Introduction

This paper reports on a federally-supported effort to substantively rethink the undergraduate mechanical engineering curriculum at San Diego State University (SDSU; see: http://sdsu.edu). The three-year initiative, funded in 2002 by the US Department of Education's *Foundation for the Improvement of Post-Secondary Education* (FIPSE; see: http://www.ed.gov/programs/fipsecomp/index.html), reflects the spirit of reform that characterizes a growing number of universities and colleges focused on the engineering sciences. Restructuring has affected three courses in particular—a significant chunk of SDSU's mechanical engineering program (see: http://attila.sdsu.edu/mechanical/).

Faculty in SDSU's Dept. of Mechanical Engineering believe that students need to be immediately exposed to and have hands-on experience with the software design tools used by today's engineers. As students move through their individual programs of study, they become more adept with these tools *and* also learn the theoretical constructs that underlie their use. Graduates are thus both philosophically and practically prepared to succeed in the engineering workforce. The three revamped courses—all offered during students' first and second years of study—are critical in this process; by the time students enroll in the third of the three courses (a capstone entitled *Simulations of Physical Systems*, described in detail elsewhere in this paper), they are fully prepared to study the principles of mechanical engineering by creating their own simulation software such as the amusement park ride shown in Figure 1.

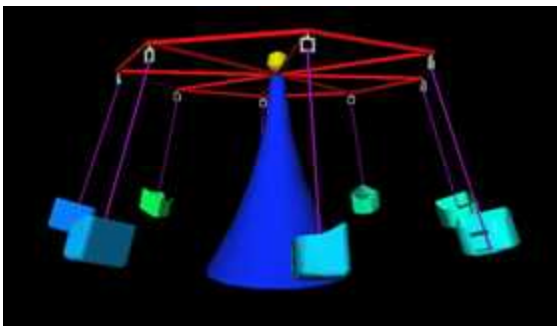

Figure 1: The Amusement Park Ride—A Typical Physics-based Virtual Machine

The Changing Face of Mechanical Engineering Programs

Those charged with educating engineers have long struggled with a number of instructional issues; core among them is how best to mix complex theory with hands-on

practice *and* engage those students (primarily women and underrepresented minorities) who tend to shun the profession (National Science Foundation, 1996; Seymour & Hewitt, 1997). High attrition rates—in particular, between the freshman and sophomore program years—can largely be attributed to the decontextualized nature of conventional instructional approaches, in particular, strategies overly focused on abstract concepts and tenets (Astin & Astin, 1992; National Science Foundation, 1996; Vetter, 1996).

In conceptualizing the new curriculum, SDSU's mechanical engineering faculty investigated innovative and energizing reform strategies that have been instituted in other higher-education programs since the early 1990s. They tend to be of two broad types: *total program reorganization* and *single course restructuring* . Each is exemplified below:

- Total program reorganization. In the mid-1990s, the University of Michigan (U-M) opted to significantly restructure its *entire* engineering curriculum (not merely update individual courses), using a comprehensive reform approach that many colleges/universities now emulate (Tilbury, Ceccio, & Tryggvason, 1997; Tryggvason, Thouless, Dutta, Ceccio, & Tilbury, 2001). U-M's Department of Mechanical Engineering (see: http://me.engin.umich.edu/)[1] opted for an inclusive process—one in which faculty, students, and alumni were equitably represented on a specially-appointed Review Committee. Needs assessment included student and alumni surveys (implemented in multiple years to ensure reliability), program reviews, and consultations with selected faculty. The alumni survey helped to establish typical career patterns (e.g., the industries to which graduates migrated as well as positions held and for what time frames), depict the skills deemed most important professionally, and determine where within the program key competencies tended to be highlighted or showcased (if at all). While the student version of the survey largely complemented the alumni form, it had a more concerted *course* focus (including perceptions of the "typical" course load and course preferences); it also featured items targeting students' future career/academic plans and expectations.

  Ultimately, the Review Committee recommended that a host of significant changes be instituted over a several-year period. Chief among them was content updating to ensure students were continuously exposed to cutting-edge ideas *and* had adequate access to advanced technologies; increased student experiences in teams and groups; greater

[1] Formerly, Mechanical Engineering and Applied Mechanics (MEAM).

emphasis on problem-solving and experiential learning; program flexibility (e.g., a larger elective pool to promote specialization); and new course sequencing. The process was so effective that *other departments* within M-U's College of Engineering adopted/adapted the reform strategies that Mechanical Engineering had employed.

- Course restructuring. Florida Tech, on the other hand (see: http://www.fit.edu/AcadRes/engsci/mechanic/mechanic.html), targeted students' initial orientation to the discipline (Larochelle, Engblom, & Gutierrez, 2003). By the late 1990s, faculty were convinced that the traditional overview course no longer sufficed; it had become overly technical and largely disconnected from the remaining program of study. The revised course was theoretically-grounded, reflecting a number of "hot-button" themes in the literature, e.g.: *fostering independent learning*; *developing leadership traits and know-how as well as good judgment*; *providing integrative, eclectic content*; and *scaffolding learning to ensure future success rather than a sink-or-swim mentality*.

  Freshmen now participate in a *cornerstone design experience* organized around 12 core outcomes that fall into several domains: *attitudinal/motivational*, *knowledge/skill*, *personal character*, and *higher-order processing*. The course is an action-oriented, lecture-lab hybrid—with students organized into teams tasked with *preparing proposals*, *generating design concepts*, *performing analyses, developing detailed production drawings*, *attending design reviews*, and *manufacturing functional physical prototypes* (p. 2).

  Course restructuring has led to improved design experiences in later courses (creative, relevant, well-structured). As important is a substantiated 15% improvement in retention from the freshman to sophomore years—and re-energized faculty.

But studious examination of novel reform strategies also led SDSU faculty to realize that change focused on instructional strategies—rather than holistic or targeted curricular reform—can be effective as well. Staff were particularly intrigued by a *learning styles* project led by the US Air Force Academy and the University of Texas at Austin; in this effort, results from three different instruments (Myers Brigs, VARK, and 6 Hats[2]) informed how faculty chose

[2] All three behavioral assessments target learning preferences, albeit differently. The Myers Briggs Type Indicator (MBTI) is organized around four "styles," specifically, the manner in which a person a) interacts with others; b) processes information; c) evaluates information, and d) comes to conclusions. Rather than being a diagnostic tool to determine learning preferences, the VARK Catalyst focuses on personal reflection. The 13-question assessment reveals how students prefer to receive and process

to take advantage of hands-on activities, interactive multimedia, and tools that build team dynamics (Jensen, Wood, & Wood, 2003). Their efforts have led to significant increases in student ratings for the affected classes, improved motivation and interest, and a unique team formation algorithm that dramatically enhanced group communications and interactions.

Finally, SDSU's mechanical engineering faculty were eager to respond to the findings of a National Science Foundation (NSF) Blue Ribbon Advisory Panel on Cyberinfrastructure (see: http://www.communitytechnology.org/nsf_ci_report/) whose members argued (in their 2003 report) that "… continuing progress in computing, information, and communication technologies [had made possible] a comprehensive cyberinfrastructure on which to a) build new types of scientific and engineering knowledge environments and b) pursue research in new ways and with increased efficacy" (2003, p. ES-2). While the Advisory Panel called on the NSF to establish and lead a large-scale ($1 billion in annual funding) cyberinfrastructure program, members also cautioned that the vision *could not move forward* without "… more broadly trained personnel with blended expertise in disciplinary science or engineering, mathematical and computational modeling, numerical methods, visualization, and the sociotechnical understanding about working in new grid or collaboratory organizations" (p. ES-3).

Thus, the curricular changes now institutionalized at SDSU build on, rather than merely replicate, the ideas that other reform-minded institutions (and the mechanical engineering programs within them) have embraced. As earlier noted, three courses have been radically revised, with more subtle changes occurring in the later classes that students complete (such as the senior capstone design experience). The following section examines reform at both the course and program level—reflecting an emphasis on *real-world problem* -solving; *learning by doing*; *student-centeredness*; *increased access to outside resources*; and *personal reflection* and *self-regulation*.

---

information; the four targeted areas include: visual (where information is depicted symbolically, via charts/graphs, arrow, flow charts, etc.); aural (where hearing is emphasized in information delivery/processing); read/write (where words are the vehicle by which information is shared); and kinesthetic (where the emphasis is on learning-by-doing). Six communication styles/roles are identified in the 6 Hats assessment—each associated with a particular color. Results are often used to form teams that are "balanced"—avoiding potential conflicts that tend to arise between/among certain "hats."

Deleted: NEED A "CATCHY" HEADING PHRASE

Program-level Restructuring: *Comprehensive Course Websites*

Although students enrolled in each of the three revised courses had always received syllabi at the start of each semester, they varied in depth/detail and overall quality. For the most part, the syllabi were traditionally structured, targeting the basics: *topics* to be covered from week to week, assignment due dates, exam dates, and required readings. They offered students a procedural rather than a conceptual roadmap.

By the end of the Fall 2004 semester, each course will feature its own comprehensive website, with content organized into modules that attend to processes and phases of development rather than topics to be covered weekly or per session (see, for example: http://attila.sdsu.edu/me295/). Modules themselves are structured around the ICARE format—developed in 1997 by professors in SDSU's Department of Educational Technology (see: http://et.sdsu.edu). ICARE emerged from a California State University (CSU) initiative designed to help faculty across the 23-campus system become proficient at teaching online (see: http://www.csus.edu/uccs/training/online/overview/t3.htm). A full explanation of the philosophy underlying ICARE (as well as details about each of its five core elements: *Introduction*, *Connect*, *Apply*, *Reflect*, *Extend*) is available online at: http://edweb.sdsu.edu/T3/Module2/Connect.htm#The%20I%20CARE%20System. What distinguishes ICARE from other well-established instructional planning systems[3] is its focus on student learning/performance—not what the instructor needs to do or consider. The ICARE framework helps students *manage their time*, *prioritize their tasks, process information more effectively*, *work more efficiency and successfully with others, independently explore ideas/concepts briefly introduced in class*, *assess their own progress*, *and revisit complex information* that may initially have seemed confusing or overwhelming. ICARE is *not* a learning management system (LMS) like Blackboard® or WebCT®; faculty (or those they designate) have full web design control—ensuring that the course structure fits learning outcomes and student needs. Faculty must also take responsibility for course management—selecting and/or formulating the tasks, activities, and assignments most appropriate for the instructional outcomes they want students to attain.

[3] See, for example: Reiser (http://www.fsu.edu/~ids/fac2002/session_id_2.html and/or http://www.fsu.edu/~ids/fac2002/lesson_delivery.htm), Hunter (http://www.huntington.edu/education/lessonplanning/Hunter.html), and Gagné (http://ide.ed.psu.edu/idde/9events.htm or http://tip.psychology.org/gagne.html).

Course-level Restructuring

Engineering 190: Graphic Communication and Virtual Reality. Engineering 190 (E-190), the first of several *required courses* in SDSU's Mechanical Engineering program, provides students with a solid grounding in the fundamentals of creating parametric solid models of parts, assemblies and drawings. Pro/Engineer® (commonly, ProE) which facilitates the creation of 3-D design objects, is the software application that students use most often for their assignments and projects[4].

E-190 unfolds in phases. In the first phase, students work closely with the instructors and student assistants to create simple solid models of basic geometry, small assemblies, and simplified drawings. Because ProE is complex and somewhat difficult to master, students are encouraged to actively explore its different features and functions—and learn from their mistakes. In the second phase, students create solid models featuring more complex geometry, larger assemblies, and drawings with notation that adhere to the ASME Y14.5M-1994 standard. Although instructor help is readily available, students must attempt to problem-solve on their own. And although they tend to work independently (at home or in the lab), they are strongly encouraged to help one another; the mood is *constructively competitive* , not adversarial.

Course resources include a brief ProE guidebook, commercially-available ProE job-aids (print and video), and instructor-created tutorials posted to the web.

Assessment is fully embedded throughout the course. Small-scale computer assignments allow the instructor to check for understanding without students worrying excessively about grades or points. Larger tasks are assessed more thoroughly, with students provided detailed feedback and suggestions for improvement. In-class quizzes call for students to produce models, assemblies and drawings within a given time frame. The final project requires students to independently create a moderately complex assembly and the 10-sheet standardized working drawing set for it.

Examples of various ProE machines that students created during the Fall 2003 and Spring 2004 semesters are presented below.

[4] In ProE, model features must not only be created in the proper order, but *all features must be fully defined* (completely dimensioned and referenced/located to other features). This requires students to conceptualize all of the features they create as well as where and how they are located on the model. This is in contrast to other solid-modeling software with more advanced GUIs and wizards that skip many design steps and do not generally require fully-defined features.

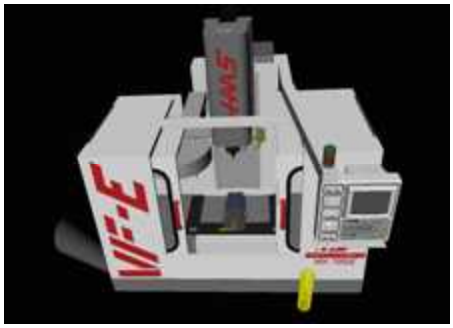

Figure 2a: Machining Center

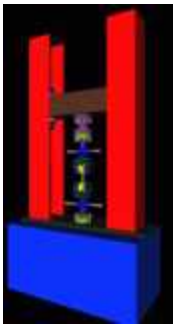

Figure 2b: Stress Testing Machine

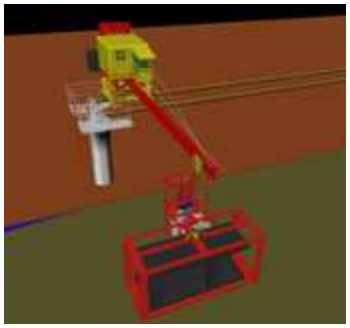

Figure 2c: Lifting Crane

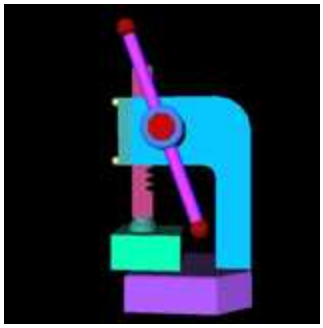

Figure 2d: Arbor Press

Mechanical Engineering 290: Computer Programming Applications. A computer programming course is required of SDSU's mechanical engineering students in their sophomore year. In traditional Mechanical Engineering programs, students enrolled in such a class exploit interpreted languages (e.g., Tcl, Perl, Python, or Matlab®) while studiously avoiding lower level compilable languages (e.g., FORTRAN or C). In Mechanical Engineering 290 (ME-290), however, this is exactly where the focus lies. The philosophical underpinnings for targeting compilable languages are several. Students who learn:

- a compilable language early in their academic careers become comfortable with the various levels of abstraction that constitute computer programming for a variety of purposes.
- programming early in their academic careers become adept at thinking logically and clearly, and develop critical "debugging" skills.

- how to implement solution algorithms of applied mathematics early in their academic careers are more able to understand how mathematics provides solutions to models of mechanical problems.
- how to communicate with CPU-equipped machines early in their academic careers are better prepared a) to embrace a world where mechanical machines can "think" and b) for the interdisciplinary skills that such a world requires.

The course is best described as lab punctuated with mini-lectures. Class sessions tend to open with field-relevant examples that orient students to basic concepts; then, while the information is fresh and support readily available, students work through programming tasks at their own workstations. Like E-190, ME-290 unfolds in phases.

- The first phase comprises one-third of the course (about five weeks of a standard 15-week semester) and features an overview of core (basic) topics, including data types, logical tests, control flow, loops, arrays, memory acquisition, and file creation.
- In the second phase, comprising about two-thirds (or 10 weeks) of a standard 15-week semester, each concept is covered in greater detail—though still at a fairly simple level. Traditional programming examples are purposefully avoided, replaced by more sophisticated and relevant ones, e.g., basic monte-carlo methods, numerical integration, roots of non-linear equations, matrix multiplication, and gauss reduction[5]. Students also learn about functions—allowing them to develop fairly complex/advanced programs *without* having to write an entire program from top to bottom.

Student progress is assessed through three exams and a series of increasingly challenging homework assignments. Daily quizzes administered during the first few weeks of the semester ensure that students stay on-task and focused, hone their text editing and debugging skills, and grow comfortable with operating system basics.

Students in this course are encouraged instead to *hammer* and *hack* at code building. This tactic fosters creativity and experimentation; reduces nonproductive competition between and among students; and builds confidence by negating the stereotypic notion that programming errors signify a person's academic weakness or inability. Within days, students realize that compiler and run-time errors are natural bi-products of programming—often trivial in nature and easy to fix.

---

[5] Note that students are exposed to the manifestations of these methods *prior* to their learning their theoretical underpinnings. This learning process reflects an instructional design strategy—*backwards design*—advocated by Wiggins and McTighe (1998).

Students have a number of resources to which they can turn as the course progresses, including Kernighan, Ritchie, & Ritchie's seminal text, *C Programming Language* (2nd ed.)[6]; the USEnet; and various websites on *C* programming. By the semester's end, students realize that competent mechanical engineers solve problems on their own but are never reluctant to questions of experts or colleagues who have successfully worked through similar issues.

Mechanical Engineering 295: Simulations of Physical Systems. Students enrolled in Mechanical Engineering 295 (ME-295), now familiar with both computer programming and computer-aided design, are ready for the challenge of learning how mechanical engineers "think." In this class, then, the focus is *conceptual integration* as students work in small groups to create 3D virtual machines that operate according to physical laws and run over the Internet. Each course phase has specific outcomes associated with it:

- *Phase 1: Data Acquisition*. LabVIEW® is the software of focus during this one or two-week segment. While the fundamental purpose of LabVIEW is machine communication, it is nonetheless a high-level interpreted language in which critical programmable constructs (e.g., data types, logical tests, control flow, loops, arrays, memory acquisition, and file creation) are represented graphically. In other words, there are graphical icons that represent the basic construct: loops, logical tests, data files, functions. One then connects these icons together and this wiring constitutes the program flow.

Figure 3a: Typical LabVIEW Module—External Control Panel

Figure 3a illustrates the external user interface for a typical LabVIEW program, while Figure 3b depicts the internal wiring.

[6] The Kernighan et al. text is extremely challenging for second-year students; ME-290 instructors tend to complement assigned readings with interpretive notes that offer simplified explanations and help students distinguish between *must know* (primary/core) and *nice to know* (secondary/peripheral) information.

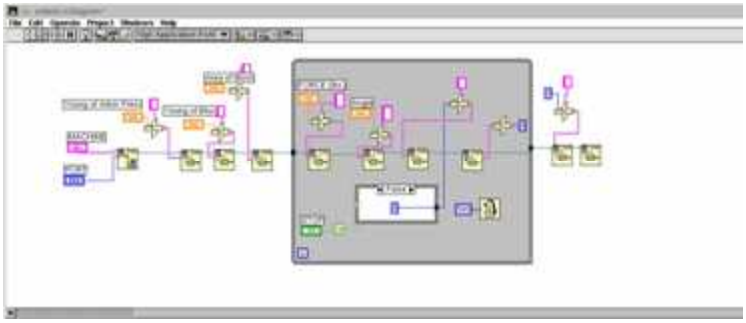

Figure 3b: Typical LabVIEW Module—Internal Wiring Connecting the Graphics Programming Icons

Once students complete an overview of the entire package, they turn to its specific features and functions—replicating the *general to specific* instructional process introduced in ME-290.

- *Phase 2: Instruction about the Internet*. The second phase of the course—accounting for about three-weeks of the semester—attends to the Internet and the ways in which students can move beyond passivity (i.e., *point and click*) and actively use the web to program data acquisition modules that control remove machines. Students learn the basic *C* interface functions that programs make to allow for communication between and among computers. They focus in particular on the fundamentals associated with both *server* and *client* tasks. Figure 4 presents a schematic of a physics server to which two clients connect: first, a LabVIEW data acquisition client, and then a 3D graphical visualization client.

Deleted: s

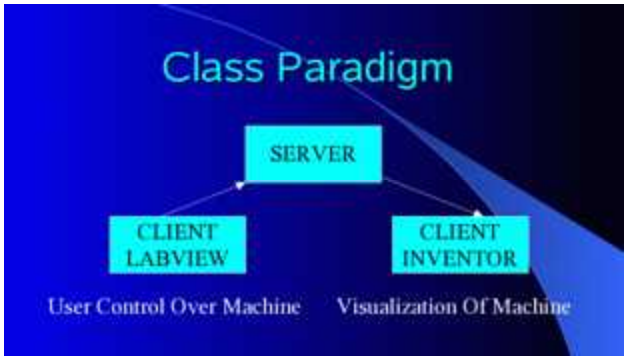

Figure 4: Network Paradigm Used in ME-295

Students emerge from this phase of the course with a more fundamental grasp of what the Internet is, how it works, and how it can be exploited to create distributed physics based machines. Their use of LabVIEW to communicate with/control their virtual machines fully mirrors how the software might control machines with on-board CPUs and unique IP addresses. In essence, they experience manipulating machines can "think."

- *Phase 3: Computer Graphics*. Here, students are introduced to the basic concepts of OpenGL®—the programming interface associated with nearly all computer graphics applications (including the very CAD tools they have used In E-190). About a week is spent on the history and evolution of computer graphics and the migration from procedural programming of data sets to the higher level object oriented interfaces that allow for 3D object manipulation. In essence, OpenGL is not studied in detail, but as a segue to standard scene graph technology.

- *Phase 4: Inventor*. Students are oriented to the concepts of the scene graph technique of Open Inventor®—a method that closely replicates how ProE registers objects (and so is familiar to them). Scene graphing begins with a fundamental object or node—for example, the geometry of a wheel. Students then equip the node with specific attributes such as position, color, orientation, and size. They soon realize that objects or nodes are *familial*, e.g., the wheel node can be owned by a car node (the *parent*)—which itself can own several wheels *(children)*.

  While more advanced scene graph viewers are available, Inventor is a simple one—well suited for beginning Mechanical Engineering students not yet ready to manage advanced nodes. An added plus is that in its coin version (see: http://www.coin3d.org), Inventor is freely downloadable.

  With Inventor programming skills, students can take an object made in ProE and export it into the standard Inventor format. In so doing, the object becomes a data set with part hierarchy and geometric coordinates that supercede a CAD package. Students are able to edit the data set, and manipulate the data representing a 3D object. They realize that while they may have created a complex assembly with the CAD package, it is nothing more than data points, graphics triangles, and coordinates.

  With additional Inventor skills students exploit the Inventor APIs to write a graphics program that reads the data set and displays it once again. They have now experienced *writing code* as opposed to *selecting options* from the menus of a commercially

Deleted: I'm confused here … what *phone* are we talking about? For me, this explanation is a bit abstract … and decontextualized. And where does the *internet* come into play??]¶

available product (ProE). With their own codes, students have animated their designs; they are able to move objects at will and arbitrarily deform.

- *Phase 5: System Integration* . This three-week culminating experience closes the semester. Students use a multi-step process to choose a machine[7]; reproduce it; analyze the physics of it; and create its distributed, virtual, physics-based version[8]. Instructor guidance is supplemented with *learn from experience* advice from *former students* eager to share their know-how.

  First, students must produce 3D CAD models *as well as* views, drawings and blueprints. Simply put, they create 3D free body diagrams of their selected machine to demonstrate their understanding of its mechanics. Figure 5, drawn from the amusement park ride earlier referenced, illustrates a 3D free body diagram that students created for one of the cars of the rotor. Specifically, the diagram relates the angular velocity of the ride to the lift of the cars.

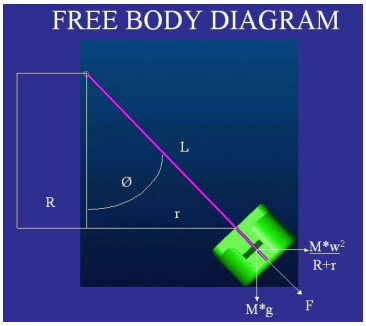

Figure 5: Free Body Diagram for Amusement Park Ride

Next, students implement their equations and write the computer program that governs the physics of the machine (based upon their free body diagrams). They then use LabVIEW to create the data acquisition control panel for their machine. Figure 6, also drawn from the amusement park ride example, depicts how students have input the material properties, e.g., the weight of the people in each car and the ride's angular velocity.

Deleted: and write a program to manipulate It. They learn to do this In the fifth phase of this class. Students study the following example. A beam and wall Is created In ProE and exported to Inventor. A labView Interface was created to position a load on the beam. A computer program was written to calculate the deflection of a beam under a load. A client server program was written so that the server (In this case, a physics code), allowed for two client connections: a LabView control client, and a 3D visualization client. Students used LabView to move a load across a beam. The data was delivered to the physics server which calculated the deflection and delivered the deformed data to the 3D Inventor visualization client. -- Too prescriptive/procedural; need to make this paragraph more about the *essence* of what you're doing … not recall of what you've done. This paper is about *ideas* not about week-to-week actions.¶

Deleted: I went back to your earlier depiction of the course, but had a hard time distinguishing the specific elements that comprise Phase 5. I'll need help here!

[7] Many students propose projects that are ill-defined, overly complicated, or simply inappropriate for the task at hand. About half are initially rejected.

[8] The length of each instructional phase tends to vary by project—reflecting each one's unique nature. Student progress is continuously monitored to ensure a manageable, balanced workload throughout the semester.

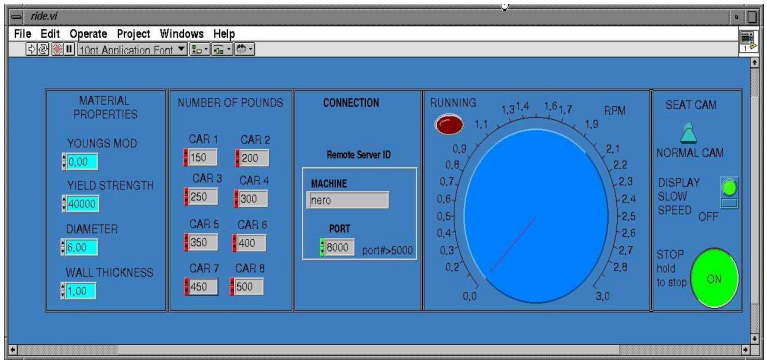

Figure 6: LabVIEW Control Panel for the Amusement Park Ride

At this point in the project, students have created a 3D CAD assembly of a machine and three separate computer programs (visualization, physics, and data acquisition). Now they must convert the physics program into a generalized physics *server* using the Berkeley Standard socket libraries as an interface to the network transmission control protocol. This server initiates the basic calls to open a socket and receives connections over the Internet from the two client programs—visualization and data acquisition (previously depicted in Figure 4).

Successful student groups realize that the three codes (visualization, physics, data acquisition) can run from different locations—but they must be able to communicate (with regard to number of bytes, data type, and order of data transmission). Thus, members strive for an equitable distribution of labor, while remaining cognizant of the work each of them is doing. Students are compelled to distinguish between the nature of input and output data. Internally, groups must agree on a) how best to arrange similar data in organized data frames for efficient transmission and b) the protocols of data transmission.

The final issue is now system speed. Students must make their programs run quickly if their machines are to function realistically. This last step forces them to re-analyze their entire conception, using creative problem-solving strategies to determine where speed can be improved. Simply put, they must "milk" the CPU for power. Their questions are several: Where is the bottleneck? To resolve it, should they simplify the mechanics? Ignore dynamics and focus on kinematics? Employ a faster solution scheme? If the problem lies in the visualization, should they go back and simplify the CAD model? If, instead, it is mathematical, should they search for a faster solution means?

Addressing all of these questions is a risk for the students, but one with powerful tradeoffs—excitement, collegiality, and the inner satisfaction of finding a solution that works.

As a last step, students produce data flow schematics that model the order of data transmission. For example, one project required the following initial and continuous flow of data:

From	To	Status	#bytes	Type	Function
LabVIEW	SERVER	Initialize	56	structure	Initialize physics
SERVER	Inventor	Initialize	32	doubles	Initialize geometry
LabView	SERVER	Main loop	8	double	Flow velocity
SERVER	Inventor	Main loop	8	double	MEM's plate position

Evaluation: Measuring the Impact of Curricular Reform

The curricular reform effort detailed above is continuously evaluated; assessment is, in fact, framed by a sophisticated and relevant methodology: Provus' Discrepancy Model (Steinmetz, 2000; Fitzpatrick, Sanders, & Worthen, 2004). This widely-accepted and well-respected framework is a mixed-methods approach attentive to continuous information management. Undergirding the model is a four-step process that calls for program/project staff to agree upon/establish clear program standards (objectives, outcomes); determine (via ongoing assessment strategies) whether or not discrepancies exist between *actual* and *anticipated* performance; determine the nature of any discrepancies as well as their severity and importance; and use detailed data about performance gaps to establish next steps (which may range from modifying or terminating selected program activities to revamping program marketing and/or recruitment techniques to revising assessment techniques).

Provus intended for evaluation to be both affirmative and constructive, not punitive. In that light, *discrepancies* are viewed positively. They ensure that program/project staff are continuously focused on goals and outcomes, and promote collaborative exchanges between/among critical stakeholders as issues are resolved. Discrepancy data have a "self-serving" nature as well, allowing for course or program improvements. In the present case, that may mean rethinking how students are organized, content is delivered, feedback is presented, course activities are sequenced, or facilitation materials are developed and rolled out to the field.

Provus argued that all programs/projects have life cycles:

- In the *definition* stage, staff define goals, processes, or activities, and then delineate the resources necessary to accomplish or complete them. *These definitions or expectations are the basis upon which ongoing evaluation depends.*
- In the *installation* stage, the definitions/expectations becomes the standards against which to judge operations. *The idea is to determine congruence—specifically, whether or not the program or project has been implemented as it was designed.*
- In the *process* (or enabling) stage, evaluation is characterized by data collection that attends to those the program/project serves. The idea is to look at *progress* to date, determining initial impact, influence, or effect.
- In the *product* stage, data collection and analysis help to determine the extent to which the program/project's *terminal objectives* (outcomes, goals) have been achieved. The expectation is to plan for follow-up (long-term) studies to determine lingering/ongoing impact.
- The (optional) c*ost-benefit* stage features opportunities to compare results with those achieved by other similar or analogous approaches.

Formatted: Bullets and Numbering

Pertinent results, or the ways in which data-gathering has led to changes in the original reform vision, are briefly described below.

- The team has developed a consistent method and a reliable process for ME-295 students to display their completed animations on the web. A graduate student in the Dept. of Educational Technology now offers a multi-session workshop (four events, each about an hour long) on web-page design and development. The ideas is to build on skills already covered in E-190—but tailored to the fairly unique requirements of ME-295 projects.
- An extended (and lively) discussion led team members to develop and implement an *informal* protocol for presenting final projects. Engineering students tend to be shy—even a bit introverted—and somewhat uncomfortable with key language arts skills (writing and speaking, in particular). However, information sharing—via specification sheets, technical reports, oral briefings, and mock-ups—is inherent to the work settings that await engineering graduates. Integrating final presentations into the process develops competencies that today's employers increasingly expect. As important, it builds confidence by encouraging students to reflect on their individual and group accomplishments; retrace their steps and articulate them in a way that others can understand; foster communications between and among team members; and build rapport with the instructor.

- Students contribute to curricular assessment by participating in end-of-course evaluation tailored to their instructional experiences. Prior to completing the traditional department surveys that faculty must administer (under provisions of the CSU/Faculty Memorandum of Understanding), students now access specially-tailored forms created under the auspices of the *Individual Development and Educational Assessment* (IDEA) Center (see: http://www.idea.ksu.edu/mission.html) at Kansas State University[9]. The surveys feature a variety of item types that IDEA staff analyze in light of course outcomes the instructors themselves identify and then distinguish by type (*essential*, *important*, *minor importance*). In the main, these instruments capture student perceptions of a) the instructor's teaching effectiveness, b) their own progress toward meeting course objectives; and c) the "relevance" of different classroom techniques the instructor may have used.
- The Dept. of Mechanical Engineering is now rigorously tracking year-by-year enrollment and retention rates (with a specific focus on women and underserved minorities), elective choices, years to graduation, semester-to-semester enrollment loads, and internship opportunities of which students take advantage. Faculty and staff believe such measures are as important to monitor and parse as student perceptions of instructional quality, facilitation, and real-world relevance.

Discussion and Summary

Curricular restructuring at SDSU unfolds in a setting that is continually in flux. Faculty efforts to reform the undergraduate Mechanical Engineering program have been both rewarding and challenging.

The Challenges

Like other universities (both private and public), SDSU has suffered through several lean budget years—and the 2004-05 and 2005-06 academic years look particularly bleak. The effects of troubled finances are dramatic and widespread—affecting, for example, the department's ability to hire both tenure-track and part-time faculty, the number of class

[9] The original IDEA forms were first used during the 1968-69 academic year in an effort to provide instructors with *comparative* course and personal performance data—results that could be constructively interpreted and lead to improved teaching and learning. In 1975, the IDEA Center was established—and the system became available to other institutions of higher education. Products are continuously updated and enhanced, reflecting advances in technology as well as ongoing research (in such areas as cognition, learning styles, receptivity to change and innovation, and teaching methodologies).

sections it can offer per semester, student fees, and equipment replacement and maintenance.

Another complication is that course content cannot remain static or constant. New software solutions are constantly available—offering attractive alternatives to what is already in place. For example, faculty hope to replace ProE with Coin, which can run on all platforms (Linux, Unix, Macintosh, Windows) and thus eliminate the need for SGI. Software decisions affect module design—their content, the activities and tasks associated with them, and their "placement" within the 15-week semester.

SDSU—like other universities—is quickly moving toward distributed course delivery. Within the next few years, this sequence will need to be restructured (or significantly modified) for online delivery—attracting students who may have language barriers, be anxious to move forward at their own pace (rather than follow a traditional week-to-week schedule), and live in several different time-zones (making team and small group difficult to organize, monitor, and manage). As important is that the team must balance precision and explicitness (a tendency to "script" the courses) with a "vagueness" that encourages faculty outside SDSU to adapt, modify, and personalize course elements to fit student needs and their own teaching styles.

Finally, it is difficult to anticipate how a course will unfold when enrollment dramatically scales up. For example, while the team can anticipate how teaching strategies must evolve to attend to the planned expansion of ME-295, members cannot fully foresee what lies ahead.

The Rewards

Despite budgetary limitations and the steady drum of progress, faculty have enthusiastically embraced the reform process—and support the changes already in place and on the drawing board.

- They recognize the importance of graduating workforce-ready (marketable) students who truly embrace the field *and* are technically and attitudinally prepared for the variety of professional options it offers.
- The new course designs allow them to work more intimately with students—making teaching both personally fulfilling and energizing. They celebrate individual and team successes, mentor students through frustrations, and witness emerging confidence and self-esteem. The mood is positive—collegial and supportive.

- They have taken advantage of the research opportunities that reform/restructuring provides. Faculty are, for example, more engaged in grant writing—actively seeking both independent and collaborative/cross-disciplinary opportunities.

Overall, then, curricular restructuring has led to improved student performance; a more progressive menu of courses; instructional flexibility—attuned to individual learning preferences, perceptions of personal relevance and responsive to technology innovations; and relationship-building between and among faculty that reflects a more entrepreneurial spirit and renewed interest in teaching.

References

Astin, A. W., & Astin, A. W. (1992). *Undergraduate science education: The impact of different college environments on the educational pipeline in the sciences.* Higher Education Research Institute: Graduate School of Education, University of California, Los Angeles.

Fitzpatrick, J. L., Sanders, J. R., & Worthen, B. R. (2004). *Program evaluation: Alternative approaches and practical guidelines* (3rd ed.). Boston, Pearson Education.

Jensen, D., Wood, K., & Wood, J. (2003). Hands-on activities, interactive multimedia, and improved team dynamics for enhancing mechanical engineering curriculum. *International Journal of Engineering Education, 19* (6), 874-884.

Larochelle, P. M., Engblom, J. J., & Gutierrez, H. (2003). An innovative introduction to mechanical engineering: A cornerstone design experience. Retrieved July 25, 2004, from the American Society of Mechanical Engineers/Curriculum Innovation awards website: http://www.asme.org/education/enged/awards/cia03/

National Science Foundation (1996). *Science and engineering degrees by race/ethnicity of recipients: 1987 -1994—Detailed statistical tables* . Washington, DC: Author. (NSF96-329)

Seymour, E., & Hewitt, N. M. (1997). *Talking about leaving: Why undergraduates leave the sciences.* Boulder: Westview Press.

Steinmetz, A. (2000). The discrepancy evaluation model. In D. L. Stufflebeam, G. F. Madaus, & T. Kellaghan (Eds.). *Evaluation models: Viewpoints on educational and human services evaluation* (pp. 127-144). Boston: Kluwer Academic Publishers.

Tilbury, D. M., Ceccio, S. L., & Tryggvason, G. (1997). Restructuring the undergraduate curriculum of the Mechanical Engineering and Applied Mechanics Department at the

University of Michigan. Paper presented at the annual meeting of the American Society for Engineering Education (ASEE), Milwaukee, WI.

Tryggvason, G., Thouless, M., Dutta, D., Ceccio, S. L., & Tilbury, D. M. (2001). The new mechanical engineering curriculum at the University of Michigan. *Journal of Engineering Education, 90* , 437-444.

Vetter, B. M. (1996). Myths and realities of women's progress in the sciences, mathematics, and engineering. In C. S. Davis, A. B. Ginorio, C. S. Hollenshead, B. B. Lazarus, P. M. Rayman, et al. (Eds), *The equity equation: Fostering the advancement of women in the sciences, mathematics and engineering* (pp. 29-56). San Francisco: Jossey-Bass Publishers.

Wiggins, G., & McTighe, J. (1998). *Understanding by design.* Alexandria, VA: Association for Supervision and Curriculum Development.

THOMAS IMPELLUSO
San Diego State University, Dept. of Mechanical Engineering, San Diego, CA 92182. USA. E -mail: impelluso@engineering.sdsu.edu

MARCIE BOBER
San Diego State University, Dept. of Educational Technology, San Diego, CA 92182. USA: Email: bober@mail.sdsu.edu