2005-1814

Curricular Integration of Computational Tools: A First Step

Timothy Hinds, Mark Urban-Lurain, Jon Sticklen,

Marilyn Amey, Taner Eskil

Michigan State University

Abstract

Calls for new paradigms for engineering education are widespread.[1-3] Yet, major curricular change is difficult to accomplish for many reasons, including having the necessary faculty buy-in.[4] Generally, efforts can be classified as either *top-down/structural*, in which faculty assess an entire program of study and address needs in each component before implementation begins; or *bottom-up/individual*, a more traditional approach that implements change in one course at a time. Faculty buy-in, consensus, and resources (unit and institutional) needed for the top-down approach make it difficult to accomplish. On the other hand, the bottom-up model is slow, the assumption that curricular reform can be affected by an accumulation of *individual* course adaptations is unproven, and the change goals need to have a more systemic focus. Unless the curriculum helps students integrate material across their courses, they have difficulty seeing how the material they learn in one course will connect to the next.

We have performed a pair of initial studies using an evolutionary approach to curricular reform that capitalized on the strengths of both the top-down and bottom-up models, and was built on the science, technology, engineering and mathematics (STEM) reform literature. This approach developed a pairwise linkage among strategic courses in the engineering curricula to promote curricular integration and helped students see connections between their first-year courses and subsequent courses.

Vertically integrated problem-based learning scenarios that link across courses are crucial to this model. Pre-reform data collected in the first study showed that students taking an introductory computing course did not see the importance of learning a particular software tool (MATLAB), because they did not see connections to their future courses. This had negative impacts on student motivation, learning, and retention. In our recent work, which was our first vertical effort, we focused on MATLAB with integration of the learning of this engineering tool in an

introductory computing course with the solution of statics problems in an introductory mechanical engineering course. Our recent study set out to determine if joint team efforts would enhance student perceptions of the set learning goal for the introductory computing students while enhancing learning outcomes for both the introductory computing and introductory mechanical engineering students.

The paper outlines this pairwise linkage model, the goals of this project, the framework for evaluating the linkage and the types of data we collected as part of the evaluation effort. Results from the initial study confirmed that problem-based teamwork enhanced student attitudes towards MATLAB. We also describe how results here will enable us to reach our long-term goal of curricular integration.

Introduction

Undergraduate education in engineering has been generally successful over the last fifty years as measured by the most important metric: a well-educated and productive cadre of effective engineers in the engineering professions. However, critics have rightly pointed out increasing difficulties in the nation's engineering curricula and resultant general shortcomings of engineering graduates as determined by outcomes assessment. Although these shortcomings take many faces, root causes are traceable to shortcomings in the core-defining characteristic of an effective engineer: strong problem solving ability. Effective problem solving is predicated on: (a) thorough understanding of technical background material required for the problem at hand or an ability to obtain that understanding; (b) ability to integrate background material; (c) ability to sharpen a stated problem and produce a well-structured problem from an ill-structured problem; (d) ability to apply the background material systematically and effectively to the problem; (e) ability to critically interpret the results of the problem solving; and (f) ability to communicate the results of the problem solving. Underlying and pervasive through this process is the ability to work in a team towards the problem solving goal.

Undergraduate engineering education as reflected in engineering curricula in the United States has focused strongly on criterion (a) above to the detriment of the other items in the list. Indeed, many if not most engineering classes have focused on a thorough grounding in the "basics" of a given discipline as delivered through lecture. This slow but steady evolution to greater reliance on lecture about more and more material is a reflection of exploding amounts of knowledge in the engineering disciplines over the last fifty years. Yet, with ever more knowledge to be imparted, engineering students find themselves with so many details to master that they have, in general, lost sight of the goal: effective problem solving predicated on *integrated student understanding* of technical material.

There are a number of NSF- or corporate-sponsored consortia in which change to (especially) tighter curricular integration has been set as the goal, often identified as *systemic curricular change*. There are two general situations in which such deep change is most feasible: (a) a new institution with a curricular blank slate; or (b) an institution in which a consensus of faculty support wholesale curricular change. Although the NSF-sponsored consortia are due praise for the results produced, most US engineering schools fall into neither of these two categories.

Despite the commonly held view that systemic curricular change is needed, the conundrum is how to achieve it. In short, how to build the necessary faculty consensus?

In a previous paper[5], we proposed an evolutionary approach to curricular reform that capitalizes on the strengths of both the top-down and bottom-up models, and builds on the STEM reform literature. In summary, this approach develops multiple, pairwise linkages among strategic classes in the engineering curricula to promote curricular integration and help students see connections between their first-year courses and subsequent courses. We capitalize on the strengths of both the top-down and single-course, bottom-up models, and build on the STEM reform literature. Instead of developing a blueprint for total change followed by implementing the entire blueprint into the curriculum, we believe that faculty consensus can be built piecemeal by building, from the bottom-up, *pairwise linkages* between courses based on content that students need to integrate across the curriculum such that each linkage will reinforce and build on prior student experience. A pairwise linkage could be a *soft link* in which conceptual material from an earlier course could be assumed and built upon in the higher level course of the linkage. More interestingly, a pairwise linkage could also be a *hard link* such as that formed by having term project teams consist of students from both courses. It is our working belief that by implementing *over time* a set of pairwise linkages across a curriculum that faculty support will evolve towards more support for further developing such linkages and, more importantly, towards maintaining the linkages.

A specific example is a concern about the lack of strong quantitative problem solving ability for undergraduate engineering students as manifest in student understanding of computer-based computational tools that support technical problem solving. General computational environments such as MATLAB, MathCad, and Mathematica are all versatile and capable of being used in most situations of relevance to undergraduate engineering students. Because of their varied background training, most engineering faculty have individually learned one or another of these computational environments. Typical undergraduate engineering curricula reflect professor preference in the assignment of computational tools for students to use to complete problem sets and projects. The implicit assumption has been that undergraduate engineering students can simply "pick up" a specified computational tool and apply it to assigned problems. This assumption is false. Modern computational environments are replete with many "features" that can each be leveraged for a given class of problem. However, this "high power" comes at a high price: a steep learning curve for students. A typical engineering undergraduate has a difficult time in applying the tools of a computational environment like MATLAB in other than cookbook fashion unless the student has systematically developed an understanding of the computational environment from an integrated viewpoint.

The use of MATLAB through the entire undergraduate experience can be thought of as a *vertical slice* through an engineering curriculum. Ideally, this vertical slice would touch all individual courses that a student would take. The development of such a vertical slice through engineering curricula requires a whole-curriculum perspective, but with an innovative twist. Instead of focusing on total conceptual content of a curriculum, we focus on one conceptual topic in this project: curricular integration of MATLAB through the Mechanical, Civil and Environmental, and Chemical Engineering and Materials Science Departments at Michigan State University. By taking the pairwise-linkage approach to reform using computational skills as the learning stream

(vertical slice), the learning outcomes of individual classes also become more connected in process, knowledge, and application for the students. The curriculum becomes less a set of courses and more an *integrated set of learning experiences*.

Project goals

We are currently in the beginning stages of this project with the goal of integrating MATLAB across the curricula of Mechanical, Civil and Environmental, and Chemical Engineering and Materials Science Departments at Michigan State University (MSU). We expect this will produce two desirable results. First, students in the three departments will attain *mastery* of the MATLAB computational environment, including both application mastery of a subset of MATLAB functionality and the ability to learn more about MATLAB as needed. To meet this goal, we are developing pairwise linkages between strategic courses in the curricula of each department.

Tools that support technical problem solving such as mathematical methods (for example, calculus) or computational computer-based tools (for example, MATLAB) should clearly not be viewed in isolation – as ends unto themselves. Rather, they should be understood in relation to discipline-specific problem solving. While a great deal of effort has been expended in examining the pedagogical linkage between (e.g.,) calculus and engineering, very little attention has been paid to the possible pedagogical linkages between computer-based computational environments like MATLAB and discipline-specific knowledge. We hypothesize that knowledge structures students build in mastering MATLAB may be used as anchors in seeing commonality between discipline-specific knowledge constructs. If this hypothesis is confirmed, the result would be a better understanding of how learning computational tools and learning discipline specific concepts interact, and how under appropriate conditions, learning to use computer-based tools like MATLAB can help students develop a sound and usable knowledge structure for understanding in their disciplinary domain.

Preliminary baseline data

We sought to establish baseline data using a team approach to teaching MATLAB in the introductory computing course for engineers (CSE131) taught by one of the authors (Sticklen) during summer, 2003 (US03.) The students (n=31) represent a typical mix of MSU engineering students. They were 81% male and 19% female. Their majors include Chemical, Mechanical, Civil, Biomechanical and Materials Science Engineering, along with computer science, mathematics, physics, no preference and business students. These students were a little older than students in this course during the fall or spring semesters: 39% were sophomores, 32% were juniors, 19% were seniors and only 6% were first year students.

The course has traditionally taught MATLAB, but during this semester, the instructor had students working in teams to learn MATLAB in the context of solving a variety of problems. At the end of the course, we surveyed the students to determine their perceptions of learning MATLAB, teamwork, and if they perceived the utility of using MATLAB in their other courses.

The vast majority (93%) reported that they enjoyed the teamwork and 78% could see the importance of teamwork; however, just over half (52%) believed that they understood teamwork

better after the experience. They believed that the teamwork helped them learn MATLAB (78%) and were confident that they could figure out how to perform tasks in MATLAB that they were not explicitly taught in the course: 78% believed that they could implement a FOR loop. There was a moderate correlation ($r=.52$, $p< .005$) between these items, suggesting that the goal of "near transfer" may have been met.

However, on the question of "far transfer" – do students make connections between MATLAB and their other courses – the students were less inclined to make these connections. Sixty-three percent of the students did not see that MATLAB could be important to their later courses and 56% disagreed that learning MATLAB will help them learn other subjects. Furthermore, these two items were highly correlated ($r=.67$, $p< .0001$) with each other and were also moderately correlated ($r = .4$ to $.5$) with the questions about near transfer and the questions about teamwork. It appears that using teams had a positive impact on the students' ability to use MATLAB not just to solve the immediate problems in the course, but on their ability to extend this knowledge of MATLAB to other problems and their understanding of the broader applications of MATLAB. These data suggest that structuring the group exercises with a more formal emphasis on these types of transfer may have a beneficial impact.

First step towards pairwise linkage

A second, more intensive study was performed during the summer of 2004 (US04.) In this work, students in the introductory computing course for engineers (CSE131) were teamed with students in the introductory engineering mechanics course, statics (ME221). The teams of students were assigned to solve mechanics problems using the MATLAB tool. This grouping occurred at the mid-point of the summer semester approximately 3 weeks into the accelerated, 7-week academic period. At this point, the computer science students had obtained a working understanding of the MATLAB tool and the statics students had gained enough knowledge to solve basic engineering mechanics problems. To enable the students in the two courses to work together on the assignments, the statics students were split into two groups that attended one of two MATLAB laboratory sessions per week with their computer science counterparts.

Twenty-six (22 male, 4 female) computer science and 31 (21 male, 10 female) statics students participated in the study. Of the 31 statics students, 19 had taken the introductory computer science course (CSE131) in a prior semester, 11 others had taken a similar computer course focused on object-centered programming (C++), and one student was concurrently enrolled in both the statics and computer science course.

Teams were composed of 2 to 3 statics students grouped with 1 to 2 computer science students forming a total of 15 teams. Although the teams were randomly selected, care was taken to balance the teams based on ability such that not all high or low ability students were placed on the same teams. Student ability was determined by examination performance to date in each of the courses.

The assignments consisted of a series of three sets of statics homework problems in which the engineering mechanics students were to tutor their computer science counterparts in the development of the problems while the computer science students were to teach the mechanics

students methods for solving the problems using MATLAB. The problems were simple in nature such that they could be solved using traditional, hand calculations. However, the problems also contained sufficient complexity such that using MATLAB would yield faster solutions. Together, the students were to teach each other their respective subjects, develop solutions to the given problems and submit a single set of problem solutions.

Data

We surveyed students in both courses before assigning them to the cross-course teams and then again at the end of the semester. The surveys asked students about their experience working in teams, their attitudes towards teamwork, their experience with MATLAB, and their perceptions of the usefulness of MATLAB for solving technical problems and helping them in their technical courses. The surveys used five point Likert scales, asking students to rate the statements from strongly agree to strongly disagree. Forty-seven students completed the first survey and 19 students completed the second survey; 17 of those students had completed the first survey so we were able to measure changes in their perceptions after participating in the cross-course teams.

Student perceptions of working in teams

There were no substantial changes in students' perceptions of working in teams after the course. In response to the question "I think that working in teams will probably help me learn to be a better problem solver," two students' responses improved, and three students responses' declined, while the rest remained unchanged. In response to the question "I think that working in teams is superfluous to 'getting the job done'," four students' responses improved, four declined, and the rest remained unchanged.

We asked students for comments about working in technical teams in general and cross-course teams in particular. Several students reported the experience was helpful. Most of those found the different backgrounds of their team members an asset. Below are some representative responses:

- It worked well; we each filled in where the other had weaknesses
- I found it helpful to have someone to ask when I encountered problems
- Other students from the class should explain the problem.
- Technical teams worked fine, but cross-class could use some improvement
- Working in a CSE131 group was fine
- Working in groups definitely helped out
- It would have been nice if we could have gotten more time to work with the ME students
- I like it

On the other hand, about half of the students found the experience to be difficult. Their concerns were mostly about communications problems and a perception of unequal work loads. Some of the communications concerns were:

- It is good only when everyone is on the same page
- It was hard to follow others work due to our lack of knowledge in their subjects
- It is not easy for students to learn the material themselves and teach others
- Information was hard to relay back and forth
- Communications outside class was an issue
- There is a small barrier; sometimes people don't understand each other, even after an explanation
- It was hard to communicate outside of the lab; but it was overall a good experience

Among the workload issues that students reported were:

- It is difficult when team members do not understand or do their share of the work
- Technical work team [sic] will be more effective if all members of the group willing [sic] to take their parts, otherwise the load will be only on some members responsibility
- Not everyone puts in the same amount of work; Counting on other people could hurt
- It is difficult to work in a team when people don't care to be part of it
- I didn't like it because not everyone had full understanding of everything
- The weight of the grade on the students overall grade needs to be the same in order for this to work
- It wasn't teamwork because they didn't know what we wanted and we didn't know what they were getting
- Not a good idea, my group members didn't do their work on time

Finally, some students felt that the team experience could have been better organized:

- It would have been easier to work in teams had the actual project been better organized and had I had more experience with teams
- With different courses, there are different guidelines
- More organization

Student perceptions of MATLAB

The impact of the cross-course teams was more evident in student perceptions of MATLAB. Over a third of the respondents rated the question "I consider MATLAB a useful tool for doing technical problems" more highly after the team exercises than before, with only three students rating this item lower. Similarly, 47% of the students' ratings of "I consider MATLAB to be a tool I need to know how to use to do well in my technical course work" improved, with only 12% of the students' ratings declining. On the question "I consider MATLAB to be a tool I need to know how to use to do well as a technical problem solver, 30% of the students' ratings improved, only one student's rating declined and the rest remained unchanged. Clearly, the students found the use of the tool in the context of solving particular problems helpful. This result is a substantial change from the summer, 2003 course offering in computer science, where 63% of the students did not see the usefulness of MATLAB in their subsequent courses in engineering.

Student academic outcomes

We wanted to know what impact these changes in the courses may have had on student academic outcomes. Research on a larger computer science course[6-8] showed that students who took the course during the summer semester differed academically from students who took the course during the school year. Summer students tended to be bimodal, with either high-achieving students who were trying to get ahead on their programs, or students who were in academic trouble and who generally did not attend class and received poor grades. Examining the grade distribution for both the CSE and ME courses from summer 2003 and summer 2004 shows that the distributions are not normal and may have a similar bimodal character. Hence, we used the Mann-Whitney non-parametric test to compare outcomes between US03 and US04.

In the mechanical engineering course, the average grade was 2.57 in US03 (n=38) and 3.03 in US04 (n=34). This change is significant ($Z = -2.22$, $p < .03$). In the computer science course, the average grade was 3.16 in US03 (n=31) and 3.40 in US04 (n=26). However, this change is not quite significant ($Z = -1.34$, $p < .18$). While it is not possible to claim that this improvement is only due to the cross-course teamwork, the content of both courses was very similar from US03 to US04, with the addition of the cross-course team activities as the major change that was implemented.

Conclusions

In our preliminary trial of creating a pair-wise linkage between a computer science course and a mechanical engineering course, we found that students had mixed responses to the cross course teams, but that their perceptions of the utility of MATLAB were substantially improved. The results of this trial also produced higher course grades in both courses, though the change was most significant for the mechanical engineering course. It should also be emphasized that the assessment presented here is indeed preliminary. Full evaluation of the project cannot be obtained until the study is continued to include statics students who also participated when they were previously enrolled in the computer science course.

Lessons learned

Although the study provided insight as to how the teams of students were able to collaborate in the solving of engineering problems, the researchers also found ways so as to improve upon the research methods in subsequent studies.

First, substantial preparation is necessary to better ensure the integration of the teams. As discussed above, each team as a whole was responsible for completion and submission of the assignments. However, quite often, the statics students would put the burden of completing the computer computations to their computer science partners. Many teams took a bipolar approach with the engineering mechanics students responsible for determining and developing the equations and other formulae necessary to solve the given problems leaving the computer science course students to create the computer solutions on their own.

Secondly, grading of the students' work in each of the courses should have been more equal. For the students in the computer science course, each assignment was a substantial portion of their individual course grades. Just as the work performed earlier in the semester by these students had been, the collaborative assignments were viewed and graded as major projects due at regular intervals. This was not the case for the statics students as the group projects were assigned as individual homework sets with much less of an overall impact on their final course grades. Thus, they formed an attitude to some extent by their perception that their individual grades did not depend on the output of work from the joint groups.

The students in the computer science course submitted the work of the entire group to a course website where it was graded by a graduate teaching assistant with all group members receiving the same score. Again, this reinforced the position where the computer science course students had a greater share of the responsibility for completion of the projects.

Unfortunately, the sets of statics problems selected were simplistic enough such that the computer science students easily developed enough mechanics skills to create the solutions. This, again, placed a larger burden of the work on those students. The problems should be selected such that the knowledge of the statics students plays a greater role in the development of the solutions.

In future endeavors, we will have the groups present their work in the laboratory setting. The presentations will be part of each student's final course grade. This requirement will further develop the teamwork aspect of the project by sharing responsibility for the group work and providing evidence to both sets of students that they have participated in a joint activity. And, it will also provide the students with an opportunity to practice their oral presentation skills. We plan to incorporate these lessons when we perform another cross course teaming study in the summer, 2005.

Bibliography

1. Engineering Accreditation Commission. Criteria for accrediting engineering programs. Baltimore, MD: Engineering Accreditation Commission; 2000. p 59.
2. National Research Council. Improving engineering design: Designing for competitive advantage. Washington, DC: National Science Foundation; 1991.
3. National Science Board. Science and Engineering Indicators 2002. Arlington, VA: National Science Foundation; 2002 April. Report nr NSB-02-1.
4. Terenzini P, and others. Students' out-of-class experiences and their influence on learning and cognitive development: A literature review. Journal of College Student Development 1996; 37(2):149-162.
5. Urban-Lurain M, Amey M, Sticklen J, Hinds T, Eskil T. Curricular integration of computational tools by evolutionary steps. 2004 June 20-23; Salt Lake City, Utah. American Society for Engineering Education. p 11.
6. Urban-Lurain M, Weinshank DJ. Is there a role for programming in non-major CS courses? In: Pavelich M, editor; 2000 October, 2000; Kansas City, MO. p T2B-7-T2B-11.
7. Urban-Lurain M, Weinshank DJ. Attendance and outcomes in a large, collaborative learning, performance assessment course. http://www.cse.msu.edu/rgroups/cse101/AERA2000/attendance.htm; 2000.
8. Urban-Lurain M, Weinshank DJ. Mastering computing technology: A new approach for non-computer science majors. 1999 April 20, 1999; Montreal, CA. American Educational Research Association.

Author Biographies

TIMOTHY HINDS is an Academic Specialist in the Michigan State University Department of Mechanical Engineering. He teaches undergraduate courses in machine design, manufacturing processes and mechanics. He also teaches a senior-level undergraduate international design project course and has taught graduate-level courses in engineering innovation and technology management.

MARK URBAN-LURAIN is Director of Instructional Technology Research and Development in the Division of Science and Mathematics Education at Michigan State University. He is responsible for providing vision, direction, planning and implementation for using technology mathematics and science education and developed several introductory computer science courses for non-computer science students serving 2000 students per semester.

JON STICKLEN is an Associate Professor in the Department of Computer Science and Engineering at Michigan State University. He has had a strong research record in computer science research, specifically in knowledge-based systems. His main contributions have been in the theory and application of principled approaches to knowledge-based systems following a school of thought known as “task specific approaches.”

MARILYN AMEY is Associate Professor in the Department of Educational Administration and Chair of the Higher, Adult, and Lifelong Education Program at MSU. She was part of a research team studying best practices in Science, Math, Engineering and Technology Undergraduate Reform for SRI and NSF, and policy evaluator for an NSF Rural Systemic Reform project on math and science curriculum reform in the Navajo Nation.

TANER ESKIL is a Ph.D. candidate in the Department of Computer Science and Engineering at Michigan State University. Mr. Eskil holds a M.Sc. in Mechanical Engineering and will soon complete his Ph.D. research in the area of Internet agent support for electronic commerce. Mr. Eskil has been instrumental in developments in the College of Engineering freshman gateway course in computational tools.