The Peer Reviews and the Programming Course

T. Grandon Gill
University of South Florida, Tampa, Florida USA

ggill@coba.usf.edu

Abstract

What would happen if a typical computer programming course were submitted for peer review by a research journal? Using a format inspired by typical peer reviews, issues relating to rigor, relevance and course design are raised. In these fictional reviews, weaknesses of the traditional "lecture and test" approach to teaching are identified in all three areas and it is argued that such weaknesses may be inherent to that design—particularly in teaching skill based courses. An alternative approach, whose inspiration is the techniques employed in U.S. submarine qualification, is the proposed and some initial results of teaching such a course are presented.

Keywords: teaching, computer programming, course design, management information systems, pedagogy

Introduction: The Cover Letter

To: Associate Editor, *Quarterly Journal of MIS Research*
From: Author 635
Re: Submission 616
Subj: Revised Submission

15 December 2005

Dear Editor:

Let me begin by apologizing for the 5 years it has taken me to get back to you with this revised submission. Addressing the reviewer comments proved more time consuming than I originally anticipated. Nonetheless, I believe that the revised manuscript—which is more like a complete rewrite than a revision—represents a sizeable improvement on the original.

The new manuscript is provided as a separate attachment. The present document contains a summary of my responses to the reviewer comments. Because of the time that has passed, I also felt it prudent to include the original reviewer comments as well, which precede my responses to them.

Thanks again for you patience. I look forward to hearing your reaction.

Respectfully,

Author 635

The Review Packet

To: Author 635
From: Associate Editor, *Quarterly Journal of MIS Research*
Re: Submission 616
Subj: Reviews of your manuscript

21 May 2000

Dear Author 635:

We have finally received all three reviews back for your manuscript titled "An Introductory Programming Course". As you can clearly see, the comments were highly critical of numerous aspects of the submission—with two out of the three recommending outright rejection. Rather than summarily dismissing your work, however, we have decided to offer you the opportunity to resubmit a completely revised version of the manuscript. Two factors motivate this decision. First, there seems to be quite a bit of inconsistency regarding the nature of the deficiencies in the described course identified by the reviewers. Second, as an entirely fictitious journal, we have a relatively limited number of submissions to choose from. You should be cautioned, however, that "low standards" are not the same as "no standards"—there is no guarantee that any decision you make to submit a revised manuscript will necessarily lead to a favorable outcome.

Attached are the forms filled out by the three reviewers, all of whom are acknowledged experts in the field (or the students of such experts). As you will see, each reviewer brings a somewhat different perspective on your manuscript. Reviewer 1 has raised a number of critical issues relating to the rigor of your protocol. Reviewer 2, on the other hand, seems most concerned about the manuscript's lack of topical relevance. Finally, Reviewer 3—who assigned the most favorable rating and provides specific guidance regarding how to proceed with such a revision—treats design issues as central.

Should you undertake a revision of the manuscript, please include a memorandum detailing how you addressed each of the substantive concerns of these reviewers. This document will be critical in evaluating whether or not the revised version will be accepted.

Regards,

Editor

Reviewer 1 Comments

Summary Ratings					
Relevance	1 - None	2 - Minimal	3 - Modest	**4 - Strong**	5 - Outstanding
Rigor	**1 - None**	2 - Minimal	3 - Modest	4 - Strong	5 - Outstanding
Recommendation					
Manuscript disposition:	**1 - Reject**	2 – Major revisions	3 – Accept, minor revisions	4 – Accept, as is	

Comments to author(s):

Summary: The paper presents the protocol of an introductory undergraduate programming course using the C++ programming language as a research investigation. While the relevance of such an investigation, properly conducted, would be high, it is doubtful that the present manuscript could ever achieve the level of rigor required for publication in QJMISR. It is therefore recommended that the manuscript be rejected.

*Detailed comments:*The manuscript is strongest with respect to relevance. Employment in the MIS field should experience rapid growth over the next decade, with MIS positions holding the top 7 spots in expected job growth for occupations requiring a bachelor's degree (Bureau of Labor Statistics, 2004). Since most MIS programs appear have at least one programming course (Gill and Hu, 1998), the topic should be applicable to most programs. Moreover, since members of the research community—the principal audience of QJMISR—typically teaches two or more courses a semester (Hu and Gill, 2000), there may be some peripheral interest in pedagogy-based research amongst this constituency as well.

Unfortunately, the course as described is riddled with methodological problems that suggest a complete absence of rigor. As a starting point, the entire course design is incomprehensible. The objective of the course, as stated in the formal design document (a.k.a., syllabus), is to “teach students the fundamentals of computer programming using the C++ programming language”. The operative word “teach” here implies a transfer of knowledge. Presumably, this should be measured by a change in subject (a.k.a. student) knowledge levels. Look at the design, however (sketched out below in Figure 1).

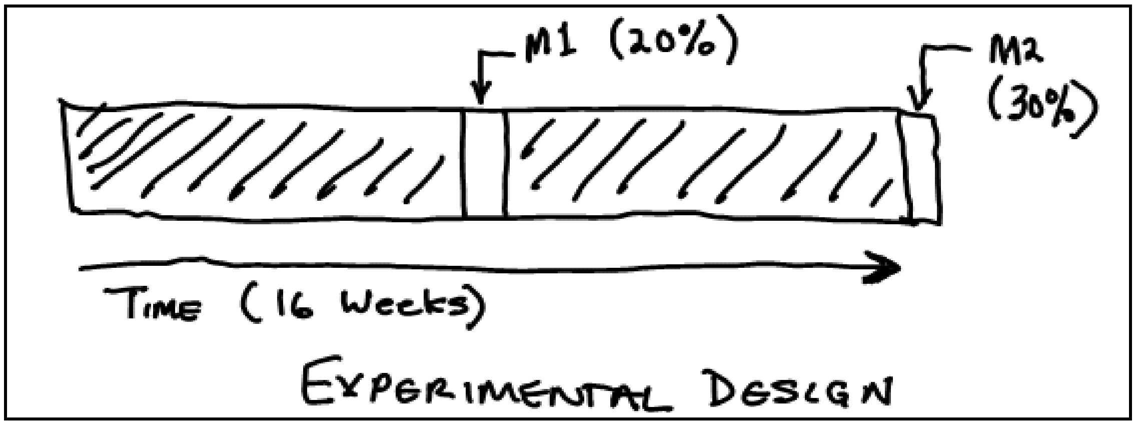

Figure 1: Course research design

This design is completely inconsistent with the proposed dependent variable in two ways. First, the initial dependent variable measure—referred to as the “midterm exam” (M1 in Figure 1)—does not take place until midway through the protocol. This might be excused if we could assume that all subjects come in with zero knowledge. Unfortunately, the literature on the subject reports this to be a very bad assumption, identifying variations in background as being the *most* serious obstacle faced in teaching introductory programming courses (Roberts, 2000). Indeed, with respect to the characteristics of the subject population, the manuscript itself reported a distribution of 51% (never taken a previous programming course), 49% (taken at least one previous programming course) and 25% (taken more than one previous programming course). Does this suggest the validity of the “no background” assumption? Second, for some inexplicable reason, M1 and M2 are added together, weighted as noted in Figure 1. On what possible grounds could this measure be treated as a proxy for the amount of learning taking place?

Ignoring, for the moment, the above fundamental design flaw, the paper provides numerous reasons to question the reliability of the measurement instruments employed. The two “exam” instruments appear to be administered with no pilot testing and—to make matters worse—the instruments are revised each time they are used. This eliminates any possibility that the reliability of the instruments can be determined. Compounding the problem, the sample instrument (presented as an exhibit in the manuscript) consisted entirely of essay-style questions. Without extensive training, how could coders (referred to as teaching assistants in the paper) possibly achieve acceptable levels of inter-rater reliability? Naturally, this implies that achieving acceptable construct validity is totally out of the question.

The remaining instruments (referred to as “assignments” in the paper) suffer from an entirely different set of methodological weaknesses. Because the paper reports they are routinely reused, it might be possible to establish their reliability and validity over time. Unfortunately, such a validation effort would tend to be wasted because of the haphazard manner in which they are administered. Although subjects are instructed to complete each instrument individually, the author(s) report(s) there is no way of ensuring this protocol is followed faithfully. In fact, the manuscript notes that the scores on the assignment instruments tend to exhibit minimal variability—far less than that observed for the exam instruments. Such an observation would be strongly suggestive of violations of the protocol, such as collaboration between subjects.

No review of the manuscript would be complete without noting some of the ethical concerns raised by a footnote in the paper that describes the protocol as a “gatekeeper course”. By this, the author(s) appear(s) to mean that one of the roles of the experiment is to discourage certain students from continuing to pursue the institution’s MIS major. Should this be the case, the protocol clearly runs afoul of the guidelines presented in U.S. Code of Federal Regulations Title 45, Part 46 (45 CFR 46) governing the protection of human subjects. Specifically, it appears that the guidelines of informed consent (45 CFR 46.116) have been completely ignored. To give some examples: a) participation in a “required course” could hardly be described as voluntary, b) subjects are not being properly informed of the risks and benefits of participation, c) explicit descriptions of the hardships associated with participation are not supplied to subjects, d) little or no effort seems to have been applied to minimize the risk to subjects (e.g., developing reliable instruments) –with the “gateway course” role implying intentional increases in such risks may be present, and e) no consent waiver is available to subjects. It seems hard to imagine a protocol less likely to gain institutional review board approval.

Taken together, these methodological deficiencies make it doubtful that any amount of revision would lead to an acceptable QMISR submission. The author(s) is/are encouraged to rethink the entire approach to the research. Replacing freeform surveys and unsupervised project instruments with validated multiple-choice surveys (for which item analysis can be conducted and principal factors can be determined) would be an important first step towards establishing rigor. A pre-test/post-test design could then be used to determine incremental knowledge transfer. Given the relevance of the topic, these changes could lead to a manuscript publishable in a peer-reviewed journal—if not necessarily QMISR.

Reviewer 2 Comments

Summary Ratings					
Relevance	**1 - None**	2 - Minimal	3 - Modest	4 - Strong	5 - Outstanding
Rigor	1 - None	2 - Minimal	3 - Modest	**4 - Strong**	5 - Outstanding
Recommendation					
Manuscript disposition:	**1 - Reject**	2 – Major revisions	3 – Accept, minor revisions	4 – Accept, as is	

Comments to author(s):

Let me start by commending you on the obvious effort you have put in towards creating a rigorous course that places significant demands on your students. We all know that essay examinations are far more effort than packaged multiple choice exams, and the sample project that you created for the course (attached as an exhibit to the original paper) demonstrates a real flare for the creative. The high student evaluations that you reported in the manuscript show that your work is appreciated.

Sadly, even the most rigorous course will not inform the QJMISR reader if it fails to be relevant. You have fallen into the trap of teaching programming in a manner that bears no resemblance to the way in which it is actually conducted in the field. This problem, which has been noted in the literature (e.g., Prey, 1995), is summarized in Table 1. As you can see, virtually everything that is prized in real world programming is discouraged in your course. Indeed, the skills most valued by managers—such as the ability to work well in groups and to identify code that is suitable for reuse—are defined to be “cheating” in your course design. What message are you sending to your students?

Table 1: Classroom vs. "Real World" programming

Programming Course	**Real World Programming**
Individuals required to program in solitude	Programming done as group activity
Small independent programs developed	Large, complex, interdependent programs
Code developed from scratch	Reuse of code strongly encouraged
Students write full standalone applications	Application maintenance dominates
Objective is to get code that works	Testing and debugging emphasized
Code is designed and developed by student	Code is written to specification

As if Table 1 were not enough, you also provide ammunition to use against your own design within the body of your paper. By your own estimates, based on internal surveys of students and alumni, only 15-20% of your MIS majors will eventually end up writing programs as all or part of their job. What value added does your course provide to these students? Does it provide them with insights into the programming process that they can use as IS managers? Hardly… Does it teach them to analyze problems and specify solutions? Nothing that isn't trivial…

Is the reader therefore to conclude that your institution's MIS concentration has so many extra credit hours sitting around that it can afford to require 85% of its students to spend a semester studying something they'll never use?

Here's what I see to be the bottom line. Twice in the past decade, at our institution, we've conducted focus groups with IS managers in an attempt to determine an appropriate model for our MIS undergraduate major. Each time, we went in expecting these managers to provide us with a list of the "hot" technical skills we needed to teach. No matter how we tried prompting them (e.g., "wouldn't it be nice if our students were experts at SQL"), however, the group came to the same conclusion each time—the most important skills our undergraduates could possess are: 1) the ability to work effectively with others (teamwork) and 2) effective communications skills. Neither of these seems to be the focus of the course described in your paper, nor do they seem likely to emerge in a revised version. Thus, I'm forced to recommend rejection.

Reviewer 3 Comments

Summary Ratings					
Relevance **N/A**	1 - None	2 - Minimal	3 - Modest	4 - Strong	5 - Outstanding
Rigor **N/A**	1 - None	2 - Minimal	3 - Modest	4 - Strong	5 - Outstanding
Recommendation					
Manuscript disposition:	1 - Reject	**2 – Major revisions**	3 – Accept, minor revisions	4 – Accept, as is	

Comments to author(s):

OVERALL COMMENTS
It is impossible to assign grades for rigor or relevance to the manuscript, since it represents only a start to a broader investigation. Since the attempts at measuring and quantifying course effectiveness presented in the paper exhibit some potential promise, the recommendation is to resubmit the manuscript once the research is completed.

Given that the author or authors probably did not view the analysis of a course that was submitted to be a starting point, it is reasonable to offer some explanation. Based upon the description provided, it is clear that the course that was the focus of the manuscript was not working. To be sure, the instructor demonstrated considerable resourcefulness in attempting to fit the content (teaching programming) into a classic "lecture and test" format described in the manuscript. But it is equally evident that the course was not meeting the needs of the MIS major—which includes the many individuals who will never program—nor was it effective in addressing the diverse backgrounds of the students (which "pegged the meter" in terms of variety of programming experience). What is needed, therefore, is a new design to which it can be compared. For this reason, the remainder of this review will focus on the nature of the design issues that must be confronted if a revised manuscript is to be successful.

A POSSIBLE FRAMEWORK

Perhaps the first issue that must be considered in a revised design is the underlying motivation for teaching the material. One dichotomy that can be helpful is that of *useful* versus *enriching*. Useful content is delivered under the assumption (shared by student and instructor, hopefully) that it will, one day, prove directly applicable to some activity the student hopes to accomplish. Enriching content, on the other hand, is bound by no such utilitarian constraints. Content that helps the student appreciate beauty (e.g., art history, music appreciation), inspires self-awareness (e.g., philosophy), experience spirituality (e.g., religious studies) or give expression to some inner muse (e.g., creative writing) all fall under this category. Some lucky disciplines (e.g., mathematics) may be fortunate enough to rate high in both dimensions whereas others (e.g., political science) may seem to require the invention of some additional dimension to justify their existence. But, for the most part, it makes sense for an instructor to focus on one or the other. Making such a choice is important because it impacts how to evaluate success. The effectiveness of a "useful" course should be assessed primarily based on the student's ability to perform some predetermined set of intended activities. Success in achieving enrichment, on the other hand, may well be in the eye of the beholder—meaning that student perceptions that the outcome has been achieved may be de facto evidence of success.

A second dichotomous course objective is that of breadth vs. depth of knowledge. A breadth-oriented course would endeavor to achieve X% knowledge of 100% of the material in preference to 100% knowledge of X% of the material. A depth oriented course would strive for the opposite outcome. Breadth-oriented approaches tend to be suitable for surveying a large body of material, as might be done in a typical introductory course. Depth-oriented courses, in contrast, tend to emphasize moving students up knowledge hierarchies, such as those proposed by Bloom (1956) or Perry (1970), with less effort directed towards assuring no content is skipped.

Another key design decision involves overall course structure. One way of thinking about this involves looking at two dimensions (as shown in Exhibit 1). The first of these considers outcome vs. process focus. An outcome-driven design defines successful course completion in terms of what the student knows (or can do) and avoids worrying about the path taken by the student to get there. A process focus, on the other hand, emphasizes monitoring the performance of a series of prescribed activities on the part of the student. Measured achievement often takes on a secondary role in such a design. The orientation chosen by the instructor will typically be manifested in policies such as requiring class attendance (process-oriented) or flexibility in offering alternative paths to course completion for students with experience (outcome oriented). Grading policy is also a good indicator. Outcome-oriented courses tend to have pre-established grading criteria, emphasizing absolute performance measures. Process-oriented courses are more likely to assign grades based on relative performance (e.g., using a bell curve) and are less likely to articulate objective grading standards. Because this dimension closely corresponds to the manner in which tasks are presented—which can be as a set of goals, a set of prescribed activities, or a combination of the two (Hackman, 1969)—it could be referred to as *task presentation*.

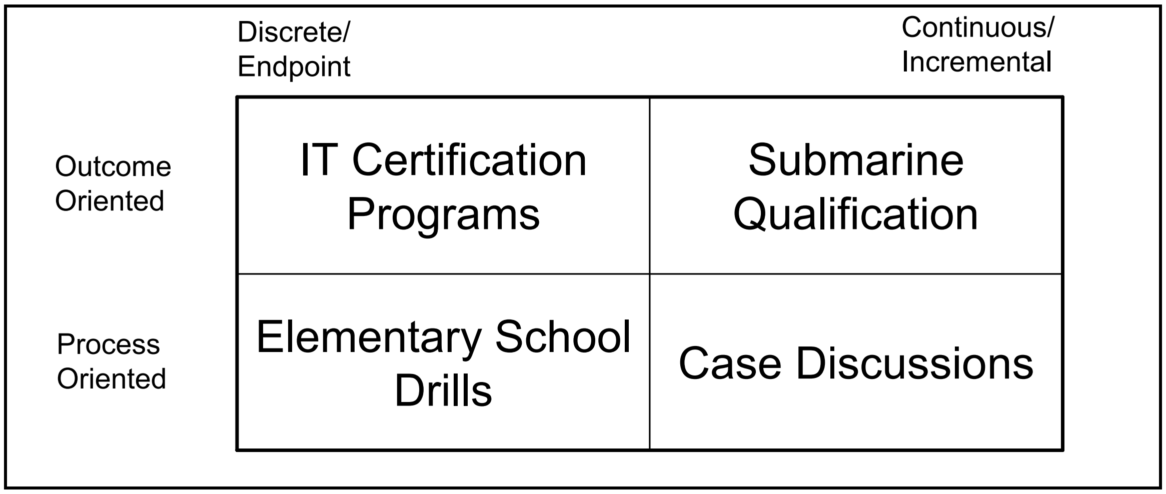

Exhibit 1: Course structures

The second dimension reflects mainly how the education process is measured, the range being from endpoint/discrete (absolute) to continuous/incremental (transformative). Where an absolute orientation is present, the principle concern is identifying the full spectrum of what the student knows at the end of the course or module. Measurements are discrete and tend to be initiated by the instructor or some external agency. Instruments attempt to capture the full range of achievement, with their validity being a source of great concern. Transformative orientation focuses on changes in individual knowledge, rather than absolute levels. Because small changes in narrow areas of knowledge or performance are easier to observe directly than full-spectrum transformations, continual evidence of incremental learning is normally sought. Such evidence will often come from student-initiated activities or interactions, meaning reliability often depends on the instructor's skills as an observer.

Interestingly enough, as shown in Exhibit 1, reasonable examples can be cited for all of the four of the design combinations. In elementary school, the process-absolute combination is in effect when students are drilled in handwriting—everyone does the same exercise for the same amount of time regardless of whether their handwriting is already neat or nearly illegible, with ultimate success perhaps being judged based on the performance of some standardized test devised by the state. Similarly, CPA and bar certifications require a process (e.g., law school or time in the field) and measure at an endpoint (e.g., the certification exam). Outcome-absolute is present for most forms of computer certification—a certificate is awarded immediately upon passing the test; the means through which the knowledge was acquired being deemed of no consequence. The process-transformative combination is embodied in the case method. Any experienced case method instructor will readily concede that measuring the absolute level of achievement resulting from case discussions is nearly impossible. Instead, the moderator conducts case discussions according to well established protocols and listens intently for transformations in the sophistication of the students. Credit is awarded for participation, granted incrementally as opposed to being measured at predetermined intervals. One clear piece of evidence that the design is based on process—rather than outcome—is that the same MIS cases that are discussed by MBA candidates are also discussed by senior MIS executives at institutions such as Harvard Business School, using the same basic protocol in both instances. It is hard to envision any objective, full-spectrum achievement measure that would be equally useful for both groups.

The final quadrant, outcome-transformative orientation, is the hardest to find realistic examples for. The problem here is that to be truly reflective of this approach, continual measures of achievement need to be acquired so that changes in it can be measured. One reasonable example of the approach can be found in the nuclear submarine qualification process. Upon reporting to a submarine, a sailor (officer or enlisted) is provided with a “qualification card” (typically a bound document consisting of dozens of pages) that identifies several hundred interviews and practical activities (standing a watch, performing a specific chemical analysis), each of which must be signed off by a qualified individual. Over a period that typically ranges from a year to eighteen months, the sailor acquires these signatures one at a time—with progress being continually monitored by the individual’s supervisor. Upon completion of the process, the sailor has a final interview with the ship’s captain or executive officer and is designated qualified in submarines (signified by a dolphin insignia thereafter worn on the sailor’s uniform).

PARTING COMMENTS
For this reviewer to recommend publication in QJMISR, the revised manuscript must contain a new course design whose outcomes can be compared with those of the design already submitted. Furthermore, such a design needs to be based upon a strong conceptual foundation (the author(s) may use the one provided above, or some other appropriate choice.)

It is hoped that such a revision will be undertaken. More high quality research into effective pedagogy is long overdue in the MIS field. Some key benefits of conducting such research are presented in Exhibit 2.

10. You'll never again have to worry about inspiring envy or jealousy in your colleagues.

9. You'll never have to make up another inane justification for why using student subjects is appropriate.

8. Whenever anybody asks you about IRB approval you can shout: "I'm exempt (45 CFR 46.101b) you loser!"

7. You can brag to your colleagues that you're on a 100% research load even while you're teaching 4/4.

6. You'll never again have to worry about reviewers checking your citations because even the Library of Congress refuses to keep most MIS education journals in its holdings.

5. If you're worried that your N is too small, just drop another assignment into the syllabus.

4. You can submit a paper beginning with an 11 page nightmare sequence (that concludes with the transformation of an IRB panelist into a capybara) and still get the “best paper” award for the innovative education track at DSI.

3. You'll get to write about yourself in the third person and refer to yourself as "the instructor". A lot…

2. Cocktail party chat: "I'm using the findings from my last three teaching research articles to run my class on a day-to-day basis. Now, refresh my memory... Who is using the results of your most recent ISR piece daily in the course of running their business?"

1. Once tenured, you'll never have to worry about facing that gut wrenching decision of whether or not to accept a competing offer from a prestigious academic institution.

Exhibit 2: Top 10 Reasons for engaging in MIS education research

Author Response to Reviewer Comments

To: Reviewers 1, 2 and 3
From: Author 635
Re: Submission 616
Subj: Response to your comments

The purpose of this memorandum is to identify the changes made in the revised manuscript in response to reviewer concerns. For organizational purposes, they are addressed in reverse order. Outcomes and conclusions sections are also provided, for all reviewers.

Response to Reviewer 3

The framework and examples introduced by Reviewer 3 have proven instrumental in the course redesign. With respect to the underlying motivation of the course, usefulness clearly dominates enrichment. This choice was a natural consequence of the course residing in an undergraduate MIS major—the MIS discipline not being known for inspiring its students with its beauty (or spirituality or ability to stimulate some inner muse).

With usefulness as its motivation, it was fairly natural that an outcome focus (vs. process focus) be established for the revised course. Where it differed from the original course was in its reliance on transformative (vs. absolute) measures. Rather than employing traditional exams, the revised course is based upon a structure closely resembling that of submarine qualification (Gill, 2005), the example suggested by the reviewer. In particular:

- Students are given a "validation card" at the beginning of the course, modeled after a submarine qualification card (an actual card can be accessed at the archival course site http://ism3232.coba.usf.edu/Archive/ValidationCard2005.pdf).
- Course requirements specify students must complete enough assignments to earn their desired grade. Each assignment has to be validated by oral exam with a teaching assistant (analogous to interviews in submarine qualification) or by passing a computer-based test.
- To ensure continuous progress is being made, students have to check in with an assigned teaching assistant each week. This can be done in person, by form or by posting an entry to a web log, which is monitored by course staff using an RSS feed. At the end of each week, students are sent a copy of a computer-generated report on their progress as an email attachment (see http://ism3232.coba.usf.edu/Archive/SampleReport.htm).
- A student's grade consists entirely of two components: 90% is based upon validated assignment scores, and 10% is based upon completion of weekly check in requirements.

Because there is little variation in assignment scores (which are typically in the 80-100% range, similar to the original manuscript), student grades tend to be based on the number of assignments completed. This is consistent with the depth of knowledge outcome. Implementing the assignment-centric also required the adoption of a self-paced format, since lectures could not be synchronized with many different rates of student progress.

To accommodate self-paced learning, students are provided with access to a large number of instructional resources—including multimedia lectures developed especially for Internet delivery and multimedia supporting material (roughly 17 hours of content) embedded in the course textbook (written by the instructor). These files are made available on a web site that is organized according to assignment, and directs the student to all relevant materials (an archival version of the Spring 2005 site can be found at http://ism3232.coba.usf.edu/Archive/).

There is also an asynchronous discussion group established for each assignment, with 100-200 postings each being fairly typical for the 4 largest projects (enrollments tend to range in the 70-80 student range, after 10-20% attrition during the first two weeks of the course). Students are given complete freedom with respect to which of these resources they use, with access being monitored only for the purposes of assessing instructional effectiveness. Assignments have no due dates, but teaching assistants assess the degree to which their assigned students are "on track" weekly.

Response to Reviewer 2

Reviewer 2's concerns specifically focused on the degree of disconnect between the original programming course and the way programming is actually conducted in the field. The revised course addresses these concerns in a number of ways:

- Students are encouraged to work in groups on assignments, with rigor being maintained through the validation process (see comments to Reviewer 3 for a further description).
- A peer-centric focus is maintained, and a sense of teamwork emphasized, through the use of undergraduate teaching assistants, all of whom recently completed the course. This is closely analogous to the role played by qualified crew members on submarines.
- Assignments are typically much more complex then those normally encountered in an introductory course (particularly the last two, which involve the creation of CGI applications; Gill, 2004), and must be written to specification.
- Students are given a substantial amount of code as a starting point for all programming projects, making code development more reflective of "real world" maintenance activities.
- The self-paced format requires students to manage their own time more effectively—or to face the consequences of their procrastination.

In addition to the above characteristics of the course, which are particularly applicable to programmers, the entire oral exam process provides students with the opportunity to practice their communications skills. There is also a secondary benefit: a number of graduates have remarked on the similarity between these exams and the job interviews conducted by firms such as Microsoft. The use of discussion groups for assignment support also teaches students to seek support using the preferred communications channel of most IT vendors.

The self-paced nature of the course tends to ensure that all students achieve a certain level of mastery of fundamental concepts that are the focus of early assignments (e.g., what is an algorithm, elementary flow charting, the software testing cycle), while only more advanced students—those seeking a B or better—are required to learn C++-specific concepts, such as pointer arithmetic, structures and object construction.

Response to Reviewer 1

The key concerns of Reviewer 1 related to the rigor of the evaluation process. Using the submarine qualification model has dramatically changed how students are evaluated. Excepting the first project, a "freebie" worth 5%, no assignment grade is recorded for a student until that assignment has been validated. The nature of the validation technique varies from assignment to assignment. Two of the assignments are fill-in-the-blanks exercises, one that deals with numbering systems and one that deals with pointers and structures. To validate these, the instructor developed two C++ programs that automatically generate exams in a format that can be uploaded to Blackboard. Two copies of the same exam question bank are uploaded. One serves as a practice test, randomly selecting 10 questions from the 500 question bank and providing feedback as to the correct answer. The other is the validation test, which randomly selects 10 questions (from the same bank) that students must take in a proctored setting. The validation scale is sliding—an assignment grade of 80-100% requires an 80% validation score, 60-79% requires a 60% validation score, 40%-59% requires a 40% score and so forth. If a student does not achieve successful validation on the first attempt, he or she may retake the exam (under supervision) as many times as desired.

For programming projects (of which there are 4), validation is accomplished through oral exams. After their project is graded, students must meet individually with a TA or the instructor, and answer questions relating to the specific code that they handed in. The nature of the questions does not presuppose that the student wrote the actual code; this would be a very bad assumption indeed. Instead, the questions focus on verifying that the student completely understands the code he or she handed in. Naturally, the more the code resembles whatever code is making the rounds in a given semester, the more probing the questions tend to be.

Students failing to validate a project may retake the oral exam—once with a second TA and, after that, only with the instructor. To ensure the consistency of exams across TAs, all students enrolled in the course who express an interest in becoming TAs must take their remaining oral exams with the instructor.

The validation process has dramatically reduced the problem posed by the diversity of experience in incoming students. An important feature of the course design is that students, in effect, set their own objectives. Students quickly discover that copying an assignment from the class expert—which can be very easy some semesters, since students have gone so far as to create web sites with assignment solutions posted—is a mixed blessing. They get a high assignment grade but then find the validation process to be like the inquisition—nearly always involving multiple trips to the instructor's office, with each retake of the exam involving questions that are ever more detailed. In subsequent assignments, these students will either seek out code of lesser quality or, in most cases, take a more active role in the development of code they submit.

The ethical dilemma posed by the “gatekeeper” role of the course has also been partially addressed by the revisions to the course structure. At the time of the original course—the years prior to and including 2000—dramatic growth in MIS enrollments was being experienced and steps needed to be taken to prevent overcapacity. Since that time, a nationwide decline in MIS and computer science enrollments has been experienced. As a consequence, emphasis has shifted towards the more laudable objective of maximizing the likelihood that any qualified student will remain in the major. Towards this end, the new design has proven particularly effective. When the same course was taught in both traditional and revised designs during the same semester, the percent of DWF (D grades, withdrawals and failures) ran 10-15% lower for the new protocol. Since the revised approach was adopted for all sections of the course, the DWF rate has further dropped about 10% (to about 25%) as modifications (most notably, weekly reports) were incorporated into the protocol. These reductions in DWF rates occurred despite course content increasing by nearly 30% over the same period.

Outcomes (all reviewers)

A number of outcomes appear to support the belief that the revised course design has been successful across a variety of dimensions. From an extensive survey of students conducted in 2003, for example:

- 93% were either satisfied or very satisfied (66%) with using Blackboard for online support, an activity on which they report spending 4 hours each week.
- 79% found TAs to be helpful in their learning. Individual ratings of TAs were high, ranging from 3.6 to 4.7 on a 1 to 5 scale.
- Students reported spending an average of 4 hours per week interacting with either the TAs or the instructor.
- 68% felt the grading system was much help or very much help to their learning.
- 75% felt that working with peers was much help or very much help in learning.
- Students reported spending an average of 6 hours per week working in groups.

- 71% were satisfied or very satisfied with group activities.
- 75% reported that the assignments contributed to their understanding of the course material (12% disagreed).
- 72% disagreed or strongly disagreed (median and mode of 1, on a 1-5 point scale) that more emphasis should be placed on tests (15% agreed).
- Students reported spending 17 hours per week on the course, more than twice as much time reported spent on typical MIS courses and more than three times that for typical business courses.
- 76% were satisfied or very satisfied with course multimedia materials.

From an economic perspective, the new design has also proven to be quite effective. A single instructor supervises numerous 35 student sections with undergraduate TA support (of roughly half an hour per student per week). Compared with assigning an instructor to each section, as was previously done, TA costs have proven to be about half the cost of hiring adjuncts, and a much smaller fraction of what staffing each section with tenure/tenure track faculty would cost. That these savings can be achieved with an increase in apparent course quality (as measured by amount of content covered and most measures of student satisfaction) suggests the revised course is a worthwhile prototype to consider for other types of classes.

Conclusions (all reviewers)

The revised course represents a relatively radical solution to the problem of a course that was not working at peak effectiveness. Although less ambitious approaches could have been attempted, the availability of a highly successful model for the approach (submarine qualification), along with the strong pedagogical justifications for preferring depth to breadth (e.g., Bloom, 1956; Perry, 1970) made the self-paced approach appear more attractive. The result has been a new way of teaching programming that appears to be more rational in design, relevant to real world programming and rigorous than its "lecture and test" predecessor.

References

Bloom, B.S. (Ed.) (1956). *Taxonomy of educational objectives: The classification of educational goals: Handbook I, cognitive domain*. New York; Toronto: Longmans, Green.

Bureau of Labor Statistics. (2004). *Occupational outlook handbook, 2004-2005 Edition (Table 1)*. Retrieved 3/30/2004 from http://stats.bls.gov/news.release/ooh.t01.htm

Gill, T.G. (2004). Engaging introductory programming students with CGI. *Decision Sciences Journal of Innovative Education*. Retrieved 1/27/2005 from http://www.mba.wfu.edu/dsjie/Tips/gill171.htm

Gill, T.G. (2005). Learning C++ Submarine Style: A Case Study. *IEEE Transactions on Education*, *48* (1).

Gill, T.G. & Hu, Q. (1998). Information systems education in the USA. *Education and Information Technologies*, *3*, 119-136.

Hackman, J. R. (1969). Toward understanding the role of tasks in behavioral research. *Acta Psychologica 31*, 97-128.

Hu, Q. & Gill, T.G. (2000). IS faculty research productivity: Influential factors and implications. *Information Resources Management Journal*, *13* (2), 12-22.

Perry, W. G., Jr. (1970). *Forms of intellectual and ethical development in the college years: A scheme.* New York: Holt, Rinehart, and Winston.

Prey, J. C. (1995). Cooperative learning in an undergraduate computer science curriculum. *IEEE Frontiers in Education Conference. Session 3c2.* 11-14.

Roberts, E. (2000). Strategies for encouraging individual achievement in introductory computer science courses. *SIGCSE 2000, 03/00 Austin TX.* 295-299.

Biography

Grandon Gill is an Associate Professor in the Information Systems and Decision Sciences department at the University of South Florida. He holds a doctorate in Management Information Systems from Harvard Business School, where he also received his M.B.A. His principal research focus is in the area of IS education, and he has published many articles describing how technologies and innovative pedagogies can be combined to increase the effectiveness of teaching across a broad range of IS topics. Currently, he teaches programming, database and managerial courses to both undergraduate and graduate students.