We then associated each CC2001 learning objective with a level of competence at which we expected our students to learn to perform. We used Bloom's taxonomy of educational objectives [3] to define the levels. Objectives for introductory courses are mostly at the lower levels of the taxonomy, emphasizing knowledge, comprehension, and application of concepts and skills. Upper-level courses include more high-level objectives requiring analysis, synthesis, and evaluation. Grappling with just what we wanted our students to learn to be able to do at each stage of their academic careers helped us to understand where our curriculum was strong and where there were gaps.

Associating objectives with competency levels showed us where to build redundancy into the curriculum. Many objectives appear in more than one course, with a higher level of competence expected as the student progresses through the curriculum. For example, in CC2001 topic PF3, Fundamental data structures, the topic on linked structures is associated with the learning objective "Write programs that use each of the following data structures: arrays, records, strings, linked lists, stacks, queues, and hash tables." In CS 1, we expect the students to learn to implement a collection manager class using a library list class given clear specifications. This objective is at the comprehension level. In CS 2, we ask them to select an appropriate collection class for the design and implementation of a collection manager, an objective at the application level. Then, in our Data Structures and Discrete Mathematics sequence, we assign the tasks of developing a list class using dynamic memory, an objective that requires synthesis of several concepts, and of demonstrating that the implementation is correct, which requires analysis.

Determining the level at which an objective should be mastered in a given course was often a challenging exercise, but working through these issues has proven to be both a powerful motivation and a crucial foundation for our assessment efforts. As motivation, knowing precisely what we expected students to be able to do coming into and leaving each course made it clear that we would need to assess how well the students had in fact mastered that material. As foundation, a clear understanding of what we want our students to learn, and when, is a necessary precondition for determining how well they, and we, are doing.

The departmental syllabus for each course lists the topics and learning objectives with the required competency levels. Faculty members use these documents as the basis for their individual syllabi. Of the roughly 45 lecture hours available for each CS course, 30 to 35 hours are devoted to the topics in CC2001, with the rest left up to the needs and interests of individual teachers and their students. A teacher may use the third of the time allocated to the course that is not covered by the departmental objectives to go beyond the specified topics and objectives, but the requirements for subsequent courses assume that only those minimums have been met.

This balance between prescription and freedom is pedagogically important; it allows instructors the flexibility to tailor each course while ensuring that students learn what they need to succeed in their later classes. It is also important for assessment purposes. Course evaluation can focus on the essentials so that the instructor

is not asked to spend too much time documenting what happens and evaluators are not overwhelmed with data.

2.2 The existing assessment process

Revising the curriculum and preparing for three program evaluations – one by our state university system, one by our university's regional accrediting agency, and one by a team from the Computing Accreditation Commission – highlighted the importance of program assessment.

Before developing COMPASS, we used five standard procedures to collect assessment data: alumni questionnaires, a capstone course, senior exit interviews, student surveys, and course portfolios. Alumni questionnaires ask how well their major prepared the graduates for work or graduate school. The capstone course includes a project that demonstrates the degree to which the students can integrate several themes from the curriculum. Exit interviews and student surveys give students the chance to reflect on the positives and negatives of their undergraduate experience. Course portfolios present the personal class assessment and student work sampling materials described by [2] as the foundation of assessment activities. They include syllabi, resources like handouts and lecture slides, and samples of student work (at least a good, average, and poor example) for each major assignment and test.

Each procedure provides valuable feedback that the department chair and undergraduate curriculum committee could use to "close the loop" in our assessment process. We have used the feedback to develop changes in the curriculum. In particular, the evidence indicated that there were inconsistencies in what students were learning in different sections of some courses, that articulation between the introductory sequence and second-year courses needed improvement, and that students needed more exposure to a variety of computing platforms and programming paradigms. We used this evidence to begin revision of several course syllabi.

Equally important, however, was that these revision efforts exposed a weakness in our suite of assessment attempts. We realized that although the data gave a good overview of what was working and what was not, we lacked detailed information about what instructors were really asking their students to learn in class. The grain of information was too coarse; it was often difficult to tell what objectives were being assessed by an assignment.

The course portfolios were the weak link in our process. The portfolios themselves were inconsistent in format and uneven in quality. For each class offered, the instructor turned in a manila folder containing the material. Much of it was on paper, some on diskettes or zip drives, and some consisted of links to course web sites. Some faculty members handed in complete portfolios on time, but others were not so efficient. Doing a good job of building and evaluating the portfolios was asking too much of already hard-worked faculty members.

Moreover, evaluating the data was difficult: there was lots of it, and it was not easy to manage, store, or access. Doing a thorough assessment was practically infeasible. Finding out how and when an objective was assessed meant digging through multiple folders and trying to understand what various assignments were intended to measure. Understanding how related objectives were tied together was even harder. Clearly, we needed a better way to collect, present, and analyze course portfolio data.

3. COMPASS

To improve the course portfolios, we decided to develop COMPASS, a web-based system to document course activities and enable better course evaluation. Our goal is to automate as much as possible personal class assessment and student work sampling. To reach this goal, we had to standardize the format of the portfolios, make entering the data as painless as possible for faculty members, and let the department chair and curriculum committee access and manipulate the data efficiently and flexibly.

We attacked the problem on two fronts: we would use a common course management system for all CS classes, and we would provide an uncomplicated way for student work to be tied to course learning objectives and levels of competence. To save money and time, and to avoid being locked into proprietary solutions, we looked for open-source software that we could use off-the-shelf or easily adapt. Technical resources in our department are limited, so the software had to be straightforward to install and administer.

3.1 The course management system

The first step was to require that every computer science class use the same course management system. We wanted it to be relatively painless to assemble course materials, without imposing restrictions on class content or teaching style. We also wanted to be able to build hooks into the system to support collection of the data on objectives and levels of competence.

Our university uses WebCT [8] as its course management system for distance education and web-supplemented classes, and several computer science faculty members used it extensively in their classes. For this project, however, WebCT was not a good fit. It has a proprietary, closed architecture, is not easy to modify, and is administered from a server over which the computer science department has no control.

Moodle, the Modular Object-Oriented Dynamic Learning Environment [5], proved better for our purposes. An open-source course management system, it is distributed under the terms of the GNU General Public License, and is used at several hundred educational institutions around the world. It runs on any system that supports the PHP scripting language [1], including UNIX, Linux, Windows, and Mac OS X. It is easy to install and administer. Moodle stores its data in a single relational database, and works smoothly with open-source database management systems like MySQL [6], which we use, as well as with commercial databases. Its modular architecture makes it easy to develop plug-ins for local customization.

Moodle does not offer every feature offered by WebCT, but it does provide what most instructors use. Faculty members can structure the course web site by weeks or by topics. They can post lecture notes and other resources in a variety of formats, specify assignments, and develop question banks. They can interact with students by giving online objective-format quizzes with automatic grading, providing feedback on assignments, conducting surveys, and running synchronous chats and asynchronous threaded discussions. Students can upload assignments, see their grades and feedback, participate in group work, and keep a journal of their activities.

The department encourages its faculty to use the course web sites as the core of their course portfolios. Slides, software, links, and

even hand-written notes captured by a smart whiteboard can all be posted. This works well not only for assessment, but for instruction, too, since students have one place to go to access course resources.

Whenever possible, students upload assignments and tests. This not only simplifies the instructor's job, but provides a complete sample of student work. If an assignment or part of an assignment cannot be submitted electronically, the instructor scans a sample of the students' work – usually five examples, ranging from the best to the worst – and uploads the scanned files. Most faculty members have adjusted to this requirement by reducing the number of assignments they give that cannot be uploaded by the students, but some still grumble that scanning documents is as much a burden as copying was under the old system. Luckily, with several classrooms equipped with computers and with a campus wireless network, even most test questions can be submitted to Moodle without faculty intervention, so the amount of material to be scanned is small.

Once uploaded, class resources and student work are stored in Moodle's database. Since this is a standard relational database, the data is easy to get to from within Moodle or by directly accessing the database tables. This will give us the flexibility to use the data in new ways as our requirements evolve.

3.2 Linking objectives and student work

With Moodle providing the means for standardizing how course materials would be collected and stored, we could develop a system to let instructors link each assignment with one or more course objectives, specify the level of competence the assignment required, and comment on how well the assignment met its goals.

First, we designed and implemented a curriculum database that stores the learning objectives and their associated levels of competence for each course. This data is organized by term, so that modifications to the curriculum do not affect information from previous terms. In keeping with our project goals, we use MySQL to store the data.

We inserted a hook into Moodle to include a link from each course homepage to a site that lets faculty members enter assessment data about an assignment. The link displays only when a faculty member logs in, so students do not see it. The linked site is a front-end to the database that stores the objectives and levels of competence. PHP scripts generate web pages that let the instructor review the course objectives, read and modify information on assignments already given, and add a new assignment.

For each assignment, whether it be a project, presentation, paper, or test, the instructor enters the name the assignment was given in Moodle, and selects from a list of the course learning objectives those that the assignment is designed to assess. For each objective selected, the instructor also selects from a drop-down list the category from Bloom's taxonomy that best indicates the level of competence that the assignment requires. Check boxes ask for information on the computing platforms, programming languages, and tools used, and let the instructor indicate whether the assignment evaluated students' oral or written communication.

In addition to this objective data, a text field provides a space for more subjective information. The instructor can reflect on how well the assignment satisfied its goals. We also encourage faculty

members to state any other concerns, such as whether the objective and competence level seem appropriate for the course, and whether the students had the necessary background to do well on the assignment.

Determining what to enter can be a bit daunting, especially the first few times a faculty member uses the system, so links offer access to helpful information like the departmental syllabus for the course, the CC2001 web site, and sites that describe the kinds of activities that correspond to the different Bloom levels, like [7]. These links open in new windows so the instructor can use this information while working on the assignment.

Once entered, the assignment data is added to the curriculum database. Since all the data in the system is held in normalized tables, access to it is very flexible. It can be retrieved by course section or by learning objective. A reviewer can see what happened in individual courses, and can readily investigate how particular learning objectives were assessed across the curriculum. Going back to Moodle, the reviewer can look at the student work for an assignment to judge whether the data that the instructor entered is accurate.

4. FUTURE WORK

Our work on COMPASS will continue in two directions. We plan to make several of the current functions easier to perform, and to add a component for student data entry.

Currently, most administrative and review functions require direct interaction with the underlying database using SQL commands. We will develop PHP scripts to generate and process web pages that hide the low-level details for performing commonly performed tasks. A reviewer will be able, for example, to select a term and a learning objective and have a nicely-formatted report display showing the courses in which the objective was assessed.

The reviewer will also be able to start or to contribute to a threaded discussion about an assessment topic from any of the system's web pages. We rely now on email and committee meetings to conduct these conversations, but online discussions would facilitate the exchange of ideas and document how we use the data to improve the curriculum.

A relatively minor, but still useful, change will be to improve the linking between Moodle and the web pages we create. We will add links from Moodle's assignment and quiz creation pages to our assignment data entry page, rather than having links from the course homepage. Faculty members are more likely to do a good job of associating assignments and objectives if we can make doing so a normal part of creating an assignment rather than an add-on that must be done later. The script that generates the data entry page can then supply the assignment name and create a link from the assessment data page back to the assignment page in Moodle.

Departmental course syllabi are currently stored as word processor documents available on the department web site, but they will be integrated into the curriculum database. When the curriculum committee uses assessment results to change the objectives of a course, the syllabus will then be able to reflect those changes without having to be updated manually.

An interesting area for investigation is to solicit and analyze student input on their learning. The system presently provides only for faculty input, but know what students believe they are

learning in their courses would add another dimension to our assessment process.

We will add a link from the students' homepage for each class to a page that lists the learning objectives for that course. Accompanying each objective will be a drop-down list whose items will indicate the degree to which the course has offered the opportunity to achieve the objective. Choices will range from "Objective has not been covered in this class" to "Objective has been covered thoroughly in this class."

There will also be a text field into which the student can enter free-form comments pertaining to the objective. The page will display previously entered comments and ratings, much like a customer review page at an online bookstore.

Student contributions will, of course, be anonymous. We will have to assure the students that their responses will not be linked to their names and will not affect their grades, and that their responses will be stored in the course portfolio database only in aggregate. Not all students will respond, but our experience with surveys and exit interviews has demonstrated that many will be eager to share their thoughts on how well our courses are meeting their goals.

5. CONCLUSIONS

Though the COMPASS project is still a work in progress, we are encouraged by early results. The technical side of the project has been almost trouble-free. Using open-source software has proved a good choice. The switch from WebCT to Moodle has gone well. Moodle's simple, modular architecture has made putting hooks into it trouble-free. Using MySQL and PHP for developing the course portfolio database and web pages has also been a success. Both have proven robust and easy to use.

More important, of course, is the effectiveness of our tools for instruction and assessment. Moodle is proving to be as popular with faculty members as it is with our technical staff. Several faculty members who resisted using WebCT because of its complexity have found Moodle easy to use. This has made standardization of the collection of course materials possible. Though there is still a lot of work involved in reviewing this information, we no longer have to fight with shelves of folders leaking paper and diskettes.

Learning to associate objectives with levels of competence has not been much of a problem. It is not important that every member of the department become an expert on Bloom's taxonomy. A reasonable appraisal of whether an assignment demands, say, basic knowledge of a concept or skilled analysis is obvious in most cases. Differences of opinion on whether an assignment requires, in Bloom's terminology, comprehension or application do not affect the usefulness of the information.

Having faculty members learn to think about the objectives they intend each assignment to assess is demanding a more significant culture change. We are still at the early-adopters stage in this. Faculty members tend to think about associating an assignment with its objectives after they have developed the assignment rather than devising the assignment specifically to assess the students' mastery of certain objectives. We are optimistic that continued practice and improved tools will make this shift in attitude possible.

Unsurprisingly, we have made more progress with courses that we are revising than with the more stable ones. Our work on the

introductory sequence, in particular, has generated much fruitful discussion about what we expect from the students and how to judge their development. In these courses we are finding that COMPASS is meeting its goals. Faculty members are assessing students' attainment of the defined objectives; discovering the extent to which students are mastering the objectives; determining whether each course learning objective is at the appropriate intellectual level; and seeing how the objectives work together to offer a solid introduction to computer science.

Faculty preparations for next term's offerings in software engineering, operating systems, intelligent systems, and net-centric computing are also showing the influence of the culture change that COMPASS is supporting. The challenge will be to continue this momentum so that faculty members will transfer what they learn working on these courses to the rest of the curriculum.

6. ACKNOWLEDGMENTS

This work was funded in part by NSF CCLI-A&I Award No. DUE-0088204.

7. REFERENCES

- [1] Apache Software Foundation. <http://www.php.net/>
- [2] Blandford, Dick K., and Hwang, Deborah J. Five easy but effective assessment methods. In Proceedings of SIGCSE'03 (Reno, Nevada, February, 2003). ACM Press, 41-44.
- [3] Bloom, B.S. (ed.) Taxonomy of educational objectives: The classification of educational goals: Handbook I, cognitive domain. Longmans, Green, New York, Toronto, 1956.
- [4] CC2001 Task Force. Computing Curricula 2001: Final Report. <http://www.computer.org/education/cc2001/final/index.htm>.
- [5] Dougiamas, Martin. Moodle. <http://moodle.org/>
- [6] MySQL AB. MySQL. <http://www.mysql.com/>
- [7] Counselling Services, University of Victoria. Learning Services Program: Bloom's taxonomy. <http://www.coun.uvic.ca/learn/program/hndouts/bloom.html>.
- [8] WebCT, Inc. <http://www.webct.com/>.