Session Number 2004-1543

Curricular Integration of Computational Tools by Evolutionary Steps

Mark Urban-Lurain, Marilyn Amey, Jon Sticklen, Timothy Hinds, Taner Eskil

Michigan State University

Abstract

Calls for new paradigms for engineering education are widespread.[1, 2, 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 topdown/ 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 class 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 the courses, they have difficulty seeing how the material they learn in one course will connect to the next.

We propose 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. 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.

Vertically integrated problem-based learning scenarios that link across courses are crucial to this model. Our first vertical effort focuses on MatLab, to integrate learning of this engineering tool in an introductory computing course with the solution of statics problems in an introductory mechanical engineering course. Pre-reform data show that students taking the introductory computing course do not see the importance of learning MatLab, because they do not see connections to their future courses. This has negative impacts on student motivation, learning, and retention.

The paper outlines this pairwise linkages model, the goals of this project, the framework for evaluating the linkages and the types of data we are collecting as part of the evaluation effort.

Results from the current study confirm that problem-based team work enhances student attitudes towards MatLab.

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 through 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.

In 1991, the National Research Council[1] criticized undergraduate engineering curricula for not reflecting the shifting needs of the engineering profession by saying that these curricula are "lacking the essential interdisciplinary character of modern design practice" (p. 4). As a result, NRC claimed, engineering graduates are poorly prepared to utilize "scientific, mathematical, and analytical knowledge in the design of high quality components, processes, and systems". The ABET 2000 criteria reinforce these perspectives, as has the National Science Foundation in the last decade.[5] Curricular reform efforts have focused on developing new paradigms for engineering education, including an emphasis on active student learning and application of knowledge (including performing design) rather than passive data gathering, faculty acting as mentors and facilitators rather than as lecturers, integration of disciplinary knowledge instead of emphasis on isolated facts, emphasis on deep problem solving including problem specification instead of "plug and chug" application of equations, innovative forms of student assessment focusing on improvement, and a variety of non-technical skills such as communication and teamwork central to the workplace.[2] Bordogna et al.[6] argue that more holistic curricula are needed that weave process knowledge and fact-based knowledge throughout the undergraduate experience. In spite of effective projects funded by NSF, its partner agencies, industry and postsecondary institutions, challenges remain in creating and institutionalizing reform initiatives to enhance learning outcomes in *science, technology, engineering, and mathematics* (STEM) fields.

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?

There are several reasons for this apparent lack of adaptation. One experience to attempt change at MIT is instructive. The MIT Department of Aeronautics and Astronautics incorporated active learning strategies and assessment tools into their Unified Engineering course after a two-year strategic planning process that involved all faculty in the department.[7] As they discovered, "changing how we teach is more difficult than changing what we teach." (p. T2A-15). This change required not only faculty buy-in, but also administrative and institutional support. There is a two-fold message here. First, change in pedagogical methods are, in fact, difficult for faculty. Second, systemic change, particularly if attempted in a "revolutionary" way (with all change to be implemented simultaneously), is even more difficult.

Although the NSF coalition program goals are laudable, such change is often difficult to accomplish on a typical campus because of the necessary faculty buy-in.[4] Put stronger … **achieving faculty buy-in is the major bottleneck constraining systemic curricular change**. Systemic curricular change can be termed a *top-down approach to reform*. Arguments can be made for and against a top-down approach to curriculum reform, in which faculty are involved in assessing an entire program of study and addressing needs in each component *before implementation begins*. In the end however, using a top-down approach proves costly in faculty time, because these resources appear to be fixed. Hence, it is difficult to obtain the levels of faculty buy-in necessary to support systemic curricular change.

Top-down reform can be successful in two general contexts. First, in a new college or department there is no existing educational program and the focus becomes "doing it right the first time." An example can be drawn from the School of Chemical Engineering at the University Rovira i Virgili in Tarragona, Spain that went beyond the departmental level with their reform efforts to incorporate project-based cooperative learning teams. There, first-year chemical engineering students are involved in design projects working in teams that are each led by two fourth-year students. These projects are authentic, real-world tasks, such as the thermal treatment of industrial wastes, which require students to learn and integrate knowledge from calculus, chemistry, physics, fluid mechanics and a number of other courses. This reform transcended an individual department to include faculty from mathematics, chemistry, physics and other disciplines. Change management was critical for this reform and included a strong industry partnership with Dow Chemical Company as part of the faculty management education.[8] The more common situation is that systemic curricular reform is attempted within an existing college, set of departments, and engineering curricula. It is in this context that the NSF-supported consortia have operated, and have largely applied a top-down approach to the problem. In the abstract, systemic curricular reform seems most cleanly and quickly achieved following a top-down approach. However, the revolutionary wholesale change in curricula involved and the associated high level of faculty buy-in necessary to implement change *before any change results are gathered and evaluated* have resulted in little, if any, emulation of the systemic curricular change programs crafted within the consortia contexts.

On the other hand, a more traditional approach is to implement change one class at a time. The seeming advantage of such "bottom-up" approaches is that change is implemented in an evolutionary manner, and thus, the results of change are observable piecemeal, and (assuming the change is successful) local faculty consensus for change is enhanced. In effect, this evolutionary change model ideally would produce a "snowball effect" in developing faculty consensus that would favor change.

However, in addition to the slower rate of change of the bottom-up model, the underlying assumption that curricula reform can be affected by an accumulation of *individual* course adaptations is unproven. The change goals need to have a more systemic focus than typically exists in many single course change efforts. A systemic orientation raises different issues in change processes that impede or support lasting reform. Fisher and Fairweather[9] recently found, for example, that failure to recruit additional faculty to teach the target course (lack of incentives and rewards in the department), lack of fit with the curricula in other engineering departments, and institutional policies regarding student enrollment and declaration of major affected course-level reform efforts. If institutionalization of course-level reform is the goal, success requires taking into account a variety of factors beyond the individual course and department hosting it. Change efforts require knowing what works, what does not work and why, and being able to translate these findings into new circumstances.[10] Innovations are often not sustainable, and even when they are, more comprehensive understandings are required for those successful strategies to be adopted and for less effective measures to be avoided. The bottom line is that a more comprehensive approach to reform is required than that typically followed in the single course change approach to bottom-up reform.

An evolutionary model

We propose 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. 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. Effective change efforts require simultaneous use of multiple strategies and levers applied in systemic and systematic ways, rather than as unrelated and individual strategies.[11] Systemic reform requires potential users to be involved during the development and implementation phases.[12] It requires that faculty and administrators recognize the importance of the innovation to the success of academic programs, rather than as simply the interest of isolated faculty in discrete class settings.[13] However, given faculty culture, reform efforts are often best implemented in smaller steps that permit faculty to maintain individual control over teaching and course development[13] and are most successful when promoted through a focus on adoption rather than dissemination or mandate. Review of the literature on engineering curriculum reform also suggests that crossfunctional and interdisciplinary programs promote breadth of knowledge, team-based skills, and business sense.[11] Faculty need to shift from teacher-centered pedagogies to those in which students become active participants in the generation of knowledge and problem solving. This often equates to "learning-by-doing", greater emphasis on early introduction of learning tools, including computational and technology applications, increased use of team projects based on industry problems, and collaborative learning strategies.[14, 15, 16]

We believe this approach should be familiar to any good engineer: when faced with a complex task, try to decompose the task into manageable subtasks. 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. Like a balance beam, the scales of faculty support will eventually tip to strongly favor the integrated curriculum based on the set of course linkage pairs based on computational tool use. The natural extension to the entire curriculum would then be to identify the unifying concepts that span the curriculum and follow the same route as we have begun to follow, developing for each unifying concept (i.e., each vertical slice) a set of pairwise course linkages, and implementing these linkages over time.

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. Failure to remember, for example, that in MatLab one routinely must build data vectors in order to utilize the rich plotting capability it has can block student use of MatLab for visualizing data in plotted form. Failure to remember (or simply not knowing) how MatLab in general deals with vector data structures can block the entire process.

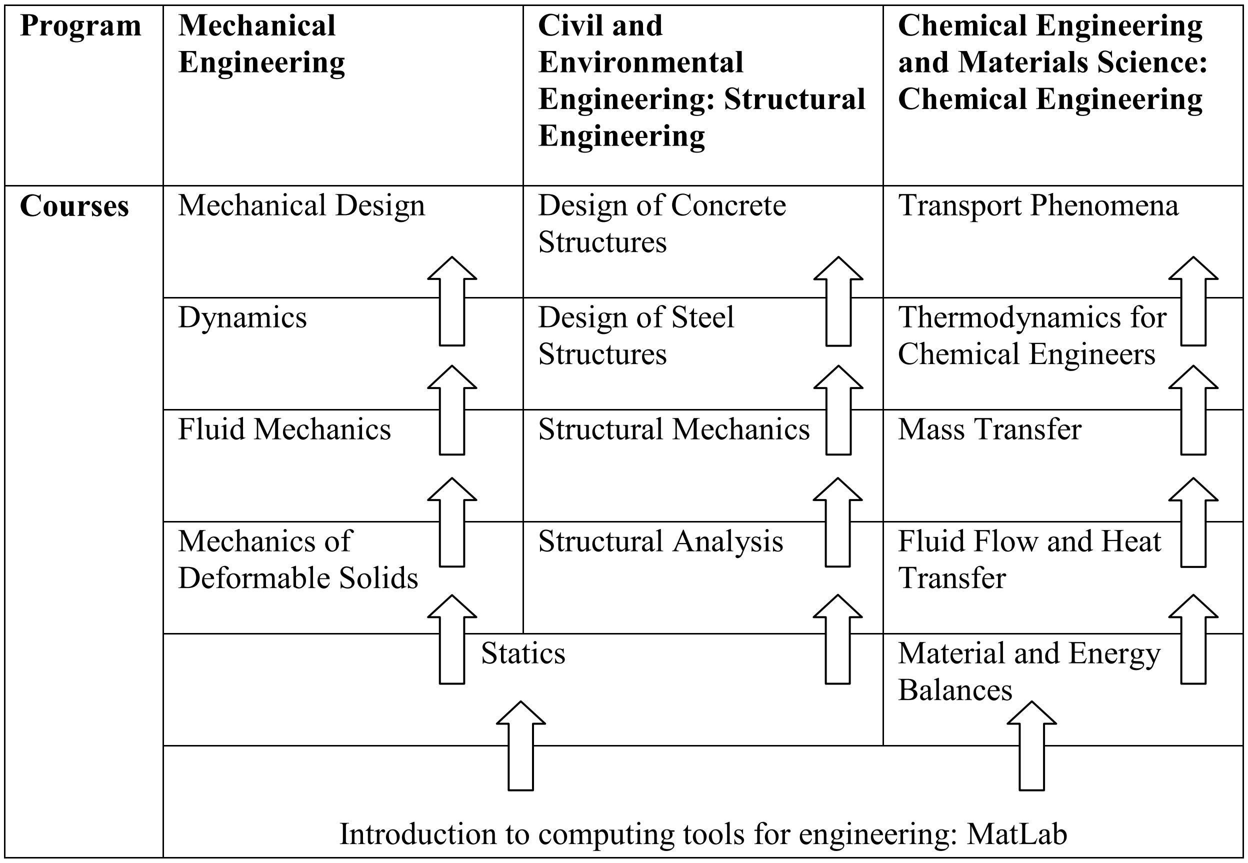
The confluence of the versatility and complexity of modern computing environments with the propensity of faculty to select their "favorite tool" for assigned use in completing problem sets and projects leaves undergraduate students in a largely untenable position. They are told to use a complex environment that either they have not touched before, or one that they have not used for several years. The result is that students seek and use cookbook approaches to the particular assigned task they face without learning integrating principles for use of the target computational environment. Cookbook use precludes student growth in understanding how the tool may be used to advantage. One might suppose that once a student takes a freshman-level course emphasizing (e.g.,) MatLab, that the student understands the basic principles that underlay any computational environment. Then the transfer of understanding from having learned a first computational environment (e.g., MatLab) could be made to a new computational environment (e.g., Mathematica) with little student effort. In practice, this simply does not happen, and the learning literature makes it clear why. In order to support transfer from one context to another, a student must thoroughly understand and integrate knowledge at a conceptual level from the first context. In order to master even a working subset of MatLab, repeated use in different contexts is not only desirable but also mandatory.

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. Even with such narrowing of focus, the question remains: how can the necessary faculty consensus be achieved to undertake curricular change that affects the entire curriculum?

By focusing on multiple, pairwise linkages of strategic classes in engineering curricula, faculty are involved across disciplines within engineering. As a result, they will see connections between first-year courses and subsequent courses, and will intentionally and naturally build connected problem-based learning scenarios into their courses using MatLab. Given that engineering faculty are far more likely than faculty in other disciplines to rely on lecture as a primary instructional strategy and far less likely to utilize forms of active or collaborative instruction,[17] the inclusion of MatLab problem-based activities designed by teams of faculty represents a significant pedagogical change in line with recommendations made for systemic change, but not frequently adopted.

Figure 1 shows one possible set of pairwise linkages for MatLab across three programs in the College of Engineering. The introductory computing course is taken by first year students in all three programs and has a linkage to the introductory Statics course (taken by Mechanical Engineering and Civil and Environmental Engineering students) and the Material and Energy Balances course (taken by Chemical Engineering and Materials Science students.) The linkages we propose (shown by the arrows) are *hard linkages*, in that students from the computing course work with students in the other two courses on projects. The projects are authentic problems in the disciplines that require students to use MatLab to compute the solutions. These problems help motivate students by introducing them to the tools that they will use in subsequent courses.

As students progress through their curriculum, similar linkages are formed across courses. These can be *hard linkages*, with students working on cross-courses projects, or soft linkages, with students carrying forward project work from one course to the next. In all cases, the underlying them of computational tools that are used across the curriculum is maintained.



Program	**Mechanical Engineering**	**Civil and Environmental Engineering: Structural Engineering**	**Chemical Engineering and Materials Science: Chemical Engineering**
Courses	Mechanical Design	Design of Concrete Structures	Transport Phenomena
	Dynamics	Design of Steel Structures	Thermodynamics for Chemical Engineers
	Fluid Mechanics	Structural Mechanics	Mass Transfer
	Mechanics of Deformable Solids	Structural Analysis	Fluid Flow and Heat Transfer
	Statics		Material and Energy Balances
	Introduction to computing tools for engineering: MatLab		

Figure 1: Examples of linkages across courses in three programs

The cooperative relationship among faculty is an important aspect of curricular reform, and should enhance the potential for institutionalizing the reform efforts. 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*. It is also important that institutional leaders are supportive early in the change process[18] in order to accommodate scheduling and enrollment in those courses faculty identify as appropriate pairwise linkages, to provide resource and infrastructure support, to insure appropriate inclusion of activities in reward and evaluation structures for participating faculty, and to facilitate inclusion of additional faculty in the future. These areas of departmental and college influence have all been shown to be critical factors in institutionalizing curricular reform and creating systemic change.[9]

Preliminary baseline data

We sought to establish baseline data using a team approach to teaching MatLab in the introductory computing course for engineers taught by one of the authors (Sticklen) during summer, 2003. 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, of 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% can see the importance of teamwork; however, just over half (52%) believe that they understand teamwork better after the experience. The believe that the teamwork helped them learn MatLab (78%) and are confident that they can figure out how to perform tasks in MatLab that they were not explicitly taught in the course: 78% believe that they could implement a FOR loop. There is a moderate correlation ($r=.52$, $p< .005$) between these items. This indicates 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 are less inclined to make these connections. Sixty-three percent of the students do not see that MatLab could be important to their later courses and 56% disagree that learning MatLab will help them learn other subjects. Furthermore, these two items are highly correlated ($r=.67$, $p< .0001$) with each other and are 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. The data suggest that structuring the group exercises with a more formal emphasis on near and far transfer may have a beneficial impact.

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. 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 will develop pairwise linkages between strategic courses in the curricula of each department. A substantial task is to develop a working taxonomy of course linkage types and to determine factors that favor one type of course linkage over another in terms of promoting systemic curricular change. We do not expect all pairwise course linkages to be of the same nature, nor do we currently know the complete set of linkages from which we will draw. We will identify *highly coupled linkages* and *loosely coupled linkages*. An example of a *highly coupled course linkage (hard linkage)* would be a course pair in which students in the two courses work together in a collaborative team setting on (for example) a common term project. An example of a *loosely coupled course linkage (soft linkage)* would be two courses in which subject matter of the later course assumes and directly utilizes conceptual topics from the earlier course.

Our second goal is to assess the feasibility of achieving meaningful systemic curricular change through a bottom-up evolutionary process grounded on pairwise course linkages. Our approach to addressing the need for systemic curricular change in general, and for the integration of a single computational tool in particular, is different in two ways from the typical top-down approach. First, the top-down approach builds a strong faculty consensus for change by appealing logically and persuasively to the body of pedagogical literature indicating change is needed, then attempts to undertake a wholesale rewriting of the target disciplinary curriculum. Building faculty consensus for such wholesale change, and then maintaining the consensus throughout the change process, has been difficult even in the context of the NSF consortia, and has not been widely emulated in engineering colleges outside the context of the consortia. Our approach will focus on *evolutionary* curricular change by sequentially implementing change in a number of pairwise linkages across the curriculum. An important corollary to our thesis above is that faculty consensus for change is more easily constructed and maintained if results of incremental steps for change are available *during* the change process. In an engineering sense, the system we are changing has internal positive feedback favoring change.

Our project will assess a path to systemic curricular change based on a two-fold decomposition of the total problem: the specific target of change (the vertical slice) and addressing the specific target by developing pairwise linkages one at a time that span the curriculum. In essence, we decompose the problem of systemic change along two dimensions: 1. **engineered systemic curricular change by major integrating concepts** (by addressing only one major vertical slice at a time); and 2. **engineered systemic curricular change over time** (by implementing additional pairwise linkages following a specified timeline).

Finally, as we continue this project, we will explore the somewhat orthogonal but important issue of determining the learning relation between computational tools for problem solving and disciplinary domain knowledge. 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 a student develop a sound and usable knowledge structure for understanding in her disciplinary domain.

Bibliography

[1] National Research Council (1991). Improving engineering design: Designing for competitive advantage. Washington, D.C., National Academy Press.

[2] National Science Foundation (1996). Shaping the future: New expectations for undergraduate education in science, mathematics, engineering, and technology. Washington, D.C., National Science Foundation.

[3] Abet (2000).

[4] Terenzini, P. , et al (1996). "Students' out-of-class experiences and their influence on learning and cognitive development: A literature review." Journal of College Student Development 37(2): 149-162 (EJ 527219).

[5] Bordogna, J. (1997). Making connections: The role of engineers and engineering education.

[6] Bordogna, J., E. Fromm, et al. (1993). "Engineering education: Innovation through integration." Journal of Engineering Education: 3-8.

[7] Hall, S. R., I. Waitz, et al. (2002). Adoption of active learning in a lecture-based engineering class. Frontiers in Education, Boston, MA, IEEE.

[8] Witt, H. J., J. R. Alabart, et al. (2002). Development of coaching competencies in students through a project-based cooperative learning approach. Frontiers in Education, Boston, MA, IEEE.

[9] Fisher, P., J. Fairweather, & Amey, M. J. (2002). EC2000 and organizational learning: Rethinking the Faculty and Institutional Support criteria. Annual Meeting of the American Society for Engineering Education, Montreal.

[10] Senge, P. and Associates (1990). The Fifth Discipline: The Art and Practice of the Learning Organization. New York, Doubleday.

[11] Moore, K. M., J. Fairweather, et al. (2000). NSF Report...Best Practices for Reform in Undergraduate Education in Science, Math, Engineering, and Technology: A Knowledge Framework. East Lansing, MI, Center for the Study of Advanced Learning Systems, Michigan State University.

[12] Tornatsky, L. and M. Fleisher (1991). The process of technological innovation. Lexington, MA, Lexington Books.

[13] Eiseman, J. and J. Fairweather (1996). Evaluation of the National Science Foundation Undergraduate Course and Curricular Development Program: Final Report. Washington, D.C., SRI International.

[14] Hargrove, S. (1996). Business and engineering interdisciplinary design. ABET Annual Meeting.

[15] Swart, W. (1996). Transforming engineering education at NJIT. ABET Annual Meeting.

[16] Trevisan, M., D. Davis, et al. (1996). Meeting the challenges of ABET 2000: Defining and measuring deisgn competencies. ABET Annual Meeting.

[17] Fairweather, J. (2002). "The mythologies of faculty productivity: Implications for institutional policy and decision making." Journal of Higher Education 73: 26-48.

[18] Druger, M. (1997). "Reform in undergraduate science education." Journal of Natural Resource and Life Science Education 26(4-5).

Author Biographies

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.

Marilyn Amey is Associate Professor in the Department of Educational Administration and Chair of the Higher, Adult, and Lifelong Education Program at Michigan State University. 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.

Jon Sticklen is an Associate Professor in the MSU 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.”

Timothy Hinds is an academic specialist in the MSU Department of Mechanical Engineering. He teaches undergraduate courses in machine design and statics as well as advises senior engineering student teams working on industrially sponsored capstone design projects. He also teaches a senior-level undergraduate international design project course and has taught graduate-level courses in innovation and technology management.

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.