AC 2009-1209: INTRACOURSES: SYNERGIES IN COMBINING TWO COURSES TO MAKE A NEW ONE

Eugene Ressler, United States Military Academy
Dr. Eugene Ressler is an Army Colonel, Professor of Computer Science, and head of the Department of Electrical Engineering and Computer Science at the United States Military Academy at West Point, New York. His research interests include neural networks for signal processing and software engineering as well as engineering education.

Michael Miller, United States Military Academy
Dr. Michael Miller is an Assistant Professor in the computer science program of the Department of Electrical Engineering and Computer Science at the United States Military Academy. In addition to engineering education, his research interests include simulation and software engineering.

Christa Chewar, United States Military Academy
Dr. Christa Chewar is an Army Major and an Assistant Professor in the computer science program of the Department of Electrical Engineering and Computer Science at the United States Military Academy, currently serving as an engineer on a major software project in Virginia. Her research interests include human-computer interfaces in addition to engineering and computer science education.

Jean Blair, United States Military Academy
Dr. Jean Blair is a Professor of Computer Science and director of the computer science program of the Department of Electrical Engineering and Computer Science at the United States Military Academy at West Point, New York. Her research interests include graph theory and algorithms in addition to education in computer science and engineering.
Abstract

We describe a new tool for curriculum design. By carefully choosing two traditional course subject areas that have a disciplinary thread in common, trimming both to their essential core topics using program outcomes as a guide, then combining the results, we create an “intracourse.” We give criteria for evaluating potential intracourse constituent pairs. We discuss alternative approaches to realizing the combination. Intracourses can be used to address several difficult curriculum design challenges. Rapid technological advances routinely create demands for new technical competencies within fixed engineering curriculum boundaries. Current trends toward increasing general education requirements reduce available time and other resources for specialized engineering courses. Intracourses allow for novel new curriculum design solutions in such constrained environments. Each intracourse also provides engineering students with direct experience in exploring the boundary between two traditional disciplines. Our methodology for formulating intracourses naturally provides measurement opportunities for assessment of program outcomes. Thus, a single intracourse can be designed to accomplish more than the sum of its parts. We give examples of intracourses formulated over the last several years along with successes and pitfalls encountered.

1. Introduction—Change In Engineering Programs

Relentless change characterizes undergraduate curricula in the engineering disciplines that are subject to rapid technology advances. Entirely new technologies appear regularly. Moore’s Law—the empirical observation that the number of transistors that fit in a given chip area roughly doubles every two years—leads to dramatic shifts in basic assumptions. What was impossible only a few years ago becomes routine and therefore newly accessible to undergraduates. What was routine becomes a commodity and therefore perhaps not worthy of study. We need look no farther than Global Positioning System technology for a current example.

The educational environment also exerts forces for change. Modern assessment and continuous improvement processes allow constituents of academic programs to state their needs and wants directly and explicitly. While immensely constructive, these inputs only add to the long list of tasks confronting curriculum designers. Students also have their say. They deserve high quality, high value educational experiences, a fact underscored by ever-increasing costs of university education. Yet, perceptions of “high quality” and “high value” are changing incessantly. What was recently a cutting-edge and relevant topic or exercise easily becomes old hat in a new context. Engineering and technology sector employment prospects shift as well, compelling the academic programs that serve them to follow suit. The contemporary need to make engineering appealing to young people, especially to women and minority students, only adds new change forces to this already complicated mixture.
If change is the hallmark of high technology disciplines, then that same change is itself characterized by fixed *constraints* and *drags* that render it extremely difficult to realize.

Perhaps the most important constraint on curriculum is time available. Our experience is that external forces invariably are toward *adding* topics, courses, and other new student opportunities. We rarely receive a recommendation to stop offering something already part of our programs. Yet there is little room to add new courses and other requirements to a typical undergraduate engineering major. In fact, the recent shift toward increasing core and general education requirements, while arguably the right course of action, is tending to decrease time for engineering coursework. Likewise, there is a fixed limit on the number and types of teaching and learning technologies that students can be asked to absorb before the enduring principles they are meant to illustrate become lost in a blizzard of factoids about perishable tools and techniques that have little long-term value.

Various drags also combine to make curricular change difficult. Sources of drag include the logistics of laboratory technologies. These tend to be expensive in dollars, with price magnified by the number of copies needed in the teaching environment. Another expense is the time spent by faculty and technicians to install and make new technologies functional for teaching purposes, including time to learn new tools and techniques and to address in advance the problems that students are likely to encounter. The latter costs are particularly hard to quantify and articulate for purposes of resource allocation within the university, since there are few parallels in other disciplines. Curriculum integration also induces drag, formally encoded in prerequisite structures. Engineering knowledge has many sequential dependencies. Therefore, an ill-considered change early in a curriculum can have unexpected, dire consequences for later learning. Faculty expertise may also be a drag. Whether a desirable new topic must be learned by existing faculty or else new faculty hired, progress must wait for these to be completed.

### 2. The Rationale For An Engineering Approach

In this setting of continual demands for change—to always add and never subtract, to overcome cost-induced drag, and yet to accomplish all within fixed constraints on time and other resources—we have a real engineering problem, one befitting real design tools and techniques. This note proposes one such tool—intracourses—that has proven useful in several settings. The concept is simple, yet requires care in implementation. We identify pairs of traditional and/or existing courses that have a common curricular thread or interface and combine them in a disciplined manner, creating curricular space for new and different outcomes to be achieved, while presenting a path for outcomes that have grown less relevant to fall away.

The rationale for this approach is simple. Short of wholesale curriculum revision\(^1\), there are only two possible levels of curricular change: traditional individual course level change and change involving only a small number of related courses. Merely adding new requirements to existing courses is pedagogically unsound. Past a distinct threshold, students are likely to learn less, not more, if syllabi are merely packed with more learning experiences. Teaching faculty placed in a situation of unreasonable expectations for student achievement are unlikely to find professional reward.
Moreover, we have found that deleting entire courses and replacing them with new ones is seldom desirable. Mature courses represent a substantial investment in assessment and continuous improvement with respect to the program outcomes they are designed to achieve. Faculty members are often heavily invested in courses at the personal professional level. Since even rapid change is usually analog and not digital, only rarely should all the outcome support provided by a given course disappear simultaneously. Rather, the tendency is for portions of courses to become less relevant over time, while other portions remain important. In some cases, we have retained courses for years that had only a minority of topics supporting program outcomes. We did so because these courses provided the only support for those particular outcomes. Consequently, we have repeatedly encountered needs to revise program content at a unit smaller than complete courses. The only way forward appears to be to give up the habitual assumption that longstanding course subject areas are indivisible.

Our approach in this paper is simply to manage curricular revision and incorporate new topics by focusing on half-courses. In the most common case, we trim away all but the absolutely essential in each of two courses and then combine them to form a new and unified whole, which we call an intracourse.

The remainder of this paper is about intracourses. Section 3 describes a four-part methodology for creating them. Section 4 gives several examples already in existence. Section 5 describes side benefits and synergies of the intracourse approach. Section 6 enumerates disadvantages and pitfalls. Finally, Section 7 draws conclusions and invites further study.

3. Methodology

In our introduction, we argued that an engineering approach to curricular change management is a natural fit for the problem and observed that making decisions at the level of entire courses can be inefficient and disruptive. We turn now to a method for making and implementing decisions at the level of half-courses. Our approach is to consider pairs of traditional course areas for combination into one course spanning both areas—an intracourse. We will describe our approach as a four-step design sequence: 1) Choose a course pair, 2) Prune content in an organized, objective manner, 3) Combine content to form the intracourse, 4) Assess and improve.

3.1. Choose

There are two ways to obtain intracourse topic pairs. If we have a desirable new area to add to our program but do not have room in the curriculum for a new course, we look for an existing course to pair with the new area in order to form an intracourse that is half new and half based on the existing course. Alternatively, the new topic may need a full course weight to accomplish new outcomes. In this case, we look for two existing courses to form a single intracourse, creating the needed full-course space.

The process of identifying courses that are feasible to combine to form an intracourse should logically be rooted in an analysis of program outcomes or—more likely—components of outcomes or the methods of achieving them. Thus, we begin by determining in advance what
aspects of a program are least valuable for reaching its desired outcomes. These are normally related to:

- **Fading technologies.** For example, transistor-transistor logic was for many years an excellent educational technology for understanding digital logic principles. It is now largely irrelevant, having been replaced by programmable devices.

- **Moore’s law effects that require different educational approaches.** In light of geometric increases in processor power and quality of compliers, the general importance of assembly- and machine-level programming skill has declined.

- **Silence from constituents.** When advisors are queried for the important aspects of a program, those not mentioned should be examined for topics that can be usefully set aside.

- **Niche topics not directly applicable to the program's constituency.** When the particular disciplinary focus of a course is only a means for achieving a specific set of learning experiences, it may be possible to achieve the same learning experiences using an altogether different sub-discipline.

### 3.1.1 Feasible courses

The key information needed to systematically make an informed choice lies in an outcome support matrix including each relevant course topic. These are often available in existing program assessment and improvement documentation. The matrix shows, in some manner, how program outcomes are supported by course topics and measurement instruments. A conceptual matrix (many forms are in actual use) is shown in Figure 1, along with a notional scoring system:

- 1 – Provide some level of support. Loss of topic would not endanger outcome accomplishment.
- 2 – Provides substantial support. Loss would have to be replaced in some manner elsewhere.
- 3 – Provides essential support. This topic contributes directly and fundamentally to outcomes accomplishment. Loss could endanger program success.

<table>
<thead>
<tr>
<th>Course 1</th>
<th>Topic 1-1</th>
<th>1</th>
<th></th>
<th></th>
<th>(\ldots)</th>
<th>(\text{Weak supporter of outcomes})</th>
</tr>
</thead>
<tbody>
<tr>
<td>Course 2</td>
<td>Topic 2-1</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>(\ldots)</td>
<td>(\text{Other strong outcome support})</td>
</tr>
<tr>
<td></td>
<td>Topic 2-2</td>
<td></td>
<td>2</td>
<td>2</td>
<td>(\ldots)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Topic 2-3</td>
<td></td>
<td></td>
<td>3</td>
<td>(\ldots)</td>
<td></td>
</tr>
<tr>
<td>Course 3</td>
<td>Topic 3-1</td>
<td>3</td>
<td></td>
<td></td>
<td>(\ldots)</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 1. Course topic support for program outcomes.**

We now focus on topic rows with small scores and also those with larger scores, but, where there is strong coverage for the corresponding outcomes elsewhere in the program. The courses housing these topics are the best candidates for combination in an intracourse. We have shaded some candidate rows based only on the partial information shown.
3.1.2 Feasible pairs

If we are looking for an existing course pair, then we proceed to consider all course pairs, seeking the pairs in which each course contains a significant fraction of lower priority topics, i.e. like those shaded in Figure 1. Combining the two courses in such a pair and then eliminating the union of lower priority topics creates the possibility of a new intracourse. Conceptually, we are marking a triangular “feasible pairs” table, such as the one shown in Figure 2.

```
<table>
<thead>
<tr>
<th>Course 1</th>
<th>Course 2</th>
<th>Course 3</th>
<th>Course 4</th>
<th>Course 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Course 2</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Course 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Course 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Course 5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Course 6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Figure 2. Feasible intracourse pair matrix for six courses.

For example, Figure 1 tentatively identifies the course pair 1-2 as worthy of consideration because both courses contain lower priority (shaded) topics. We have marked the corresponding matrix entry in Figure 2.

If we are only looking for a half-course topic space, then we consider Figure 1 alone, looking for existing courses where roughly half the time is dedicated to topics that have low priorities relative to program outcomes or are bolstered by redundant support elsewhere in the curriculum.

3.1.3 Pedagogical Soundness

Support for program outcomes only serves to identify feasible courses for pairing. The pedagogical soundness of combining a given pair is equally important. Perhaps the most important criterion is a common thread or interface based in the principal knowledge of the discipline. We consider a thread to be a logical or cognitive connection that spans both course areas. An interface, on the other hand, is a condition where learning in one course area supports the development of the other. This support may be mutual—each supports the other. An example might be a pair involving related theoretical and practical treatments of the same phenomena. Support may also be sequential, with the full development of one area leading logically to the beginning of the other. Such sequential courses might be more closely integrated by linking common elements across multiple courses to reduce compartmentalization and provide an integrative experience.\(^2\) We normally find such interfaces represented by prerequisite structures. Indeed, prerequisite pairs are likely targets for intracourses.

Logistical considerations also play a role. Availability and cost of textbooks may be important. If the course areas involve different kinds of laboratory space, then laboratory scheduling must be considered.
3.2. Prune

The Figure 1 analysis is also the basis for creating a list of new topics to be covered by the intracourse. We use the effect on program outcomes to make prudent decisions about what ought to be discarded. It is important that the designer use all available experience to develop a final level and amount of course content consistent with the credit-hours available. When complete, old topic lists must be replaced by the pruned (intracourse) list in the Figure 1 matrix and adequate outcome support re-verified.

3.3. Combine

The intracourse is complete when a new syllabus has been designed that smoothly incorporates the pruned list of course topics. Because we are typically dealing with course subject areas that are not normally treated together, this may require creativity. One strategy is to draw the topics as nodes in a graph, where arcs in the graph correspond to logical dependencies among topics. An arc from A to B means that topic A must be learned before B is possible. Now, any valid topological sort (extension of the dependency graph to a total order), is valid order to cover topics in the course.

We have encountered several patterns of dependencies. In some cases, the graph dictates that the only valid approach is to treat the original course topics as units. This situation is shown in Figure 3a. The white nodes represent topics drawn from one course and the gray nodes topics from the other. The dependencies require all the white topics to precede the gray ones. This often occurs when a curricular interface (rather than a thread) connects the original courses: one starts where the other finishes. Another common pattern is a series logical “blocks” from both original courses that may be alternated throughout the intracourse. This is shown in Figure 3b. We have encountered this pattern when combining courses treated, respectively, engineering science and engineering design using the same technologies. Finally, some course pairings allow coverage of topics from both original courses either in parallel or in very quick alternation. This is shown in Figure 3c. Combining a theory-oriented course with a practical one in the same area may lead to this result.

Figure 3. Samples of topic dependency patterns.

It is noteworthy that when blocks of the original course syllabi are retained intact in the intracourse, work for the course designer and instructor is generally reduced because existing, mature course content is often re-usable. For this reason, we suggest this approach as the normal starting point for intracourse syllabus design. However, with subsequent offerings of the
intracourse the blocks should become more naturally integrated. This more detailed integration ought to result from systematic assessment.

3. 4. Assess and improve

The method we have described naturally provides the framework for the assessment of a new intracourse because it begins with assessment information. The intracourse is developed by establishing its desired contributions to program outcomes from the start. It remains only to inject appropriate embedded indicators—specific points of evaluation with rubrics defining levels of achievement—to determine where the design effort has been successful and where improvement is required. Typical results of systematic assessment include modifications to enhance topic integration and course continuity, changes in pace and level of coverage to match time available, and discovery of unintended consequences, both favorable and unfavorable.

4. Examples
Following are several examples of intracourses that have been used to solve curricular problems at our institution during the last nine years.

4.1 Language-Based Simulation. Our computer science program required, for over 12 years, required compiler design course in the second semester of the senior year as an integrative experience for earlier-covered topics: computer architecture, machine and assembly languages, algorithms, data structures, formal languages and automata, elementary software engineering, and programming of substantial systems. It also introduced semantic attribute evaluation. A new constituent requirement arose to add discrete event simulation to the major. An analysis similar to the one discussed here revealed that program outcomes could be best accomplished by constructing an intracourse based on a semester-long project to build a simulation language processor. The intracourse retained its integrated treatment of algorithms, data structures, formal languages and automata, elementary software engineering, and programming of substantial systems. It also incorporated new outcomes from discrete event simulation concerning probability and statistics, simulation modeling, event processing, and results analysis. From the compiler course, we eliminated code generation, code optimization, bottom-up parsing (we retained the top-down techniques), and parser error recovery.

In the first offering of the course, students mastered the reduced list of topics from the original compiler course at least as well as in the original course. They also mastered the essentials of discrete event simulation mathematics and were able to produce simple simulation programs of their own. When working with the large system that is the common thread through the course, there was evidence of need for deeper understanding of the way that abstract syntax serves to represent the simulation model during execution. This area is the subject of ongoing course development.

4.2 Design Patterns and Engineering Intermediate-Level Systems. In this software engineering-based example we combined an advanced programming course that provided a significant and complete software engineering design-build-test experience with essential components of a design patterns course. The result is an object oriented design course providing
exposure to and practice with event-driven programming and design patterns, but also including
a significant software engineering design-build-test experience.

The course being replaced, *Advanced Programming*, covered advanced programming concepts
including polymorphism, dynamic dispatching, and concurrency. It also provided, at the
individual student level, an initial opportunity for open-ended project work, requiring the full
software development experience of identifying and articulating requirements, designing a
solution to meet those requirements, designing an appropriate test plan, implementing the
solution from scratch, and validating its correctness using the test plan.

The original course and also the new intracourse that replaced it depend on two prerequisite
programming courses that use Ada 2005, a language that encourages excellent programming
habits, in order to impart a solid foundation in coding techniques, modular design, algorithm
development, and debugging. In addition, the second course uses data structures as a vehicle for
the students to internalize encapsulation and inheritance concepts. Students also were exposed to
polymorphism and overloading in the second course.

The new intracourse is called *Object Oriented Concepts*. It retains, from the Advanced
Programming course, coverage of polymorphism, dynamic dispatching, and concurrency. It also
retains the initial opportunity for open-ended project work and experience through the full
software design process. However, we decided to trade away the requirement to develop a
software solution from scratch, having determined, through analysis similar to that described in
Section 3.1.1, that this experience was already being provided. This created space for students to
learn to recognize common design patterns and solve larger problems through code reuse.

The added traditional *Design Patterns* course content includes basic design patterns, immersion
in the object-oriented paradigm, and a new (for ours students) programming language—Java.
The new language serves several purposes: it reinforces knowledge of object-oriented
programming through exposure to an object model significantly different from Ada’s, it provides
a large library organized on design pattern themes, and it instills confidence by allowing students
to discover that learning a second programming language is easy after a first is known. The new
intracourse design also makes use of the relatively new pair programming technique. By
establishing a set of rules that precludes the divide-and-conquer technique of traditional student
team work, the pair programming approach to the implementation stage of software development
allows students to learn from each other.

Though assessment of the results is still in progress, both subjective student feedback and
objective measurements of student accomplishment—including inter- and intra-collegiate design
competitions—are strong.

4.3 *Introduction to Computer Architecture*. For approximately 14 years, a course in
computer architecture and another in computer organization were offered within the same
department by different majors’ programs. Over time, the courses grew to overlap in topics by
approximately 65%. In order to reduce resource needs by one course between two different
majors programs, the courses were combined. Topics eliminated from the architecture course
included details of input-output and memory system operation and PC-specific architecture.
From the organization course, SPARC assembly and C language programming were eliminated. This intracourse has been effective for six years in both programs. It achieved the desired effect of reducing faculty workloads and simplifying curriculum management.

Since this effort has been successful, a similar one is in progress to examine three parallel courses on networks and data communications with the goal of reducing these to two intracourses or even one.

4.4 **Electronics and Power.** The main constituency of our graduates is the field Army, where electrical power production and distribution have recently arisen as important areas of expertise for our graduates. Concurrently, the importance of power conservation and “green technologies” has been growing in all forms of engineering. In order to address these new demands, we are examining our service course in fundamental electrical and electronic concepts in order to incorporate power-related topics. A multi-disciplinary workshop is being organized in accordance with intracourse design principles.

4.5 **Thermo-Fluids.** Lacking the space for both a thermodynamics and a fluid mechanics course in several engineering programs, we combined two related courses to create an intracourse called Thermo-Fluids, exploiting the common curricular thread of flow and the associated common mathematical techniques to connect the two subjects.

4.6 **Statics and Strength.** Similarly lacking space for traditional static force analysis and strength of materials courses, we combined the two to form a statics and strengths intracourse, based on the curricular interface of forces in a structural element: where statics ends with computation of forces, strength takes up the task of choosing materials and shape to withstand those forces.

4.7 **Physics-based computer graphics.** When a new need arose to add physical system simulation to an electives list, but no resources were available to teach a complete course, we analyzed a computer graphics course and decided to eliminate lower priority topics dealing with highly realistic rendering. This led to an intracourse based on the curricular thread of simulating a physical system and portraying it with real time, cartoon-quality animations. During this transition, computer game development courses—inadvertently quite similar to ours—became available in other curricula.

5. **Synergies and Side Benefits**

With two traditional course subject areas treated in one intracourse, students can benefit from the easy availability of a teacher to foster their discovery of relationships at the boundary between. Such relationships include those between theory and practice, those between model and experiment, and those between tool or technique and respective application. These relationships are the core of technology and system integration topics, which are difficult to teach, yet can occur naturally within well-designed intracourses.

Intracourses also hold opportunities for faculty development. In a typical scenario, the faculty members who teach the original, constituent courses form a team to construct the new
intracourse in its initial form. Ultimately, both faculty members are able to teach the intracourse because each becomes newly familiar with topics of the alternate course during the development process.

Faculty and student perceptions are also an important side benefit. By presenting a new mechanism for incremental curricular change in an efficient, well-ordered fashion that is responsive in a relatively short period to changes in environment, intracourses promote an engaging climate of vitality and currency.

The process of considering courses as candidates for an intracourse has side benefits derived from the careful look at each of the component courses. Intracourse development serves as a “weeding out” process. It encourages the authors to identify course components that are absolutely essential for achieving program outcomes and educational objectives, regardless of individual faculty preferences, the pedagogical excellence of existing course material, and similar considerations that are ancillary to sound curriculum design decisions. For the faculty who participate, it is an opportunity to clarify the roles of courses in their programs and maximize their levels of program support and integration. Finally, the decisions required during intracourse design are likely to illuminate any gaps or weaknesses in program educational outcomes—a key contribution to continuous program improvement.

6. Disadvantages and Pitfalls

Intracourses also carry some risk. There is a vast existing educational infrastructure to support courses defined in terms of conventional topic areas. Intracourses can currently do little better than adapt. Included are faculty members who experienced the conventional courses as students and consequently have an innate sense for teaching them. Textbooks are also important. It may be difficult or impossible or to find a coherent series of readings in the two intracourse constituent textbooks, especially when topics are covered alternately or in parallel. If new textbooks need to be chosen, then the benefits of re-using constituent course content is limited. Even when the original textbooks mesh well, it may not be possible to prescribe that students buy both for cost reasons. It is worth noting, however, that the recent trend toward modular textbooks purchased on a “per chapter” basis by students, aligns well with intracourses, at least in concept. To date, however, such textbooks seldom address engineering topics.

The difficulty of intracourse design must also be considered. Suitable intracourse constituents may not exist. Moreover, constituent pairs that seemed good at design time may not work well in practice; hence, the intracourse must be undone, a painful process. The thermo-fluids course discussed in the examples may ultimately fall in this category. Despite their common curricular thread of continuum mathematics, careful design, and the enthusiastic efforts of expert teachers, it has proved difficult for even some fine students to be successful in this course. Reassessment is in progress.

It is important that the title and course description of a new intracourse accurately convey to the reader the main theme of the course and level of depth of coverage. For example, if we were to give the intracourse described in example 4.1 the title “Compilers and Simulation,” it may seem to both students and future employers that the course is shallow because it is forcing two
normally deep courses into one. However, the actual course serves as a disciplinary integrating experience that in some programs would be considered the capstone course.

7. Conclusion

We have presented a case that rapid change is an integral part of engineering education in disciplines that involve technological advance, especially those driven by Moore’s law. We have proposed intracourses as a concept and a methodology for implementing curricular change in a well-ordered fashion that is more flexible than change implemented at the level of whole courses. We have given examples of intracourses that have been successful. We have also pointed out unavoidable risks and pitfalls.

Certainly there is nothing new in the idea that topics rendered less important by the passage of time must continuously be supplanted by new ones in order to maintain the quality of any college curriculum. Yet the rate of change in the engineering disciplines that concern this paper is extraordinarily high, perhaps uniquely so. Accommodating rapid change with an engineering approach is also nothing new. The contribution of this paper is to note the two simultaneously and apply the latter in the context of the former.

We expect to continue using intracourses as a change mechanism in our own curricula and recommend them to others confronted with similar challenges. Future work includes developing a formal assessment and evaluation process for determining the effectiveness of an intracourse move.

Bibliography