Volume 10, Issue 5
October 2012
ISSN: 1545-679X

INFORMATION SYSTEMS EDUCATION JOURNAL

In this issue:

The **Information Systems Education Journal** (ISEDJ) is a double-blind peer-reviewed academic journal published by **EDSIG**, the Education Special Interest Group of AITP, the Association of Information Technology Professionals (Chicago, Illinois). Publishing frequency is six times per year. The first year of publication is 2003.

ISEDJ is published online (http://isedjorg) in connection with ISECON, the Information Systems Education Conference, which is also double-blind peer reviewed. Our sister publication, the Proceedings of ISECON (http://isecon.org) features all papers, panels, workshops, and presentations from the conference.

The journal acceptance review process involves a minimum of three double-blind peer reviews, where both the reviewer is not aware of the identities of the authors and the authors are not aware of the identities of the reviewers. The initial reviews happen before the conference. At that point papers are divided into award papers (top 15%), other journal papers (top 30%), unsettled papers, and non-journal papers. The unsettled papers are subjected to a second round of blind peer review to establish whether they will be accepted to the journal or not. Those papers that are deemed of sufficient quality are accepted for publication in the ISEDJ journal. Currently the target acceptance rate for the journal is about 45%.

Information Systems Education Journal is pleased to be listed in the 1st Edition of Cabell's Directory of Publishing Opportunities in Educational Technology and Library Science, in both the electronic and printed editions. Questions should be addressed to the editor at editor@isedj.org or the publisher at publisher@isedj.org.

INFORMATION SYSTEMS EDUCATION JOURNAL

Editors

ISEDJ Editorial Board

Software Engineering Frameworks: Textbooks vs. Student Perceptions

Kirby McMaster
kmcmaster@weber.edu
CSIS Dept, Fort Lewis College
Durango, CO 81301, USA

Steven Hadfield
steven.hadfield@usafa.edu
CS Dept, U.S. Air Force Academy
Colorado Springs, CO 80840, USA

Stuart Wolthuis
stuartlw@byuh.edu
CIS Dept, Brigham Young University-Hawaii
Laie, HI 96762, USA

Samuel Sambasivam
ssambasivam@apu.edu
CS Dept, Azusa Pacific University
Azusa, CA 91702, USA

Abstract

This research examines the frameworks used by Computer Science and Information Systems students at the conclusion of their first semester of study of Software Engineering. A questionnaire listing 64 Software Engineering concepts was given to students upon completion of their first Software Engineering course. This survey was given to samples of students at three universities. To identify which topics were most important, students were asked to rate each concept on a ten-point scale. From their responses, we calculated the average perceived importance for each concept. This paper analyzes the results of this survey for the three student samples. We then compare the student ratings with word frequencies exhibited by authors of Software Engineering textbooks. In this way, we show how student frameworks relate to frameworks presented by Software Engineering authors.

Keywords: Software Engineering, framework, gestalt, schema, concept, rating.

1. INTRODUCTION

Learning is more effective if course topics and concepts are organized within an overall mental *framework*, or *gestalt*. By gestalt, we mean "a configuration or pattern of elements so unified as a whole that it cannot be described merely as a sum of its parts" (www.thefreedictionary.com). Each concept is introduced as a "piece" of a puzzle. The framework allows the pieces to fit together into a meaningful "whole". Other similar terms used by authors include *schema*, *paradigm*, and *mental model*.

According to Donald (2002), a course needs a schema to enable and improve understanding.

> A *schema* ... is a data structure of generic concepts stored in memory and containing the network of relationships among the constituent parts.... If we are to understand the relationships between concepts, we need to know in what order and how closely concepts are linked and the character of the linkage.

Bain (2004) describes why instructors should provide frameworks for courses, rather than rely on students to form their own.

> The students bring *paradigms* to the class that shape how they construct meaning. Even if they know nothing about our subjects, they still use an existing mental model of something to build their knowledge of what we tell them.

Frameworks are common in virtually all Computer Science and Information Systems (CSIS) courses. Often, primary concepts are organized into a *layered* framework, where services received at one layer are provided by algorithms and data structures in a lower layer. Computer Network courses favor layers consisting of a blend of the OSI Model and the Internet Protocol Suite (Peterson & Davie, 2011). Operating Systems courses include topics from the hardware, kernel, system services, and application layers (Silberschatz, Galvin, & Gagne, 2011). Computer Hardware has layers from simple digital logic up to VLSI circuits and functional components (Patterson & Hennessy, 2008). Database courses insert a DBMS software layer between application programs and operating system files (Connolly & Begg, 2009).

Not all computing frameworks are layered. The usual framework for Object-Oriented Programming (Lafore, 2001) includes sets of interrelated classes, arranged according to established design patterns (Gamma, Helm, Johnson, & Vlissides, 1994). Data Structures course topics are divided into data structure and algorithm categories, such as stacks, queues, linked lists, searching, and sorting (Drozdek, 2008). Artificial Intelligence has utilized a variety of frameworks over the years for search strategies, game playing, learning models, knowledge-based systems, and intelligent agents (Russell & Norvig, 2009).

But which frameworks are suitable for Software Engineering (SE) courses? Pressman (2009) and Sommerville (2010) offer common variations (such as "waterfall" and iterative) of the classical life cycle approach to software development. Schach (2010) focuses more on object-oriented methods. Cohn (2009) encourages successful management practices to integrate agile development with Scrum.

In our previous research (McMaster, Rague, Hadfield, & Anderson, 2008), we examined frameworks for software development from the viewpoint of textbook *authors*. We determined which words are used frequently in three samples of books: Object-Oriented Programming, Database, and Software Engineering. Our assumption was that words used most often in a book indicate the gestalt of the author. From each sample of books, we constructed a framework (or scale) as an ordered list of most frequent words.

In this research, we sought to determine what mental frameworks *students* had developed at the completion of their first SE course. We examined whether their frameworks were consistent across courses taught by different instructors at different schools. We also compared the student frameworks with those of authors of commonly used SE textbooks.

The remainder of this paper is organized as follows. First, we present our methodology for gathering data on student ratings of SE concepts. Next, we analyze the results to determine which concepts students perceive as most important. We then look at rating pattern variations for courses taught by different instructors. Finally, we compare student ratings with word frequencies in SE textbooks.

2. METHODOLOGY

In this section, we describe the methodology used in our study. A questionnaire listing 64 Software Engineering concepts (see Appendix B) was given to CSIS students upon completion of their first SE course. All but one of the concepts are described by a single word or acronym (e.g. *agile*, *design*, *quality*, *UML*). The concept *use case* is presented as a word pair.

These concepts were selected from a variety of sources. First, we chose topics that ranked high on a Software Engineering *gestalt* scale that we previously developed from frequently used words in SE books. We supplemented this word list with topics we felt were important, utilizing input from other instructors that teach SE courses. To encourage responses at the low end

of the scale, we intentionally added several words that are not SE-specific (e.g. *activity*, *language*). Once the list was compiled, it was randomized so that there would be no implied significance to the order in which the concepts were presented to students.

The SE concept list was included in a survey given to samples of students at three schools. School-1 consisted of 9 SE students at a state university, School-2 consisted of 27 SE students at a national university, and School-3 consisted of 19 SE students at a private university. Almost all students were juniors or seniors and had completed courses in programming and data structures. Some students had also taken a database course. The course sections had different instructors and textbooks, but each sample of students received a fairly traditional first semester SE course with an emphasis on systems analysis and design.

To identify which SE concepts were valued most, students were asked to rate each concept on a 10-point scale, with 1 indicating "least important" and 10 indicating "most important". From the responses, we determined the average perceived importance for each concept within each sample. We calculated *trimmed means*, removing approximately the top and bottom 11% (1/9 or 2/19 or 3/27) of the individual ratings, so that extreme responses would not unduly influence the concept ratings.

We found that the trimmed means for the 64 concepts differed in a biased way between the three schools. To make the data for the samples comparable, we *standardized* (rescaled) the concept means within each school, so that the three sets of 64 scores had the same average (7.20) and standard deviation (1.00). This rescaling kept the combined mean at 7.20, but changed the standard deviations slightly. Note that we did not rescale individual student ratings. We rescaled the trimmed means in a way that preserved the ordering of concepts within each school. We could have achieved a similar result by converting the trimmed means to ranks, but then the concepts would have been equally spaced (except for ties).

After gathering and transforming the survey results, we had two types of data to analyze and compare: (1) student ratings for the three schools, and (2) textbook word frequencies from our prior research. We first examine the concept ratings for the three schools, both separately and combined. Next, we look at the ratings variation for each concept within schools and between schools. Then we compare the combined student ratings with word frequencies in SE textbooks.

3. CONCEPT RATINGS

In this section, we analyze the concept ratings for the three student samples. Table 1 presents the 32 top-rated Software Engineering concepts (out of 64), along with the rescaled trimmed means for School-1, School-2, and School-3.

Table 1. Top 32 concept ratings for schools.

SE Concept	School-1 N = 9	School-2 N = 27	School-3 N = 19	Combined Rating
design	8.71	9.19	8.71	8.87
quality	9.15	8.72	8.00	8.62
requirement	8.13	9.21	8.47	8.60
test	8.56	8.96	8.24	8.59
implementation	8.27	8.67	8.00	8.32
user	7.98	8.88	8.00	8.29
development	8.13	7.97	8.40	8.16
software	8.56	7.72	8.00	8.10
interface	8.42	8.30	7.38	8.03
information	7.98	7.76	8.24	7.99
analysis	7.83	7.35	8.79	7.99
solution	7.98	7.76	8.08	7.94
prototype	7.98	8.18	7.38	7.84
performance	7.83	7.68	7.85	7.79
customer	6.96	9.25	7.14	7.79
project	7.83	7.31	8.08	7.74
team	7.54	7.89	7.69	7.71
application	8.42	7.26	7.38	7.69
method	8.27	7.06	7.69	7.67
model	8.42	7.55	6.99	7.65
product	7.98	8.34	6.59	7.64
management	6.96	8.34	7.61	7.64
diagram	7.69	7.43	7.77	7.63
engineering	7.40	7.01	8.47	7.63
organization	7.54	8.38	6.83	7.59
program	7.83	7.10	7.69	7.54
system	7.40	6.97	8.08	7.48
data	7.98	6.56	7.77	7.44
function	7.83	6.85	7.61	7.43
code	7.69	7.10	7.46	7.41
process	7.40	6.52	8.32	7.41
architecture	6.96	7.72	6.91	7.20

We include a column showing the average rating of each concept for the combined sample.

The combined ratings are unweighted to prevent the larger School-2 sample from dominating the results. The concepts are listed in decreasing order, based on average rating.

A quick visual inspection of the three schools in Table 1 reveals substantial rating similarities for the concepts. In this table, the top five rated concepts, all with combined ratings above 8.30, are *design*, *quality*, *requirement*, *test*, and *implementation* (four life cycle phase descriptors, plus an umbrella goal). These five words received a mean rating greater than 8.00 within each school. Close behind are the ratings for *user*, *development*, and *software*.

The other 24 concepts in Table 1 have average ratings at or above the mean (7.20) for all 64 concepts. The 32 concepts having average ratings below 7.20 are presented in Appendix A.

Another way to view these results is with an ordered list of the 10 highest-rated concepts for each school. These three lists are presented in Table 2.

Table 2. Top 10 concepts by school.

Rank	School-1	School-2	School-3
1	quality	customer	analysis
2	design	requirement	design
3	test	design	requirement
4	software	test	engineering
5	interface	user	database
6	application	quality	development
7	model	implementation	process
8	implementation	organization	test
9	method	product	information
10	algorithm	management	solution

The concepts *design* and *test* are included in the Top-10 lists for all three schools. *Quality*, *requirement*, and *implementation* are listed for two of the schools. The remaining 18 concepts in Table 2 appear only once.

We can gather the top-rated words and several of the 18 unique words from Table 2 into brief conjectural descriptions of how the three SE courses differ.

> School-1: Quality is #1. The methodology uses models and algorithms to build applications.
>
> School-2: The customer is #1. Organization and management are necessary to create a product that will satisfy users. (Students in this course worked on real-world projects.)
>
> School-3: Analysis is #1. Databases are developed to provide information and solutions. (This was a CIS course.)

Among the bottom 32 concepts, four received ratings below 6.00: *change* (5.72), *domain* (5.44), *discipline* (5.33), and *formal* (4.56). There are several possible reasons why a concept received a below-average rating. Some concepts apply to later stages in the software development life cycle, such as *construction* (7.01), *integration* (6.59), *deployment* (6.57), *validation* (7.08), *verification* (6.95), and *maintenance* (7.03). These concepts presumably would receive more emphasis in a second-semester SE course.

Other concepts relate to a narrow range of the life cycle or to a specific technology, so they are less likely to receive continual emphasis during a semester. This includes concepts such as *agile* (7.01), *formal* (4.56), *incremental* (6.36), *pattern* (6.04), *UML* (6.74), and *use case* (6.87). And, as mentioned earlier, some concepts are fairly general rather than SE-specific, such as *activity* (6.38), *change* (5.72), *discipline* (5.33), *document* (6.67), *language* (6.56), and *state* (6.05).

Over the 64 concepts, the school ratings were reasonably consistent. The correlation coefficients between pairs of schools are summarized in Table 3. The correlations range from 0.480 (School-2 vs. School-3) to 0.576 (School-1 vs. School-3). These values suggest a moderate positive relationship between the concept ratings for the separate samples. The fact that the correlations are not larger suggests that some notable differences in ratings exist between the three schools. We examine sources of this variation in the next section.

Table 3. Ratings correlations between schools.

Correlations	School-1	School-2	School-3
School-1	1.000	0.568	0.576
School-2	0.568	1.000	0.480
School-3	0.576	0.480	1.000

4. RATINGS VARIATION

We collected concept ratings from students in SE courses at three schools. The previous section focused on ratings differences between SE concepts, especially with respect to concepts that are considered most important by students. In this section, we describe how ratings vary for one concept at a time.

4.1 Within-School Variation

The variability in ratings for each SE concept can be divided into two sources: *within-schools* and *between-schools*. We are primarily interested in between-school variation, which should better reflect the concepts that instructors emphasize in their courses. We computed within-school variation for each concept to provide a reference point for evaluating course differences.

For each of the 64 SE concepts, we calculated the (untrimmed) *standard deviation* for student ratings within each course. Rather than present individual values of these statistics, we summarize the pattern of variation by school in Table 4.

Table 4. Between-student ratings variation for concepts at each school.

Statistic	School-1	School-2	School-3
Min Std Dev	0.88	1.63	0.93
Max Std Dev	3.22	2.91	3.04
Avg Std Dev	1.86	2.25	1.92

The 192 standard deviations ranged from a low of 0.88 (School-1) to a high of 3.22 (again School-1). The average standard deviation value was slightly below 2.0 at School-1 and School-3, but was over 2.0 at School-2. So a "typical" measure of student-to-student variability for a concept is about 2.0. This is a relatively large amount of variation, considering that a "well-behaved" distribution has about 95% of the scores within two standard deviations (+/- 4.0) from the mean. On a 10-point ratings scale, this would be an interval of width 8. Many ratings distributions tended to be skewed, so the 95% rule is less relevant in these cases.

We also calculated the *range* of the ratings scores for each concept within each school. School-1 had an average range of 5.31, while the average range for School-3 was 6.39. The average range for School-2 was somewhat larger (8.05), which is consistent with the larger standard deviation for this school.

4.2 Between-School Variation

We now summarize the variation in ratings between schools in terms of patterns for concept means. For (untrimmed) means of random samples of size N, the variance of the means will vary inversely with the sample size N. So for a sample of size N = 9 (School-1), the standard deviation of the sample means would be approximately 2.0/3 = 0.67, assuming that the individual scores have a standard deviation of 2.0. For larger sample sizes, the means would vary less.

Two features of our methodology limit the strict validity of the above probability model for this study: (1) our samples were not random, and (2) we calculated trimmed means for each concept. The large within-school variation described earlier was part of the motivation for using trimmed means. Still, the above discussion provides a context for the way we interpreted differences in means between schools.

Table 5 lists the SE concepts for which the between-school ratings showed the largest differences.

Table 5. Concept ratings mean differences. (highest H or lowest L for concept)

SE Concept	School-1 N = 9	School-2 N = 27	School-3 N = 19	Range= Hi - Lo
database	6.37	5.98	8.47H	2.50
algorithm	8.27H	6.85	5.89	2.38
CASE	5.64	5.73	8.00H	2.36
customer	6.96	9.25H	7.14	2.30
cost	6.08	8.05	7.22	1.97
formal	3.89	5.85H	3.93	1.96
UML	6.37	6.01	7.85H	1.84
document	5.50L	7.22	7.30	1.80
process	7.40	6.52	8.32	1.80
product	7.98	8.34	6.59L	1.75

For each concept, we calculated the standard deviation and the range of the three school means. The *ranges* are shown in the table, with concepts listed in decreasing range order. We only include concepts with a range above 1.70, which is much larger than the random variation model for means described above. Four of the concepts--*database*, *algorithm*, *CASE*, and *customer*--have ranges larger than 2.0. This suggests that the SE instructors in our study vary noticeably in how they present these topics.

When a large range is obtained from three values, several patterns are possible:

1. One value can be much *higher* than the other two.
2. One value can be much *lower* than the other two.
3. The values can be evenly spread, with the middle value spaced about equally between the high and low values.

Looking *horizontally* at the mean ratings for each concept, we have marked a rating with an

H if it is much higher than the others, and with an L if it is much lower. For example, the *database* rating for School-3 is 8.47H, and the *document* rating for School-1 is 5.50L. Note that the low *formal* rating of 5.85 for School-2 is marked with an H, as the other two schools have even lower ratings for this concept.

We can also look *vertically* at the concept ratings in Table 5 to view the distinct ratings patterns for each school. Concepts may not have been rated as important, but they were rated much higher or lower by one of the schools. From this perspective, School-1 is high for *algorithm* and low for *document*. School-2 is high for *customer* and high (less low) for *formal*. School-3 is high for *database*, *CASE*, and *UML* and low for *product*.

4.3 Ratings Profiles

In Table 5, we listed SE concepts having the largest differences in mean ratings between schools. Now we provide a visual representation of the top-24 (of 32) concepts from Table 1, where concepts are ordered by decreasing average rating. Figure 1 provides a graph of the concept ratings for each school, with a separate "line" for each school.

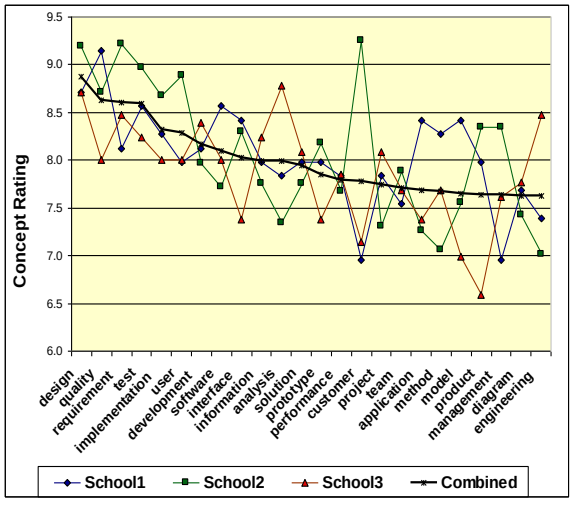


Figure 1: Top 24 concepts--profiles of 3 schools.

This figure presents the ratings pattern for each school as a *profile*. The successive differences between concept means for schools gives the illusion of random variation in most cases. Two exceptions are the concepts *customer* and *product*, where the ratings vary most widely. These concepts are included among the Table 5 concepts with large mean ratings differences.

5. STUDENTS VS. TEXTBOOKS

We now compare average concept ratings by students with two measures of word usage in Software Engineering textbooks. We exclude *use case* from this analysis, because this concept involves two words. Our textbook word counts are for single words only. For the remaining 63 concepts, we recorded how *often* and how *consistently* these words appear in a (nonrandom) sample of 36 SE books. Table 6 shows the concept ratings, word frequencies, and book counts for the top 32 student-rated concepts. Textbook results for the bottom 32 concepts are included in Appendix A.

Table 6. Top 32 concept ratings--students vs. textbooks.

SE Concept	Concept Rating	Textbook StdFreq	Books
design	8.87	158.3	**35**
quality	8.62	108.7	17
requirement	8.60	183.2	29
test (testing)	8.59	221.0	24
implementation	8.32	90.0	13
user	8.29	131.6	26
development	8.16	208.0	**36**
software	8.10	**377.8**	**36**
interface	8.03	103.5	18
information	7.99	109.4	27
analysis	7.99	92.4	26
solution	7.94	112.5	6
prototype	7.84	106.2	2
performance	7.79	61.5	7
customer	7.79	126.6	17
project	7.74	229.8	30
team	7.71	154.2	17
application	7.69	108.1	26
method	7.67	120.1	27
model	7.65	201.3	33
product	7.64	165.9	26
management	7.64	99.0	25
diagram	7.63	123.1	15
engineering	7.63	136.8	19
organization	7.59	108.3	16
program	7.54	145.6	26
system	7.48	**358.1**	**35**
data	7.44	154.9	32
function	7.43	93.1	21
code	7.41	118.8	27
process	7.41	**259.1**	**36**
architecture	7.20	117.3	13

To measure consistency of word use, the Books column gives the number of books (out of 36) that include the word in its concordance. The concordance is a list of the 100 most frequently used words in a book (excluding common English words). In Table 6, the words *software*, *development*, and *process* are listed in all 36 concordances; *design* and *system* are in 35 concordances.

To measure how often a word appears in a book, we rescaled each word frequency so that the average word frequency within a concordance was 100. This compensates for books having different total word counts. The standardized frequency (StdFreq) for a word is the average rescaled frequency across all books that include the word in its concordance. Based on this measure, the three most frequent words are *software* (StdFreq = 377.8), *system* (StdFreq = 358.1), and *process* (StdFreq = 259.1).

In Table 6, the word *model* has a StdFreq value of 201.3 for the 33 books that include this word in their concordances. The interpretation of this measure is that *model* occurs about twice as often as an average concordance word in SE books that include *model* in their concordances.

The below-average rated concepts from our questionnaire are not shown in Table 6. Three of these words--*discipline*, *incremental*, and *validation*--are not in the concordance of any of our sample books. This does not imply that these words do not appear in the books. It just means that they do not occur frequently enough to be listed in the concordances.

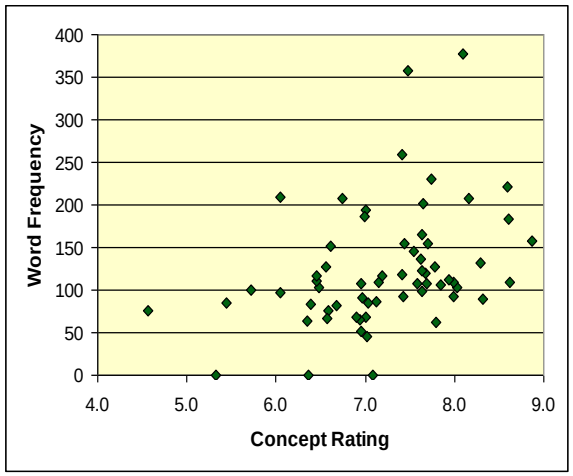


Figure 2: Concept rating vs. textbook word frequency.

Of current interest, the word *agile* (not in Table 6) appears in the concordances of just two SE books. In contrast, the standardized frequency of *agile* is 194.4, suggesting that these two books utilize this word heavily.

The scatter diagram in Figure 2 displays the relationship between the combined concept ratings for the students vs. the standardized frequencies of these words in the SE textbooks. Note the "diamonds" along the horizontal axis, representing the three books that were not listed in any concordance (and therefore received StdFreq values of 0.0)

In this graph, the words *software* and *system* appear as "outliers", in that the frequencies are noticeably higher for these words. One possible reason for the prevalence of these words is that they apply throughout the development cycle and are mentioned in multiple chapters in SE books. On the other hand, the highly rated word *quality* applies to every life cycle stage, but SE authors use this word less often.

The caution here is that word *frequency* does not necessarily imply *importance*. If we accept that the phrase "repetition brings conviction" applies to SE courses, perhaps we should emphasize important concepts such as *schedule* (StdFreq = 91.4, *cost* (StdFreq = 86.3), *maintenance* (StdFreq = 84.7), *document* (StdFreq = 81.8), and *performance* (StdFreq = 61.5) throughout the course, regardless of how sparingly these words appear in textbooks.

The correlation coefficient between combined concept ratings and textbook word frequencies is 0.373 (0.381 with the two high outliers removed), indicating a modest positive linear relationship. Not surprisingly, this is lower than the correlation coefficients for concept ratings between pairs of schools (which range from 0.480 to 0.576).

Thus, the students in this study agree more with each other on the relative importance of topics than they do with textbook authors, even though the students had different instructors and different textbooks. We are content that the correlation between concept ratings and textbook word frequencies is not negative. In the Internet era, many students do not bother to purchase or read course textbooks.

Figure 3 displays the relationship between combined concept ratings and the number of SE books containing concept words in their concordances. In this figure, no "outliers" are

obvious, probably because the number of books is bounded by 36.

The correlation coefficient between student concept ratings and number of textbooks is 0.415, which is slightly higher than the correlation between ratings and word frequencies. The diagram does illustrate how much "scatter" can be present in a relationship having a correlation of approximately 0.400.

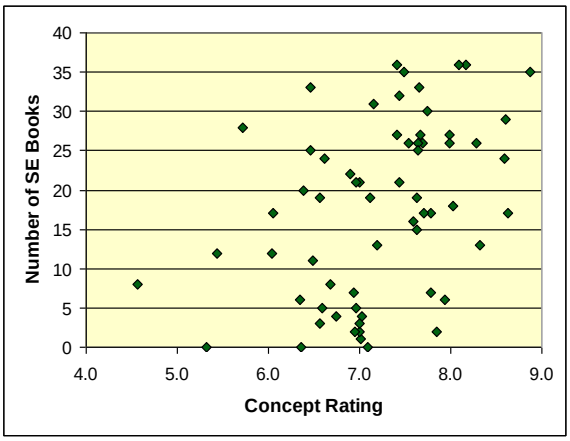


Figure 3: Concept ratings vs. SE books.

To summarize, we found a modest positive relationship between student ratings of concepts and the two measures of word occurrence in textbooks. Most of the concepts with above-average student ratings appeared in the concordances of the majority of the SE books and had a standardized frequency above 100. From the textbook point of view, all three SE words that failed to appear in any concordances had below-average student ratings.

6. SUMMARY AND CONCLUSIONS

Constructing a framework for knowledge is essential for students in a Software Engineering course. A successful mental framework can help students organize course topics into a meaningful whole, which promotes learning.

In a previous study, we developed an authors' SE framework based on word frequencies in popular SE books. In this current research, we surveyed students at three schools on the relative importance of topics in an introductory SE course. We chose 64 concepts that students might use in constructing their own mental frameworks for SE. After standardizing the data from students at each school, we obtained relatively consistent concept ratings.

The five highest rated words were *design*, *quality*, *requirement*, *test*, and *implementation*, based on averages across the three schools. Concepts that apply to early states or multiple stages of the software development life cycle tended to have higher ratings. Concepts that arise late in the life cycle or involve a specific technology had lower ratings.

Within schools, variability of student ratings for concepts was quite large, with an average standard deviation of about 2.0 (for a 10-point scale). There was less ratings variation between schools, partly due to our calculating trimmed means for each concept. The largest between-school variation occurred for four concepts--*database*, *algorithm*, *CASE*, and *customer*.

Overall, the ratings profiles for the top-24 concepts were reasonably consistent for the schools, with two exceptions (*customer* and *product*). As faculty, we often agree on what is most important, but we have difficulty agreeing on what is less important. As a result, each instructor emphasizes certain extra things that make her/his course distinctive.

When student ratings for concepts were compared to frequent (concordance) words in a sample of 36 SE textbooks, only a moderate positive relationship was found. Highly rated concepts appeared more often in the sample books, but three lower-rated words were not in the concordances of any of the books.

Current Software Engineering instructors can benefit from comparing results on student ratings as summarized in this paper with their own perception of most important concepts. Where there are differences, consider how you highlight your favored SE concepts. In particular, how do emphasize important concepts that do not appear frequently in SE textbooks?

On a related note, are you certain that the frameworks of your students are consistent with the primary objectives of your SE course? Not all student learning comes from listening to lectures, reading textbooks, and doing homework assignments. You are encouraged to use the questionnaire in Appendix B to obtain feedback from your students.

6.1 Future Research

Future research will include a replication of this study with larger samples to verify our preliminary findings. With additional data, we can check how specific textbooks used in

Software Engineering courses affect ratings of concepts. SE instructors could be surveyed in a similar manner to discover which concepts they feel are most important. We would then be able to assess how closely instructor ratings match those of their students.

We would also like to extend this research to examine how student frameworks evolve after taking additional SE courses, especially the SE II course. We would study how students perceptions change as they gain more experience with the later stages of the software development life cycle.

The focus of this research has been on words that form frameworks for Software Engineering. Beyond a collection of words, a framework should provide a meaningful context that explains how the words fit together. A special challenge for future research is to examine various ways that SE words can be integrated into a unified Software Engineering framework.

7. REFERENCES

Bain, K. (2004). What the Best College Teachers Do. Harvard University Press, pp 26-27.

Cohn, Mike (2009). Succeeding with Agile: Software Development Using Scrum. Addison Wesley.

Connolly, T., and Begg, C. (2009). Database Systems: A Practical Approach to Design, Implementation and Management (5th ed). Addison Wesley.

Donald, J. (2002). Learning to Think. Jossey-Bass, p 15.

Drozdek, A. (2008). Data Structures and Algorithms in Java (3rd ed). Cengage Learning.

Gamma, E., Helm, R., Johnson, R., and Vlissides, J. (1994). Design Patterns: Elements of Reusable Object-Oriented Software. Addison Wesley.

Lafore, R. (2001). Object-Oriented Programming in C++ (4th ed). Sams.

McMaster, K., Rague, B., Hadfield, S., and Anderson, N. (2008), Three Software Development Gestalts. In *The Proceedings of the Information Systems Education Conference 2008*, v 25 (Phoenix).

Patterson, D., and Hennessy, J. (2008). Computer Organization and Design (4th ed). Morgan Kaufmann.

Peterson, L., and Davie, B. (2011). Computer Networks: A Systems Approach (5th ed). Morgan Kaufmann.

Pressman, R. (2009). Software Engineering: A Practitioner's Approach (7th ed). McGraw-Hill.

Russell, S., and Norvig, P. (2009). Artificial Intelligence: A Modern Approach (3rd ed). Prentice Hall.

Schach, S. (2010). Object-Oriented and Classical Software Engineering (8th ed). McGraw-Hill.

Silberschatz , A, Galvin, P., and Gagne, G. (2011). Operating System Concepts (8th ed). Wiley.

Sommerville, I. (2010). Software Engineering (9th ed). Addison Wesley.

APPENDIX A

Bottom 32 Concept Ratings--Students vs. Textbooks.

SE Concept	School-1 N = 9	School-2 N = 27	School-3 N = 19	Combined Rating	Textbook StdFreq	Books
problem	6.52	7.72	7.22	7.15	108.8	31
cost	6.08	8.05	7.22	7.12	86.3	19
validation	6.37	7.97	6.91	7.08	--	0
maintenance	6.96	7.14	6.99	7.03	84.7	4
construction	7.69	6.60	6.75	7.01	45.2	1
agile	7.98	6.60	6.44	7.01	194.4	2
algorithm	8.27	6.85	5.89	7.00	68.4	3
class	7.40	6.14	7.46	7.00	186.7	21
schedule	6.37	8.01	6.52	6.97	91.4	5
specification	6.52	6.97	7.38	6.96	107.8	21
verification	6.37	7.35	7.14	6.95	51.0	2
database	6.37	5.98	8.47	6.94	65.9	7
control	7.25	6.48	6.99	6.90	68.8	22
use (case)	6.52	6.72	7.38	6.87	--	--
UML	6.37	6.01	7.85	6.74	207.3	4
document	5.50	7.22	7.30	6.67	81.8	8
component	7.10	5.89	6.83	6.61	152.0	24
integration	6.52	6.97	6.28	6.59	75.9	5
deployment	7.25	6.72	5.73	6.57	67.3	3
language	6.52	6.10	7.06	6.56	127.9	19
module	6.23	6.56	6.67	6.49	103.2	11
tool	5.94	6.14	7.30	6.46	110.9	25
CASE	5.64	5.73	8.00	6.46	117.4	33
activity	6.96	5.52	6.67	6.38	83.5	20
incremental	5.94	7.01	6.12	6.36	--	0
framework	6.52	6.64	5.89	6.35	63.4	6
state	5.79	6.01	6.36	6.05	97.6	17
pattern	6.81	5.81	5.50	6.04	209.5	12
change	6.23	6.06	4.87	5.72	100.2	28
domain	5.35	5.40	5.58	5.44	84.3	12
discipline	5.94	5.56	4.48	5.33	--	0
formal	3.89	5.85	3.93	4.56	75.9	8

APPENDIX B

Software Engineering Topic Identification Name _____

For each topic/concept listed below, please rate on a scale from 1 to 10 the importance of the topic in this Software Engineering course. On this scale, 1 represents "least important" and 10 represents "most important".

	Topic/Concept
_____	implementation
_____	algorithm
_____	model
_____	test
_____	activity
_____	domain
_____	deployment
_____	formal
_____	problem
_____	design
_____	interface
_____	data
_____	maintenance
_____	diagram
_____	discipline
_____	change
_____	customer
_____	cost
_____	agile
_____	schedule
_____	program
_____	UML
_____	CASE
_____	language
_____	code
_____	project
_____	engineering
_____	tool
_____	use case
_____	integration
_____	verification
_____	software

	Topic/Concept
_____	product
_____	construction
_____	performance
_____	pattern
_____	framework
_____	state
_____	system
_____	process
_____	development
_____	database
_____	class
_____	application
_____	requirement
_____	management
_____	organization
_____	architecture
_____	user
_____	control
_____	document
_____	incremental
_____	prototype
_____	quality
_____	validation
_____	module
_____	team
_____	solution
_____	information
_____	method
_____	function
_____	component
_____	specification
_____	analysis

Teaching Management Information Systems as a General Education Requirement (GER) Capstone

Bogdan Hoanca
afbh@uaa.alaska.edu
Computer Information Systems
University of Alaska Anchorage
Anchorage AK 99508, USA

Abstract

Although many IS programs nationwide use capstone courses in the major, this paper reports on the use of an upper division Management Information Systems (MIS) class as a general education requirements (GER) capstone. The class is a core requirement for all majors in the Bachelor of Business Administration (BBA) program at the University of Alaska Anchorage, which includes the MIS major. The BBA program is accredited by the Association to Advance Collegiate Schools of Business (AACSB).

The explosive developments in information technology have both economic and cultural impacts on society, and often lead to ongoing debates. In dealing with the impact of technology on society, the capstone class challenges students to integrate GER knowledge, business and their major-specific knowledge, and IT knowledge. Students must demonstrate skills across five dimensions: 1) knowledge integration, 2) effective communication (oral and in writing), 3) critical thinking and problem solving, 4) information literacy, and 5) quantitative perspectives. The five GER dimensions are assessed using a research project and a series of four hands-on projects (information literacy, database management, data mining, and decision support). The research project is based around a debate on topics relating to the impact of technology on society, and challenges students across all five dimensions. The hands-on projects focus more on information literacy, critical thinking and quantitative perspectives.

Assessment data collected over the past five years (spring 2007 to spring 2011) show that a majority of students (75% or more in recent years) consistently achieve passing scores across the five GER dimensions.

Keywords: general education requirement, capstone, management information systems, assessment

1. INTRODUCTION

Management Information Systems programs nationwide often include a discipline capstone course, focused on e-commerce (Abrahams & Singh, 2010), systems development (McGann & Canili, 2005) or emerging technologies (Janicki, Fischetti, & Burns, 2007) – and emphasizing soft skills (communication, interviewing, and client interaction). Instead, this paper reports on the use of an upper division MIS course as a general education requirement (GER) capstone.

A number of colleges and universities require GER capstone courses, mainly to give students an integrative experience, but also to facilitate assessment (Rowles, Koch, Hundley, & Hamilton, 2004). Such capstone courses are intended to help students integrate better across the seemingly disparate courses they took to

fulfill their GER. Additionally, because GER capstone classes rely on knowledge students acquired in their general education classes, assessments in a GER capstone class can evaluate the overall impact of general education courses on students (Wilson, et al., 2008).

Unlike discipline-specific learning, student learning in the general education classes is difficult to assess. Students have a choice of classes to meet GER, and they often transfer coursework from other institutions. Also, some of the GER skills are taught in multiple disciplines, with different approaches, expectations and outcomes (for example, critical thinking means different things in philosophy and in sociology) (Bers, 2000). While standardized tests or exit interviews can be used to assess GER, using papers in a capstone class appears to be a particularly good means in terms of: student motivation, costs of the instrument, and the ability to reflect both quantitative and qualitative aspects of the learning (Bers, 2000).

GER capstone classes have been used for many decades at some institutions. A survey of 707 institutions showed that 549 of them offered one or more capstone course, but most of these were discipline capstones, taught by a single faculty member in the discipline (Henscheid, Breitmeyer, & Mercer, 2000). The survey also uncovered the need for a more comprehensive assessment of the capstone classes.

More recently, assessment has taken center stage. Nancy Fernandez describes the assessment-focused culture at CalState Pomona and how the process has resulted in changes that improved student learning (Fernandez, 2006). The Pomona model involves an Integrative General Education Program culminating in a capstone course. Portland State developed their capstone model in 1994 (Kerrigan & Jhaj, 2007). Their assessment involves three types of feedback: a mid-quarter qualitative feedback session led by a trained facilitator in class; a quantitative student evaluation at the end of the term; and a qualitative survey of students' perception of their learning, also at the end of the term. Southeast Missouri State University assesses students both at the beginning and at the completion of their studies, including longitudinal and across sections comparisons (Blattner & Frazier, 2004).

Many GER capstone courses must satisfy multiple sets of requirements: departmental requirements (because the capstones are usually housed in an academic department), university wide requirements (applicable to all GER capstone courses at a given institution), and requirements from external accreditation agencies (Claus & Hawkins, 2007). Most if not all GER capstone courses tend to include some form of information literacy (ability to locate and evaluate information), communication, and critical thinking skills. The assessment tools used in the courses include research papers (with an oral presentation component) or portfolios (Brock, 2004).

This paper describes goals and achievements of a GER capstone class built around the Management Information Systems class at the University of Alaska Anchorage, in the College of Business and Public Policy. The class is a core required class for all non-accounting majors in the Bachelor of Business Administration program. Since the class became a GER capstone, accounting majors are often taking it to satisfy GER requirements.

First taught as a GER capstone in fall 2006, the class has been successful in achieving the intended goals. Assessment is built into the curriculum, and it is based on student artifacts that document student performance across a series of five GER capstone required areas (described below). Part of the assessment data is used for AACSB accreditation assessment in the College, but the data collected encompass a more extensive set than required for accreditation. Data collected over the past five years indicate that a majority of students perform well across the five GER dimensions.

The paper first introduces the GER capstone requirements at UAA and describes the curriculum development process (Section 2). Section 3 describes how the MIS class fulfills the GER capstone requirements. In Section 4, we present assessment data collected over the past five years, and we discuss student feedback and future plans. We present conclusions in Section 5.

2. GER CAPSTONE REQUIREMENTS AT THE UNIVERSITY OF ALASKA ANCHORAGE

University of Alaska Anchorage (UAA) is part of the State of Alaska public university system. UAA is the largest independently accredited

university in the state, and it is located in the largest population center. Anchorage is home to almost half of the 650,000 citizens of the state, and is the main hub for transportation, oil and gas, and health care industries. UAA celebrated its 50th anniversary in 2004 and offers close to 200 degrees and programs ranging from certificates to (joint) doctoral degrees. There are 20,000 students enrolled in one or more courses either at the main campus or at one of the six community campuses in South-central Alaska.

UAA is an open admission university, enrolling many first generation college students. About a third of the students are minorities, many Alaska Natives from villages across the state. A large number of students are pursuing a second career, and many are in the military, taking classes at UAA during a limited time of deployment in Alaska. Many students transfer to UAA from other colleges in or outside the state, and many transfer from UAA to complete their degrees elsewhere. UAA is regionally accredited by the Northwest Commission on Colleges and Universities (NWCCU).

Curriculum development at UAA is controlled by faculty. Undergraduate courses are initiated by faculty members in the departments, and are then vetted by curriculum committees in the colleges. The Undergraduate Academic Board (UAB) reviews and approves undergraduate curriculum, while the Graduate Academic Board handles graduate courses and programs. Ultimately, the Faculty Senate approves all new courses and programs, as well as changes to existing ones.

A subcommittee of the UAB is in charge of pre-screening GER courses, before they are submitted to the UAB. In late 1990's, the subcommittee started working on revising the GER framework at UAA, partly in response to requirements from the regional accreditation body, the NWCCU. The Faculty Senate passed a motion in late 2002 that a GER integrative component be built into the new GER framework. In response to this motion, in March 2004, the UAB subcommittee submitted a proposal to require a GER capstone for all four-year programs at UAA. The proposal was approved in early 2005, and the subcommittee made available grants for faculty to develop GER capstone classes.

The development of the GER capstone framework was guided by four considerations. First and foremost was the goal of providing an integrative experience to students. Second, while the GER were not programmatic in nature, the capstone lent a programmatic nature to the GER coursework. Third, the revision was not to increase the credit requirements for degrees. Finally, the capstone was intended to provide assessment data for GER for accreditation.

Before students can register for a GER capstone class, they must complete their Tier 1 GER (basic skills) and the Tier 2 (disciplinary distribution areas). Serving as a culminating point, GER capstone classes must satisfy at least four of the five capstone requirements, and at least three of the four must be specifically addressed by the course outcomes assessment. The five capstone requirements are: 1) knowledge integration, 2) effective communication, 3) critical thinking and problem solving, 4) information literacy, and 5) quantitative perspectives. Such requirements are common among capstone models, particularly those of information literacy, communication and critical thinking, for example Portland State (Kerrigan & Jhaj, 2007), Southeast Missouri State University (Blattner & Frazier, 2004).

Faced with the challenge of developing a GER capstone course, academic programs often choose to expand the scope of existing discipline specific capstone courses to incorporate additional requirements towards GER integration, although they may also create new integrative courses (Hawthorne, Kelsch, & Steen, 2010). Adapting existing courses is a key mechanism for introducing GER capstones without increasing the credit requirements for degrees. By simply broadening the instructional goals for the class to meet capstone requirements, a discipline capstone class can serve a dual purpose. Some capstone experiences are for a homogenous group of majors, "magnets" that demand mastery of the core of the discipline, while other capstone courses are interdisciplinary or multidisciplinary in content and are places where diverse groups of students arrive to a common "mountaintop," in the terminology in (Rowles, Koch, Hundley, & Hamilton, 2004). The course described in this paper is a "mountaintop," a College wide capstone, as opposed to the departmental MIS Senior Projects capstone class (the "magnet").

3. CIS 376 – MANAGEMENT INFORMATION SYSTEMS AS A GER CAPSTONE

Business administration is one of the five most popular majors at UAA. Consistent with university policies, the College of Business and Public Policy programs has an open admission policy, but students must satisfy GPA requirements to move up to upper division (taking classes at and above the 300 level). Several programs in the College (including the BBA) are AACSB accredited. Outcomes assessment is a key component of AACSB accreditation, and is based to a large extent on data collected in the core courses (required for a majority of the students in the College).

CIS 376, Management Information Systems, was already one of the core courses in the Bachelors of Business Administration program at UAA in 2006. The class was required of all BBA majors, except for accounting majors who were required to take an Accounting Information Systems class. There are three sections of 25-35 students offered in any given semester, and the class is offered every year in both fall and spring, and occasionally in summer.

Faculty in the Computer Information Systems department realized the opportunity they were facing. CIS 376 was a good candidate for the first GER capstone class in the College of Business and Public Policy, before discipline specific capstone classes could be developed. Because the class was already required of most majors in the college, it could accomplish the GER integration goals without requiring additional credits to complete the degree. Taking advantage of one of the curriculum development grants, faculty modified the class over the summer of 2005 to meet the GER capstone requirements. The revised CIS 376 received Faculty Senate approval in spring 2006 and was effective for fall 2006.

The redesigned course is intended to be accessible for the non-MIS majors, while still challenging for MIS majors. Students are encouraged to cooperate on projects, but must submit individual work on assignments. They are free to share ideas and solutions at the concept level, as long as they put the concepts in practice on their own. Faculty have an open door policy, and help students overcome roadblocks, guiding them through the projects without actually pointing the way.

CIS 376 is at the core a typical introduction to MIS class. Topics include basic information systems components (hardware, software, databases, data networks concepts) as well as the development, acquisition and use of specific functional or cross-functional information systems. There are two exams, based on short answer essay questions and brief case studies. Weekly multiple choice quizzes about the theory concepts are delivered and graded online, and students can retake any quiz (with a different set of questions) until they master the material.

Many MIS theory concepts lend themselves to supporting knowledge integration (for example Moore's Law relates via economics concepts to the growth of the internet). The GER capstone requirements are fulfilled by a set of assignments designed around this core of MIS theory. Two types of assignments are particularly relevant: a research project on current issues related to MIS, and a series of hands-on projects where students apply theory to solving business problems. The mapping from the course assessment tools to the five GER capstone requirements is outlined in Table 1 below.

Assessment tool	KI	COM	CT	IL	QP
Exams (2)	√				
Weekly quiz	√		√		
Hands-on projects					
#1			√	√	
#2			√		√
#3	√		√		√
#4	√		√		√
Research projects					
Debate	√	√	√	√	
Website	√	√		√	
Paper	√	√	√	√	√

Table 1. Outcomes assessment mapping (KI - knowledge integration, COM - communication, CT - critical thinking, IL - information literacy and QP – quantitative perspectives.

The rest of this section outlines the essential features of the hands-on projects and the research project, focusing on how they uniquely highlight the students' achievements of the five GER capstone requirements. An example of each type of hands-on project is included in the Appendices.

Hands-on projects

There are four hands-on projects during the semester, spaced 2-3 weeks apart and closely related to lecture topics, challenging students to apply concepts and to demonstrate skills working independently. Each assignment is a set of 10-12 multiple choice or multiple answer questions, using an open time and open book format. Each of the four hands-on assignments is worth 5% of the final grade. Because of the test format, the hands-on assignments are scored automatically online, which allows students to receive immediate feedback on their work.

i) Information literacy project

The first project is designed to be relatively easy, to encourage students and to familiarize them with the format of the hands-on tests. The project is about assessing the credibility of an online business, using a variety of tools (domain registration data, Better Business Bureau data, online forums, analysis of published company policies, etc). A sample test is included in Appendix 1.

ii) Database project

The second hands-on project is due after the completion of the chapter on database management. Students are given a scenario or a large data set (10,000 records) in a flat file and are asked questions about organizing the data in a relational database. For non-MIS majors with only a rudimentary understanding of database concepts, this is a very difficult project. In fact, students find this the most challenging of the four hands-on projects, which is also reflected in the lower scores. Along with the other aspects of the course, this hands-on project has also been modified over the years to address the low scores. Because database concepts are so difficult for non-MIS majors, the only workaround has been to offer a make-up test, with a different scenario, which generally leads to much improved test scores. Revisiting database concepts in hands-on 3 (below) is another way to ensure that students get a better understanding of the topic. A sample test is included in Appendix 2.

iii) Data mining project

The third hands-on project is due after the completion of the chapter on business intelligence. The lectures cover several tools, including online analytical processing, RFM (recency, frequency and monetary) analysis and market basket analysis, and this project is a rather straightforward application of the techniques.

As part of the data processing, students may need to revisit database concepts yet again. For example, they may need to normalize a flat file to be able to conduct some of the more complex queries. If the data set includes records of transactions with multiple products, normalization may be required to calculate the number of transactions for a given sales person. Although the same goal could be accomplished with simple SQL statements, the students are not MIS majors and have not had sufficient background to carry out such tasks. The assignment does not require a particular approach, and students are free to use SQL, but most students find it easier to normalize the database (a process they have learned about) and then use pivot charts on the normalized database tables.
Having had additional exposure to database concepts by this point, as well as theoretical exposure to the data mining techniques, students typically do well on this project. An in-class workshop gives students some general guidelines and an opportunity to ask questions, especially about the database concepts. Feedback from graduates is that the skills learned in this project are directly applicable in many business jobs. A sample test is in Appendix 3.

iv) Forecasting and decisions support project

The last hands-on project is typically due at the very end of the semester, and is less connected with the lecture. Instead, it is a diverse set of questions with direct relevance for making business decisions. The first part of the project deals with forecasting, with a number of scenarios of increasing difficulty and having to do with break even analysis. Progressively, students must calculate the growth rate that will keep a company from running out of cash, then with a cash reserve, and finally with a cash reserve even in the presence of inflation. A second component of the project is an optimization problem, using the Microsoft Excel package Solver. A final component is a rudimentary decision support system for choosing among a set of health insurance plans. Many students have not yet had to make choices

of this nature, and are not familiar with deductibles and out of pocket payments. Although many of the concepts tested in this assignment are not covered in the lecture, an in-class workshop gives students the opportunity to ask clarification questions and provides a general overview of the problem. A sample test is in Appendix 4.

Together, the four hands-on projects test students on four of the five areas of the GER capstone: knowledge integration, critical thinking and problem solving, information literacy, and quantitative perspectives. These projects are highly structured in the types of questions students are asked to solve, and cannot be used to assess the communication skills. In contrast, the research projects are open-ended and manage to assess all of the five areas, including communications.

Research project on current issues in MIS

The research project has three separate components: an oral presentation (using a debate format), a website (which is also used as the presentation tool for the debate) and a formal research paper. The research project counts for 25% of the final grade, with 13% for the paper, 5% for the website and the reminder for the oral debate.

The topics for the research papers change regularly to reflect current topics in MIS. Topics include secondary uses of data, employee monitoring, using Facebook for screening potential hires, and mandating subsidies for broadband access.

Students choose their own topics from the list, sometimes expecting that there is a "right answer" to the debate. They soon realize that there are no definitive answers to the debate question – and are horrified to learn that they might have to defend the side of the debate they do not agree with. For the oral debate, students must prepare both sides. The side they actually get to defend is decided by a coin toss, right before the actual debate.

Students work in pairs on their research question. They can share sources and exchange ideas, but must prepare websites and write research papers independently. Moreover, the oral debate pits the two students against each other in front of the class. As mentioned above, students must prepare both sides of the debate and they end up debating one side as decided by a coin toss. The website must include rich multimedia and must be suitable for presentation in front of a medium size audience (25-35), but it must also be structured to allow for self-paced browsing, guiding the reader and providing sufficient information for somebody who has not seen the oral debate.

The research paper must include a balance of arguments on both sides of the issue, followed by a critical analysis and a personal position point, written in the first person. Throughout the research project, students must choose strong arguments and must provide evidence from reputable sources. Papers are rather extensive, 2500-3500 words, and are graded using strict standards for presentation, formatting and contents.

To research the topics, students must demonstrate information literacy skills. Formulating arguments requires critical thinking and at times quantitative skills. The three components of this project (oral debate, website and paper) make it a heavily communications-based assignment. As such, the research paper is uniquely able to assess all five areas of the GER capstone requirements – but it requires considerable efforts both on the part of the students, for research and writing, and on the part of the instructor, for grading.

4. DISCUSSION

CIS 376 has been taught as a GER capstone class since fall 2006, but the assessment tools have evolved. Since spring 2007, the number and nature of the assessment tools have been unchanged, allowing a longitudinal comparison of student achievement levels.

The assessment data (Fig. 1) shows the percentage of students who achieve a passing grade (70% or higher) on each type of assessment. Because tools test various types of skills, it is possible to infer the overall skill levels of the students across the GER capstone areas. Numerical data are included in the table in Appendix 5.

Over the course of the five years, a majority of the students (75% or better within the last three years) achieve passing scores on all of the assignments (except for the Hands-on 2 on database management, which is rather technical, and which is not part of the GER

capstone outcomes). There is a slight trend up in the scores, although the volatility of the scores and the relatively low number of data points do not lend statistical significance to this trend.

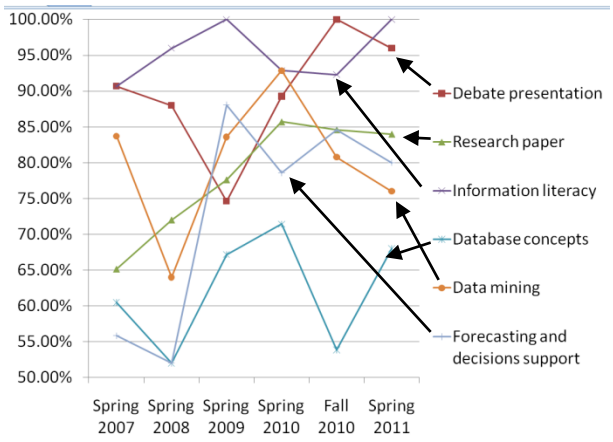


Fig. 1. A majority of students achieve passing scores across the entire spectrum of assessment instruments. The lowest curve is for Hands-on #2: Database concepts.

5. CONCLUSIONS

CIS 376 is a core MIS class required for non-accounting business majors in the College of Business and Public Policy at the University of Alaska Anchorage. The class was converted to a GER capstone format, by ensuring that assessment tools track student performance across five dimensions: 1) knowledge integration, 2) effective communication, 3) critical thinking and problem solving, 4) information literacy, and 5) quantitative perspectives. Although the class is not required for accounting majors, many choose to take it as their GER capstone. Assessment data collected over the past five years indicate that a majority of students achieve passing scores (70% or better) across the five dimensions. Within the last three years, 75% of students achieved passing scores on the five GER dimensions.

6. REFERENCES

Abrahams, A. S., & Singh, T. (2010). An Active, Reflective Learning Cycle for E-Commerce Classes: Learning about E-commerce by Doing and Teaching. *Journal of Information Systems Education , 21* (4), 383-390.

Bers, T. (2000). Assessing the Achievement of General Education Objectives: A College-Wide Approach. *The Journal of General Education , 49* (3), 182-210.

Blattner, N. H., & Frazier, C. L. (2004, July–August). Assessing General Education Core Objectives. *Assessment Update , 16* (4), pp. 4-6.

Brock, P. A. (2004, January-February). From Capstones to Touchstones: Preparative Assessment and Its Use in Teacher Education. *Assessment Update , 16* (1), pp. 8-9.

Claus, B. A., & Hawkins, S. T. (2007). Indiana State Introduces Liberal Studies Capstone Course in FCS. *Journal of Family and Consumer Sciences , 99* (1), 33-37.

Fernandez, N. P. (2006, May-Jun). Integration, Reflection, Interpretation: Realizing the Goals of a General Education Capstone Course. *About Campus , 11* (2), pp. 23-26.

Hawthorne, J., Kelsch, A., & Steen, T. (2010, March 17). Making general education matter: Structures and strategies. *New Directions for Teaching and Learning , 2010* (121), pp. 23-33.

Henscheid, J. M., Breitmeyer, J. E., & Mercer, J. L. (2000). *Professing the disciplines: An analysis of senior seminars and capstone course.* Columbia, SC: National Resource Center for the First-Year Experience and Students in Transition.

Janicki, T. N., Fischetti, D., & Burns, A. T. (2007). Incorporating Real World Projects and Emerging Technologies into One MIS Capstone Course. *Information Systems Education Journal , 5* (24), 1-8.

Kerrigan, S., & Jhaj, S. (2007, April 1). Assessing General Education Capstone Courses: An In-Depth Look at a Nationally Recognized Capstone Assessment Model. *peerReview* , 13-16.

McGann, S., & Canili, M. (2005). Pulling it all Together: An IS Capstone Course for the 21st Century emphasizing experiential and conceptual aspects, soft skills and career readings. *Issues in Information Systems , VI* (1), pp. 391-397.

Rowles, C. J., Koch, D. C., Hundley, S. P., & Hamilton, S. J. (2004, January-February). Toward a Model for Capstone Experiences: Mountaintops, Magnets, and Mandates. *Assessment Update , 16* (1), pp. 1-2, 13-15.

Wilson, E., Jones, K., Sullivan, B., Crank-Lewis, D., Guneyli, V., Zeizer, J., et al. (2008). *General Education Capstone Assessment Report 2008.* Cottleville, MO: St. Charles Community College.

Editor's Note:

This paper was selected for inclusion in the journal as the ISECON 2011 Best Paper. The acceptance rate is typically 2% for this category of paper. This is based on blind reviews from six or more peers, including three or more former best papers authors who did not submit a paper in 2011.

Appendix 1. Hands-on project #1: Information Literacy

1. According to the www.sellyourmiles.com web site, the physical address for the company is in
   a. Alaska
   b. California
   c. Florida
   d. New York
   e. No address is given

2. According to WHOIS information, the registrant for the www.sellyourmiles.com web site is
   a. John Allen
   b. Martin Ferrari
   c. Donna Wilson
   d. Gabriel Wilson
   e. Sell Your Miles, Inc.

3. Search the web site and locate a contact email address. You may be surprised that the address is not something@sellyourmiles.com, but something different, which will point you towards a different web site. You may verify that the registrant for this second web site is the same as for www.sellyourmiles.com. On this second company web site, locate the Certificate of Registration from the State government. On the certificate, locate the official company name:
   a. World Wide Travel
   b. World Wide Travel, Inc.
   c. World Wide Travel Services
   d. WWT Consulting
   e. WWT, Inc.

4. Do a Google search for BBB and the state where the registrant of the two web sites is located. On the list of Google results, locate the BBB office that services the city where the registrant is located. At that web site, do a search for the company name you found in #3 above. The company ID on the BBB site is:
   a. 13042635
   b. 40000104
   c. 13058553
   d. 13074883
   e. 13142441

5. According to the BBB site, the company has had a BBB record since
   a. 2/1/2000
   b. 10/20/2000
   c. 4/24/2006
   d. 4/12/1975
   e. No date is available

5. The company rating on the BBB site above is
   a. A
   b. B
   c. C
   d. D
   e. F

6. According to the BBB web site, this company rating is
   a. An exemplary rating. This means that nothing in our files causes us to have any doubt about the company's reliability.
   b. An excellent rating. A company with this rating may not rate higher because of a greater number of rate-lowering factors, but we do not consider them to be factors that would likely adversely affect consumer transactions.
   c. A very high rating. A company with this rating would not have a significant number of complaints or other considerations that could pose a problem to consumers.
   d. A good rating that still implies reputability. The rating may relate to length of time in business, a past problem that's been corrected, or something else that does not cause problems for consumers. We believe a company with this rating would generally conduct business and respond to any complaints satisfactorily.
   e. We strongly question the company's reliability for reasons such as that they have failed to respond to complaints, their advertising is grossly misleading, they are not in compliance with the law's licensing or registration requirements, their complaints contain especially serious allegations, or the company's industry is known for its fraudulent business practices.

7. Search now on the BBB web site for the business associated with the website www.sellyourmiles.com (you might need to try different search types to make sure you are using the correct name). According to the web site
   a. The business is listed and has a better rating than the company you searched for in #4.
   b. The business is listed and has the same rating as the company you searched for in #4.
   c. The business is listed and has a lower rating than the company you searched for in #4.
   d. The business is listed, but not rated.
   e. The business is not listed.

8. According to the www.sellyourmiles.com web site, selling miles is
   a. Legal in all 50 states
   b. Legal in most of the US states
   c. Legal in few of the US states
   d. Legal in only one state
   e. Illegal

9. Do an internet search and read about the legality of selling miles, then answer the following:
   a. Selling miles is prohibited by federal laws
   b. Selling miles is prohibited by state laws in most states
   c. Selling miles is legal, but not in as many states as the site advertises
   d. Selling miles is legal in most US states, but prohibited by other means
   e. Selling miles is legal and a totally legitimate transaction

10. Based on your findings so far, a reputable business in need of travel arrangements should
   a. Use this site with confidence, any time
   b. Use the site only for domestic (US) travel
   c. Use the site only to travel to and from states where the service is legal
   d. Use a similar service, but from a more reputable business with a higher BBB rating
   e. Avoid using the services as well as the web site

Appendix 2. Hands-on project #2: Database Management

The second hands-on project deals with database design.

You are managing a small school for airline pilots and you need to keep track of aircraft airtime (for maintenance schedules) and pilot flight hours (for certification). For simplicity of the problem, each aircraft can only accommodate exactly one pilot (but cannot fly without a pilot). All pilots are certified to fly on any of the aircraft you have. You are designing a database to manage this data.

Start by laying out an E-R diagram based on the requirements above. Then answer the questions below.

1. Which of the following should be tables in the database? (check all that apply)
   a. Pilots
   b. Aircraft
   c. Total aircraft airtime
   d. Flight durations
   e. Flights

2. Which of the following would be an appropriate primary key for the Pilots table? (check all that apply)
   a. First name
   b. Last name, First name
   c. Weight
   d. SSN
   e. Flight time

3. Which of the following would be an appropriate primary key for the Aircraft table? (check all that apply)
   a. Aircraft type (model)
   b. The combination of aircraft model and serial number
   c. Aircraft weight
   d. Aircraft owner
   e. Automatically generated unique key

4. What is the most appropriate relationship between pilots and flights (think about actual facts, not about database tables)?
   a. One to one
   b. One pilot to many flights
   c. One flight to many pilots
   d. Many pilots to many flights
   e. There is no relationship

5. How would you accomplish the relationship in #4 above?
   a. Use a foreign key in the Pilots table. The foreign key is the primary key of the Flights table.
   b. Use a foreign key in the Flights table. The foreign key is the primary key of the Pilots table.
   c. No need to do anything, because there is no relationship.
   d. Use an intersection table between Pilots and Flights.
   e. Pilots and Flights go in the same table, because this is a one-one relationship.

6. What is the most appropriate relationship between pilots and aircraft (think about actual facts, not about database tables)?

a. One to one, because there can be only one pilot per aircraft

b. One pilot to many aircraft (one pilot will fly on many aircraft, in turn)

c. One aircraft to many pilots (many pilots will fly on any one aircraft, in turn)

d. Many pilots to many aircraft (many pilots, each one will fly on many aircraft)

e. There is no relationship

7. How would you accomplish the relationship in #6 above?

a. Use a foreign key in the Pilots table. The foreign key is the primary key of the Aircraft table.

b. Use a foreign key in the Aircraft table. The foreign key is the primary key of the Pilots table.

c. No need to do anything, because there is no relationship.

d. Use an intersection table between Pilots and Aircraft.

e. Pilots and Aircraft go in the same table, because this is a one-one relationship.

8. Which of the following fields can be part of the Pilots table? (check all that apply)

a. Pilot name

b. Flight duration

c. Aircraft ID for the flight

d. Pilot age

e. Pilot weight

9. What is the best way to track pilot flight time (the total number of hours a pilot has flown)?

a. Use a field in the Pilots table, and update this field after each flight

b. Use a field in the Flights table, and set up a query to calculate total time

c. Use a field in the Flights table and update this after each flight

d. Use a field in the intersection table of Pilots and Flights

e. Set up a separate table with the Pilot Flight Time

10. You change your mind about the requirements, and decide that you need to accommodate multiple pilots per aircraft in your database design. In fact you discover that the number of pilots could be very high – a whole group might take off at the same time on one plane, and take turns piloting while up in the air. What changes do you need to make to accommodate this?

a. Easy, you do not need to make any changes to accommodate multiple pilots.

b. You need to add another field in the Flights table.

c. You need to add another field in the Pilots table.

d. You need to add one or more tables.

e. You cannot accommodate such a request, no matter what you do.

Appendix 3. Hands-on project #3: Data Mining

The third hands-on project deals with data mining. You will need to process data into information that might be useful in making business decisions.

The file "Spring 2009.txt" contains data about purchase transactions for a small Alaskan company. The fields are separated by tabs, and contain in order, the transaction year, month and day, then ID of the salesperson who made the sale, the ID of the customer who made the purchase, the ID of the transaction, the product ID and the sales price. The questions below involve either revenues generated (the sum of the sales prices) or volumes sold.

This packet includes two files (see below):
"Spring 2009.txt" is the data set
"Data mining hands-on.ppt" is a file with directions on setting up your queries

In answering the questions, you might find it useful to import the data in a database, and to run some queries to help you get to the answers. You might also need to use a spreadsheet to process the results from the database queries, although you can also do that with a hand calculator. In getting answers to most of the questions, you might find using pivot tables or pivot charts as helpful.

You are not required to submit any of the files you used, but only to answer the questions. As with previous hands-on projects, you can only submit the answers one time. Answers are omitted for some questions, to save space; the numerical answers include a list of ten randomly generated possible answers, to reduce the chance of a random guess.

1. What type of relationship is there between sales persons and customers, based on the data in the file?
a. 1:1
b. 1:N
c. N:M
d. Cannot specify based on the data in the file
e. It depends on the user's point of view

2. Which customer generated the highest total revenue over the entire transactions window?
a. Customer 31
b. Customer 32
c. Customer 33
d. Customer 34
e. Customer 35

3. What is the value of the highest average revenues per transaction among all customers?

4. Which customer is closest to a 513 in the RFM analysis?

5. Which sales person should be encouraged to share best practices with the others?

6. Which is the best month of the year in terms of total revenues?

7. What is the support value for the two products that are the best candidates for bundling (and should be marketed together)?

8. What is the lowest support value, for the two products that are most likely to be substitutes for each other?

9. What is the best selling product (highest volume)?

10. For individual customer-salesperson relationships, what is the largest number of items any customer purchased from any one salesperson?

Appendix 4. Hands-on project #4: Decision Support

CIS 376 – Management Information Systems
Hands-on project -- part 4

The fourth hands-on project deals with business forecasting and decisions support systems. You are encouraged to create a spreadsheet to answer the questions below. You do not need to submit the spreadsheet.

It is December 2010. You are planning to start a small airline in bush Alaska. The grand opening is January 2011.

You have $250,000 startup capital. You have fixed payments to make for your airplanes, staff and office space, at $80,000 per month. Your variable costs are $120 per passenger and you charge an average of $180 per passenger.

You expect to have 800 passengers in January 2011 and you expect a uniform rate of increase in this number, some X % month to month. Set up a spreadsheet so that you can calculate your cash balance at the start and end of each month, given the number of passengers for that month. Link your cells to allow you to specify the month to month growth rate X% in a single cell.

Answers are omitted for some questions, to save space; the numerical answers include a list of ten randomly generated possible answers, to reduce the chance of a random guess.

1. In any given month, how many passengers do you need to be profitable? (to make enough money during that month just to cover your expenses for the month)

2. If you start with 800 passengers in January and the growth rate X is zero (no growth), what is the first month at the end of which you will have a negative cash balance?

3. Calculate the smallest rate of increase in the number of passengers per month X, to make sure you do not run out of cash at any time (you always end up with some cash left at the end of each month). You might want to use goal seek (try various starting values for X to help goal seek to converge).

4. Calculate the rate of increase in the number of passengers per month X if each month you must maintain a cash reserve (at the end of the month) of at least 10% of the current month's expenses.

5. Redo the previous question if the inflation rate is at 1% per month (assume that all your expenses increase 1% per month).

6. Faced with high demand on one of your routes, you charter a larger airplane for a one-time flight. You are able to sell first class tickets at $1200 per person (but will only be able to sell at most 10 tickets), economy tickets at $400 per person or you can carry cargo for $1.20/lb.

Each first class passenger comes with 600 lbs of weight (luggage, passenger and in-flight meals) and each economy passenger weighs in at 300 lbs (including luggage, passenger and in-flight meals). According to FAA specifications, the aircraft can carry no more than 25,000 lbs, including both passengers and the cargo weight.

Additionally, you need to figure out space limitations on board. Each first class seat takes 30 sq. ft. of space and each economy seat takes 13 sq. ft. You can pile up cargo 50 lbs/sq. ft. The total floor space in the plane is 1000 sq. ft., which needs to accommodate all the passengers and the cargo. For simplicity, you do not need to have full rows of seats (i.e., you could have 17 seats on the whole plane) and do not need to worry about aisle space.

Use solver to figure out how many passengers and how much cargo you can carry to maximize your revenue for the flight. Make sure you consider all the conditions you need for solver. The program does not understand the realities of life :).

How many pounds of cargo will you need to carry to achieve this maximum?

7. You also need to purchase insurance for your employees. The three options available are given in the following table.

Plan A: Monthly charge: $30, Deductible $1300, Out of pocket maximum $5000
Plan B: Monthly charge: $60, Deductible $500, Out of pocket maximum $2000
Plan C: Monthly charge: $150, Deductible $200, Out of pocket maximum $750

Employees may elect to participate in any one of the three plans, or to opt out of insurance totally. Employees who select a health plan pay the monthly charges for all the twelve months per year; no fractions of a year are allowed.

We use the term "medical care expenses" for the amount billed by the medical providers. This amount is paid in part by the patient, with the balance covered by the insurance. "Patient costs" are the charges incurred by the patient (which include monthly charges and the patient's portion to the medical providers' bill).

As employees incur medical expenses, they pay for part of the medical care and the insurance pays for the balance. Given a certain cost of medical care expenses, the relative share of the employee and the insurance company are as described below. The employee must pay for the full cost of the medical care until the expenses exceed the Deductible. For the medical care expenses in excess of the Deductible, the plan pays for 80% of the expenses, and the employees are responsible for the remaining 20%. Finally, once the expense incurred by the employee reaches the Out of Pocket, the plan pays for 100% of the medical charges. The Out of Pocket charge does not include the Monthly Charges, nor the Deductible. Both the deductible and the out of pocket amounts are for the year; at the end of the year, the patient needs to start over and meet the deductible and out of pocket anew.

If the employee selects Plan B, what is the maximum amount of patient costs they will spend on health care by the end of the year (including Monthly Charges and their portion of the medical care, not covered by insurance)?

8. How much do the medical care expenses need to be (at least) for the employee to have to pay the maximum figure, as in the question above?

9. At what cost of medical care is the employee paying the same amount whether using insurance (the lowest cost plan) or paying for medical care entirely on her own? You might want to use goal seek for this question.

10. An employee expected the cost of medical care for the following year to be $7,000. Based on this assumption, the employee chose the plan with the lowest expenses for that level of medical care. If the actual expenses are in fact $9,000 at the end of the year, this choice of plan might not be the best anymore. How much worse off is the employee because of the error in estimating medical expenses? (what is the difference between what the employee would have paid under the best plan and what she is actually paying in the scenario above?)

Appendix 5. Assessment data 2007-2011

Semester	Enrollment	Debate presentation	Research paper	Hands-on #1: Information literacy	Hands-on #2: Database concepts	Hands-on #3: Data mining	Hands-on #4: Decision support
Spring 2007	43	90.70%	65.12%	90.70%	60.47%	83.72%	55.81%
Spring 2008	25	88.00%	72.00%	96.00%	52.00%	64.00%	52.00%
Spring 2009	67	74.63%	77.61%	100.00%	67.16%	83.58%	88.06%
Spring 2010	28	89.29%	85.71%	92.86%	71.43%	92.86%	78.57%
Fall 2010	26	100.00%	84.62%	92.31%	53.85%	80.77%	84.62%
Spring 2011	25	96.00%	84.00%	100.00%	68.00%	76.00%	80.00%

Table A.5. Assessment data for all five years CIS 376 was taught as a GER capstone with a consistent set of assessment tools. The numbers indicate the percentage of students who achieved 70% or better on each assessment tool.

Is Student Performance on the Information Systems Analyst Certification Exam Affected By Form of Delivery of Information Systems Coursework?

Wayne Haga
haga@mscd.edu

Abel Moreno
morenoa@mscd.edu

Mark Segall
segall@mscd.edu

Department of Computer Information Systems
Metropolitan State College of Denver
Denver CO 80217

Abstract

In this paper, we compare the performance of Computer Information Systems (CIS) majors on the Information Systems Analyst (ISA) Certification Exam. The impact that the form of delivery of information systems coursework may have on the exam score is studied. Using a sample that spans three years, we test for significant differences between scores obtained on three of the areas of the ISA exam by CIS majors who completed the coursework via classroom delivery with those who completed the coursework via online delivery. Results from the study are analyzed and conclusions discussed. Opportunities for further study are proposed.
Keywords: online delivery, Information Systems Analyst Certification Exam, Core Information System Areas

1. INTRODUCTION

Third-party feedback is a fairly unbiased option for the assessment of academic programs. Our CIS program has been using the ISA exam (McKell et al 2005) for assessing our program outcomes and objectives for several years. While helping us meet the internal assessment expectations for programs offered at our institution, the ISA exam results are also used in our ABET (Accreditation Board for Engineering and Technology) accreditation process. Further, students may benefit from the exam score as those scoring fifty percent or higher may attain professional certifications (ICCP 2011). As in the case at most institutions of higher learning, our CIS curriculum is delivered in a traditional classroom setting and, with a few exceptions, in an online format. Given the relevance that the ISA exam has in our program, we want to explore whether or not the type of delivery has an impact on the ISA exam score. The paper is

organized as follows. In the next section, we describe the ISA exam and its relationship to the Information Systems (IS) curriculum as required by ABET (2007). The next section describes the curriculum areas being considered and the study's methodology. Results are presented next. In the last section, results are discussed and conclusions offered.

2. THE INFORMATION SYSTEMS ANALYST EXAM

An important benefit from using the ISA exam for assessment purposes is that the exam content maps to the IS2002 model curriculum for undergraduate Information Systems education (Gorgone et al 2003). Further, there is a defined linkage between the IS2002 model curriculum learning units and the six IS core areas defined by IS curriculum ABET (2007) accreditation guidelines [Landry et al 2006], *i.e. hardware and software, modern programming language, data management, networking and telecommunications, analysis and design and role of IS in organizations* [ABET 2007]. Thus, ISA exam scores are useful for: 1) meeting institutional assessment requirements, 2) ABET accreditation of our program, and as indicated in the previous section 3) offering our students an opportunity to attain professional certifications.

The ISA exam is jointly administered by the office of the Institute for Certification of Computing Professionals (ICCP) and the Center for Computing Education Research (CCER) – a division of the ICCP Education Foundation. The ISA exam has been designed for graduating seniors from 4-year undergraduate Information Systems degree programs. A 50 percent or higher score in the approximately 3-hour long ISA examination (can be split into two 105 minute exams), plus an undergraduate degree, qualifies an individual to receive the title of ISA-Practitioner. A 70 percent or higher score is specified as ISA-Mastery level. A holder of the ISA certification is automatically enrolled into the ICCP Recertification program. When a student takes this examination at our College, they are given the option of paying for the credential right after the score is received and the examination is passed (50 percent or higher). The certificate is mailed to the student based on the ICCP receiving confirmation directly from our College of the student having graduated successfully from our CIS program. (ICCP 2011)

The exam-taking mechanics is as follows. The student first registers for the exam and receives a password. The exam is delivered over the internet to a proctored testing site. The exam requires about three hours to complete and includes 258 questions. The exam score is reported upon completion of exam. Table 1 shows a summary of exam results for our institution over the three-year period considered in this study (see appendix).

3. IMPACT OF TYPE OF DELIVERY ON ISA EXAM SCORES

Online delivery of courses has advantages and disadvantages. Working students can take an online course at times that are convenient to them, however online courses can be more difficult as seen by their higher dropout rates. Attrition rates are generally higher in courses delivered online. Terry (2001) reported higher attrition rates in Finance and Statistics online MBA courses (37%) versus Campus courses (17%). The dropout rate for one online MBA program as 43% compared to 11% for the campus based program (Patterson and McFadden 2009).

Online delivery is not always viewed favorably by students. Davis et al (2010) report that only 37% of students gave high rates of effective or vary effective to "pure online" courses, compared to 59% to "hybrid" and 76% for "on-ground with online supplements". There is also the question of whether all students are suited to succeed in online courses. For example, "Mid-range" students typically earn grades 10-15% lower in online courses (*Marold and Haga 2003*).

As indicated previously, most of our program is offered in an online format. We decided to investigate whether or not the type of delivery of the coursework has any impact on the ISA exam score. A sample of 131 students was used in our study over a three-year period. The average ISA exam score for the sample was 48.2 with a standard deviation of 12.3. The average student age was 30.1. The male/female ratio was 68%:32%.

Characteristics of our courses delivered online are as follows. Students complete similar or identical assignments as students taking the same course in a traditional classroom setting. Online students take all of their tests on campus.

Of the six IS core areas defined by IS curriculum ABET (2007) accreditation guidelines, we offer online versions of courses in the areas of *modern programming language, data management, and networking and telecommunications,* and thus our study will be focused only on those areas.

4. RESULTS

The authors first analyzed the overall ISA score compared to the total number of CIS courses the student completed online. Figure 1 shows a scatter plot of this comparison. A positive correlation of .267 (p-value=.002) indicates that students' overall ISA score actually increased with more classes taken online. The equation is ISA score = 45.4 + 2.00 number online (R^2 = 7.1%).

A second analysis was run on the overall ISA score with just the required core classes in our CIS program. The core classes would logically be the ones that would have the largest impact on the ISA exam since the core classes cover required concepts for the IS 2002 model curriculum on which the ISA exam is based. Of the seven required core CIS classes, only five are offered online.

Figure 2 shows a scatter plot of the student's composite ISA score versus the number of core CIS courses taken online. Again, there was a positive correlation, which increased slightly to .271 (p-value=.002) indicating that students overall ISA score also increased with more of the core classes taken online. The equation is ISA = 46.5 + 3.28 number core online (R^2 = 7.3%).

The analysis was next broken down by ISA exam sub-scores. For each sub-score, a t-test was run to determine if taking the course covering the majority of the material for that area was taken online or in the classroom. As indicated in the previous section, only the *modern programming language, data management, and networking and telecommunications* sub-scores were analyzed as the department does not offer online versions of courses in the other three ISA exam subcategories. These three courses are also required for the CIS major.

For the Programming Language sub-score, there were 12 students that completed our CIS 3145 – Business Application Development with Visual Basic course online, and 59 that completed the classroom version. The sample size is considerable smaller for this test. This can be attributed to the fact that prior to the latest major curriculum revision, students had a choice of several classes to meet their programming requirement. Currently all students are required to take CIS 3145 - Visual Basic as part of the core. This course also tends to have a lower success rate and since students that did not successfully complete the course in the first attempt were again removed from the analysis, this likely contributed to the smaller sample size.

Table 2 summarizes the results. The online students mean was over six points higher than the classroom students, but the difference again was not statistically significant.

Table 2. Programming languages sub-score analysis

Delivery	N	Mean	St dev.	t-test
Classroom	59	40.7	13.5	t= -1.05
Online	12	47.2	20.8	p-val=.317

For the ISA Data Management sub-score, there were 18 students that completed CIS 3060 – Database Management Systems online, and 91 that completed the classroom version. Students that did not successfully complete the course in the first attempt were removed from the analysis. Table 3 summarizes the results. The online students mean was over 3 points higher than the classroom students, but the difference was not statistically significant.

Table 3. Data Management sub-score analysis

Delivery	N	Mean	St dev.	t-test
Classroom	91	47.1	14.5	t= -1.12
Online	18	50.6	11.7	p-val.=.273

For the ISA Networking sub-score, there were 18 students that completed CIS 3230 – Networking and Telecommunications Systems online, and 90 that completed the classroom version. Students that did not successfully complete the course in the first attempt were again removed from the analysis. As shown in Table 4, the online students mean was over six points higher than the classroom students, but the difference was still not statistically significant.

Table 4. Networking sub-score analysis

Delivery	N	Mean	St dev.	t-test
Classroom	90	40.6	18.0	t= -1.30
Online	18	47.2	19.8	p-val. =.205

5. CONCLUSIONS

Based on the results obtained, no significant difference was observed between the ISA exam scores of those students who completed the coursework online and those who completed the coursework in a traditional classroom setting. This is an important preliminary finding on the comparability of online versus regular course delivery. We effectively used the ISA exam to show that student outcomes are being met with both delivery methods.

However, we could establish that there seems to be a small, but significant positive relationship between the number of courses taken by a student in an online format and his/her overall ISA exam score. This could be an indication that the skills needed to succeed in online courses are also useful for success in Information Systems.

O'Neil (2009) discusses the student characteristics in an online environment. This author used an 18 question checklist to compare students in three groups of students taking online courses: Seniors, Freshmen, and Freshmen in a "First-year Experience Campus". The last group is considered an 'unprepared' group. The unprepared group was more likely to say No to the questions:

- "I am not intimidated by using technology for learning",
- "I am an independent learner", and
- "I easily understand what I read".

Both Freshmen groups were more likely to respond no to the questions:

- "I am a self-starter"
- "I am open to working in an un-structured setting"

Seniors with more experience taking classes in general, and online courses in particular, will perform well in online courses because they have the skills to do so.

Student in our department can take online or regular classes and will self-select the type of course they prefer. Thus the students with the skill to do well in an online course, independent, self-starters, able to read and learn in un-structured environments, and not intimidated by technology, should also do very well on the ISA exam.

Future studies can look at additional factors that lead to success in CIS programs and online courses, such as overall skill levels as measured by GPA scores, age, professional experience, motivation and learning styles. We can also look at the performance in the specific courses and in the corresponding ISA core area scores as they relate to the online and regular classroom delivery modes.

6. ACKNOWLEDGEMENTS

We would like to thank Jeffery Landry, Harold Pardue, and Bart Longenecker for assistance with the ISA data.

7. REFERENCES

ABET Computing Accreditation Commission, (2007). *Criteria For Accrediting Computing Programs.* Approved March 17, 2007, ABET, Inc., Baltimore, Maryland. Retrieved June 6, 2011 from http://www.abet.org

Davis, G., Kovacs, P., Scarpino, J., and Turchek, J. (2010). Determining the Effectiveness of Various Delivery Methods in an Information Technology/Information Systems Curriculum. *Information Systems Education Journal, 8* (32).

Gorgone, J., Davis, G., Valacich, J., Topei, H., Feinstein, D., and Longenecker, H. (2003). Model Curriculum and Guidelines for Undergraduate Programs in Information Systems, *Database for Advances in Information Systems*, Winter 34 (1).

Institute for Certification of Computing Professionals (ICCP) Information System Analyst Certification. (2011). Retrieved June 6, 2011 from http://www.iccp.org/#8

Landry, J., Pardue, H., Reynolds, J., and Longenecker, H. (2006). IS 2002 and Accreditation: Describing the IS Core Areas in Terms of the Model Curriculum. *Information Systems Education Journal*, 4 (21).

Marold, K., & Haga, W. (2003). The emerging profile of the on-line learner: Relating course performance with pretests, GPA, and other measures of achievement. *Proceedings of*

the Information Resource Management Association (IRMA) Conference (pp. 248-251). Idea Group Publishing.

McKell, L., Reynolds J., Longenecker H., Landry J., Pardue H. (2005). Information Systems Analyst (ISA): A Professional Certification Based on the IS2002 Model Curriculum, *The Review of Business Information Systems*, Summer (9:3), 19-24.

O'Neil, T.D. (2009). The success of the unprepared student in the distance education classroom in higher education. Proceeding ISECON, V26, §2324.

Patterson, B., McFadden, C. (2009). Attrition in Online and Campus Degree Programs. *Online Journal of Distance Learning Administration*. 12(2).

Terry, N. (2001). *Assessing Enrollment and Attrition Rates for the Online MBA*. T H E Journal, 28(7), 64-68.

Appendices and Annexures

Table 1
MSCD Exam Summary

Core Area	# of Items	All Schools	MSCD
Hardware and Software	10	41.8	47.3
Modern Programming Language	12	40.9	44.8
Data Management	44	46.0	51.2
Networking and Telecommunications	12	45.1	45.3
Analysis and Design	108	47.5	51.2
Role of IS in Organizations	72	52.0	56.3

Figure 1. ISA score versus Number CIS courses taken online (r=.267, p=.002)

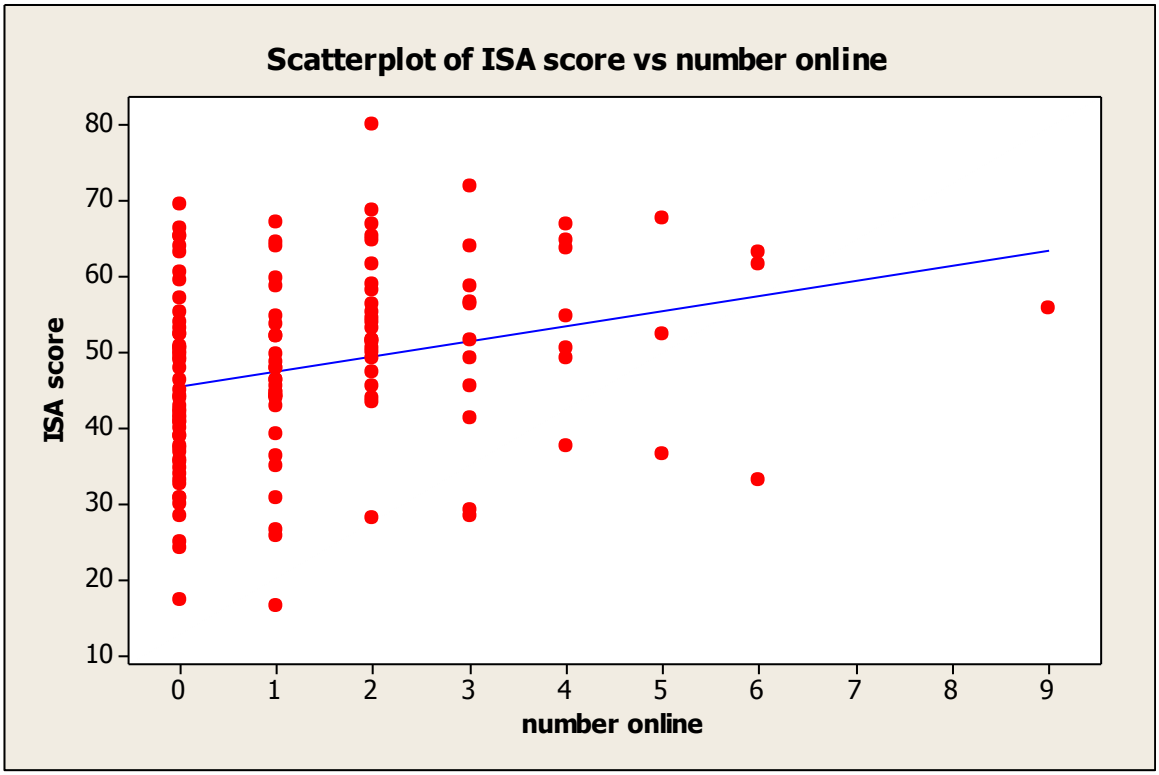
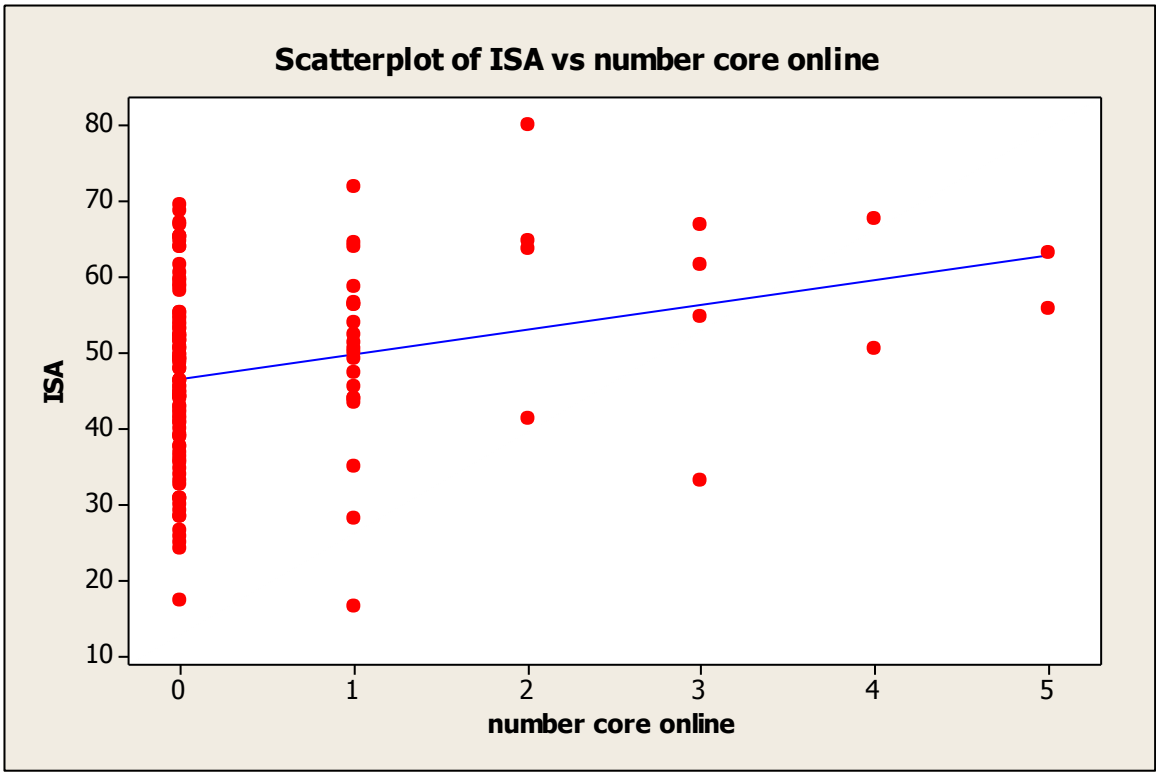


Figure 2. ISA score versus Number core CIS courses taken online (r=.271, p=.002)



CIS Program Redesign Driven By IS2010 Model: A Case Study

Ken Surendran
ksurendran@semo.edu
Department of Computer Science

Suhair Amer
samer@semo.edu
Department of Computer Science

Dana Schwieger
dschwieger@semo.edu
Department of Accounting

Southeast Missouri State University
Cape Girardeau, MO 63701-4799, USA

Abstract

The release of the IS2010 Model Curriculum has triggered review of existing Information Systems (IS) programs. It also provides an opportunity to replace low enrollment IS programs with flexible ones that focus on specific application domains.

In this paper, the authors present a case study of their redesigned Computer Information Systems (CIS) program that comes into effect in Fall 2012. Of the four tracks in the program, two are aimed at students interested in two diverse application domains: Business Administration and Graphics Communications (Multimedia). The authors describe the context and design constraints in choosing the tracks, as well as the process used in designing their flexible CIS program with consideration made for ABET accreditation. They also discuss how well the core courses in the redesigned CIS program fare against the IS2010 Model recommendations. Further, for the CIS Business track, they illustrate how the courses collectively satisfy the IS Body of Knowledge recommended in the Model document. In addition, they map the domain-related courses in that track onto the different levels of a two-dimensional learning taxonomy to help design the assessments in those courses. They also provide an outline of the Multimedia track they developed using the same process.

Keywords: IS2010 Model, Flexible IS Program, IS Tracks, Intersecting courses, Learning Taxonomy, IS BOK

1. INTRODUCTION

To identify solutions to the current credibility crisis in the IS discipline, Firth et al. (2011) developed six propositions. One of the most poignant of the six being that "the credibility of the IS discipline lies in the design and delivery of excellent courses and curriculum." According to Dick et al. (2007), declining student enrolment contributes significantly to the current crisis that

IS departments face. The IS2010 Model (Topi et al., 2010) is the latest set of curriculum guidelines that educational institutions can use in designing their IS Programs. It may not, however, get us through the crisis completely without the other complementing initiatives to address the issue in a holistic fashion. The IS2010 Model acknowledges the broader scope of the IS discipline by allowing the curricula to go beyond the schools of business and management to attract more IS students interested in different application domains. In this case study, we discuss two IS tracks designed with the IS2010 curriculum guidelines in mind: Business and Graphics Communications. The new CIS program, to be implemented in Fall 2012, also has two other tracks (IT Services and System Development) which are not discussed in this paper.

In Software Engineering, contributors describe the body of knowledge by indicating the levels of understanding using Bloom's taxonomy (Bourque & Dupuis, 2001). We use a two-dimensional cognitive model adaptation of Bloom's Model (Anderson & Krathwohl, 2001) for mapping the knowledge levels of CIS business track courses.

In section 2, we briefly review the structure and characteristics of the IS2010 Model as well as Anderson and Krathwohl's Cognitive Taxonomy. In section 3, we summarize the process of our CIS program redesign. We then discuss the local factors that influenced our CIS program redesign in section 4. In section 5, we discuss the structure of the new CIS program with four tracks and the design details for two IS tracks. In section 6, we verify how these courses meet the IS2010 curriculum guidelines and map the CIS business track courses onto the knowledge elements recommended in the IS2010 model. We also apply Anderson and Krathwohl's Cognitive Taxonomy to those courses for determining appropriate assessments. In the conclusion section, we emphasize the opportunity that exists in enhancing the CIS programs with newer tracks in different application domains.

2. LITERATURE REVIEW

Among the five disciplines under computing (Computer Science (CS), Computer Engineering (CE), Information Systems (IS), Information Technology (IT) and Software Engineering (SE)), the IS discipline is most concerned with organizations (JTFCC, 2005) and application systems in various domains that enable the organization to function, succeed, and comply with legal and regulatory requirements (Agresti, 2011). The crux of the IS discipline is in the value provided by the application of the technology rather than the technical components. With the variety, number, and demand for strategic application of domain-centric applications rapidly increasing, declining enrollments in IS programs and related computing disciplines is of serious concern (Dick et al. 2007). To improve enrollments, Firth et al. (2008) suggested revising the focus of those courses, early in the IS program, to focus more on IS than on CS or IT. Some institutions have already redesigned their IS curricula (e.g., Koch, Van Slyke, Watson, Wells, & Wilson, 2010; McGann, Frost, Matta & Huang, 2007) to address the recruitment problems. In this context, we recognize the value of the new IS2010 model curriculum in addressing enrollment issues through application beyond business domains.

IS2010 Curriculum Recommendations

Based upon periodic reviews, the IS Curriculum Task Force came up with the current IS2010 model curriculum (Topi et al., 2010) that is flexible, domain-independent and well structured. The IS2010 model curriculum cuts across the usual departmental silos by allowing the inclusion of any application domain (i.e., going beyond schools of management and business).

IS2010 specifies a set of structured outcome expectations starting with high-level IS capabilities which are translated into three categories of knowledge and skills: foundational, IS- specific and domain fundamentals. With just seven core courses addressing the high-level IS capabilities, this model offers flexibility for designing IS programs with several tracks emphasizing various application domains. It provides catalog descriptions and learning objectives for the core and elective courses as well as a mapping for the depth of knowledge metrics for these courses along with the IS Body of Knowledge.

Cognitive Taxonomy

In 1956, Benjamin Bloom published a learning taxonomy consisting of cognitive (mental), affective (emotions/ feelings), and psychomotor (physical skills) domains focusing on the

cognitive domain (Bloom, 1956). Anderson and Krathwohl (2001) revisited Bloom's taxonomy with intentions of updating, revising and "…refocusing education's attention on the value of the original Handbook…" and assisting educators "…as they struggle with problems associated with the design and implementation of accountability programs, standards-based curriculums and authentic assessments (p. XX1)." (See Figure 1.)

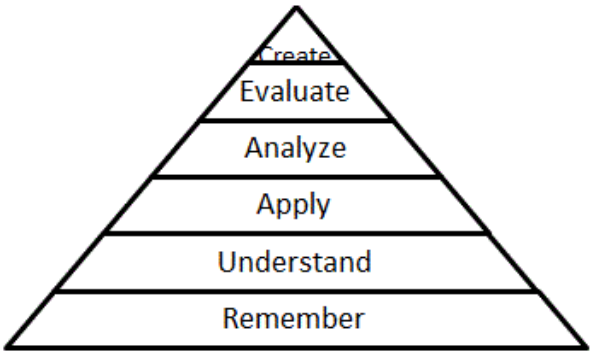


Figure 1 - Anderson & Krathwohl's Cognitive Model

Anderson and Krathwohl noted that, "The revision emphasizes the use of the Taxonomy in planning curriculum, instruction, assessment, and the alignment of these three (2001, p 305). Thus, this model is well structured to use as a guideline for evaluating ABET accreditation standards. The revision represents a significant shift from Bloom Taxonomy's primary focus on assessment to the teaching process where faculty can use the model to classify and identify project objectives.

Similar to Bloom's Model, students' levels of learning progress from a state of memorization of facts, to eventual application of concepts in a distinct functional domain. However, unlike Bloom's single dimension taxonomy, Anderson and Krathwohl's (2001) framework is represented by a two dimensional table consisting of carefully defined categories of knowledge and cognitive processes. The "Knowledge" dimension is divided into four categories: factual knowledge, conceptual knowledge, procedural knowledge, and metacognitive knowledge. The "Cognitive Process" dimension provides a means of assessing the retention and transfer of knowledge and is described through six categories of processes illustrated in Figure 1. The "Retention" dimension is most closely aligned with the basic "Remember" level of the model. The "Transfer" of knowledge gains progressively more depth as the tasks involved move from "Understand" to "Create." The following breakdown provides a brief description of each of the categories of knowledge and their associated cognitive process dimension in parentheses.

Factual knowledge (Remember level) is the basic form of knowledge described whereby a student becomes familiar with a discipline and its technical vocabulary. The associated cognitive processes focus upon the retention of concepts through recognizing and recalling relevant knowledge from long term memory.

Elements of the next three knowledge transfer categories can be found in differing degrees throughout the remaining levels of the cognitive taxonomy (Figure 1). The process classification, (in parentheses) is determined by the task being applied.

Conceptual knowledge (Knowledge transfer) describes a systems-type concept in looking at the "interrelationships among the basic elements."

Procedural knowledge (Knowledge transfer) focuses upon appropriately applying knowledge to solve a subject matter specific issue.

Metacognitive knowledge (Knowledge transfer) is essentially an awareness of what one knows: strategic knowledge, self-knowledge, and knowledge of the cognitive demands for a task.

When designing our IS tracks, we considered the different levels of Anderson and Krathwohl's (2001) Cognitive Model. Next we explain our redesign process. Later in section 5, we apply Anderson and Krathwohl's two-dimensional model for courses in one of the CIS tracks.

3. REDESIGN PROCESS

The intention of our CIS redesign was to select application domains having viable minors and intersecting courses. The first step required assigning a coordinator for managing the team effort and delivering the end product. The CIS team utilized an iterative process for the program redesign which involved:

- Setting basic criteria for the program such as alignment with model curricula.

- Identifying CIS program outcomes that reflected the department's program educational objectives.
- Setting limits on the number of new courses with consideration made for teaching load constraints.
- Identifying viable tracks having courses intersecting with computing by contacting program coordinators in various departments in the University.
- Structuring the program architecture to serve as a baseline design.
- Utilizing existing courses, and involving faculty members for developing new (or redesigning) courses in their respective areas of expertise.
- Liaising with all stakeholders such as the Registrar's Office to ensure the program met all the university-wide requirements.
- Revising the architecture and designs based on internal and external reviews and feedback.

The whole exercise took over a semester. The coordinator was given some release time to manage this redesigning exercise. The IS2010 gave a framework for structuring the program. The existing program outcomes were modified to suit tracks other than business and to allow for possible future ABET accreditation.

The design considered the following constraints: keep the number of newly developed courses to a minimum; make use of existing courses; and identify courses that could be shared among the IS Core courses, foundational and university required courses and domain fundamental courses. As a result, seven new courses were created. One section will be offered for each course per year and added to the teaching load. Intersecting courses, linking IS with application domains, were also identified.

In the next section, we discuss the rationale for selecting the tracks. The local context played a major role in limiting the number of tracks to four.

4. CIS REDESIGN: IDENTIFYING TRACKS

Our CS department, located in the College of Science, offers two programs, CS and CIS. The present CIS program shares several courses with the CS program. The program's intent has been to provide a generalized curriculum in the applied aspects of computing or informatics (Duben et al., 2006). Although the CIS program addressed the domain fundamentals of IS2010 (by requiring a minor or another major), it lacked *intersecting course*s applying the concepts to specific domains. In view of our course load constraints, the domains, already having such intersecting courses, are good candidates for CIS tracks. Next we explain how the local context played a role in choosing the tracks for CIS redesign.

Application Domains

Because computers are used in every discipline, we can have, theoretically, a CIS track for every field of study. The consideration of application domains (as tracks within CIS) will vary from institution to institution, depending on the programs offered and the availability of intersecting courses. Initially, we considered the following academic domains as program tracks: Business, Multimedia /Graphic Arts, Healthcare, Education, Law/Security, and Science.

Local Context

The redesigned CIS program was intended for students wishing to study either the application of computers in a chosen domain or an area of specialization within the computing discipline. At our institution, a track in Business Administration helps fill the gap created by the Fall 2011 termination of the MIS program housed in the AACSB-accredited Business School. Digital Art and Graphic Communications have several intersecting courses, thus also providing a strong option. With their overlapping courses with the CS core curriculum, Science and Mathematics are also natural candidates for domain specific tracks. However, since our CS program requires 12-credit hours of science courses and additional mathematics courses, students with an aptitude in Mathematics and Science may most likely consider majoring in CS rather than CIS. Further, we wish to avoid CIS competing with CS for enrollment. We are also considering developing tracks in Healthcare, Education, and Law/Security. Since many application domains require new intersecting courses, only four tracks, which offer the greatest potential to attract students, will be initially offered.

CIS Tracks

Students choosing the CIS program will be encouraged to choose a track pertaining to an

application domain or an area of specialization within computing. The authors' university does not offer separate programs in IT and SE so two specialization tracks in these two computing disciplines were developed:

1. Business (includes a minor in Business Administration)
2. Multimedia (includes a minor in Graphic Communications Technology)
3. IT Services (includes a minor in Computer Networking)
4. Software Development (some other minor or specialization in a computing area such as web or game development)

All four tracks have a common core of IS courses discussed in the next section. Each of the first three tracks has a specific minor. In the fourth track, a student chooses a minor other than those three or additional courses relating to system development. To illustrate the broader scope of the IS2010 Model, we confine our discussions to the Business track and its modified application to the Multimedia track. To begin, we discuss the architecture of the CIS program.

5. REVISED CIS PROGRAM

The total credit-hour requirements for our CIS program stands at 124 (41 at three credit hours and 1 at one credit hours) as distributed in Table 1.

CIS Architecture

Our institution requires every undergraduate program to include 17 general education (University Study) courses. Since the CIS core utilizes two of these courses, the program requires 45 credit-hours of courses toward foundational knowledge and skills. Each of these courses addresses some of the generic student learning outcomes including:

- Demonstrate capabilities for critical thinking, reasoning, and analyzing.
- Demonstrate effective communication skills.
- Demonstrate the ability to integrate the breadth and diversity of knowledge and experience.
- Demonstrate the ability to make informed, intelligent value decisions.

These general education courses represent the foundational courses referenced in the IS2010 model. The redesigned CIS program also has courses in domain fundamentals and courses that intersect the domains and computing. In the following, we provide the details of the CIS program architecture.

Table 1: CIS Architecture

Category	Credit hours
University Studies	**45**
CIS Major	**55–58**
Core	40
Supplemental	15-18
Additional Requirements	**21 – 24**
Mathematics	6
Minor or advised courses	15-18
Total	**124**

CIS Core

In a recent study examining the alignment of current IS programs with the IS2010 model, Apigian and Gambill (2010) reported that only four of the seven IS2010 core courses (Fundamentals of Information Systems, Data and Information Management, IT Infrastructure, and Systems Analysis and Design) are in 80% (or more) of the current IS programs. Our CIS program includes these courses as well as a capstone project course that helps entwine the learning experiences of these courses, and others, as the students prepare to enter the workforce.

We split the IS courses into two groups: common *Core* and track-specific *Supplemental*. The additional requirements include track specific courses, which could be for a minor in the chosen domain.

In Table 2, we list the 14 CIS core courses and map 11 of them onto the IS2010 Model. The numbers under the IS2010 Model column correspond to the order in which the core and the sample electives are listed on page 35 of the IS2010 Model document (Topi et al., 2010). Each is a three credit hour course except for, CS495, a one credit hour senior seminar. The Discrete Structure course is included for addressing one of the knowledge areas in computing. The Senior Seminar course focuses upon social and ethical issues in computing.

The Capstone Experience is a project course that consolidates the various knowledge and skills learned in other courses. A zero-credit hour (IS003) Information System Assessment is also required but not listed in Table 2.

CIS Supplement to Core

The (five or six) CIS supplement courses differ according to track. Track specific courses prepare students to meet the technology needs of problems in that discipline (ABET, 2011). For application domain tracks, such as Business and Multimedia, intersecting courses are included under the CIS supplement. These are discussed later for the Business and Multimedia tracks. For the IT Services and System Development tracks, additional relevant CS courses are included.

Table 2: Mapping of CIS Core Courses on IS2010 Model

Course	Name	IS2010 Model
IS175	Computer Information Systems – I	Core - 1
IS275	Computer Information Systems – II	Core- 3 & Elective- 3
IS340	Information Technology	Core- 5
IS375	Database and Information Systems	Core- 2
IS445	Systems Analysis & Design	Core- 6
IS448	IS/IT Project Management	Core- 4
IS575	IS/IT Strategy and Management	Core-7 & Elective- 6
IS130	Application Development – I	Elective- 1
IS245	Web Development and Security	Electives– 1 & 6
IS320	Human Computer Interaction	Elective- 4
IS330	Application Development - II	Elective- 1
CS245	Discrete Structure	
CS495	Senior Seminar	
UI450	Capstone Experience	

Business Track

The Business track prepares students planning for a career involving application of computers in all areas of business administration. This track replaces MIS (no longer available at our institution) with greater technical content. The supplemental courses are listed in Table-3.

Table 3: Supplements - Business Track

Course	Name
AC330	Accounting Information Systems
IS360	Mobile Application development
IS440	Web Design for Electronic Commerce
IS465	Management Support Systems
MK555	Internet Marketing

Multimedia Track

The Multimedia Computing track is for developing skills required for implementing multimedia designs using computers. The supplemental requirements include courses from Art as well Computer Science as shown in Table 4.

Table 4: Supplements - Multimedia Track

Course	Name
AR104	Design Foundations
AR323	Art & New Technology
IS360	Mobile Application development
IS440	Web Design for Electronic Commerce
IS465	Management Support Systems

IT Services

The IT Services track is centered on a minor in Computer Networking. This track was developed for students considering a career in IT services, such as infrastructure development or support. The supplement requires CS courses in programming, operating systems, and data communications.

System Development

The system development track has built-in flexibility to cater to changing demands in the field. It is oriented toward students who are interested in applying computers in a domain outside those offered through the other CIS program tracks or in a specialized computing area such as web computing or game development. The supplement requires courses in programming, operating systems, and mobile applications development. Students will take 15

hours of domain related or CS/IS courses, as advised by faculty, towards their goal (e.g., web computing, game development).

6. DETAILED CASE EXAMPLE- BUSINESS TRACK

In this section, we provide comprehensive details for the Business track in view of its historical significance. In the next section, we also provide design details pertaining to the Multimedia track – an alternative to the Business track.

Design

As indicated in Table 1, the Business CIS track requires 124 credit-hours of study (41 three credit hour courses plus the one credit-hour CS495). This includes the 15 foundational (University Studies) courses, 14 CIS core courses (excluding the zero-credit assessment course), five supplemental courses, two Mathematics courses, and five courses towards a minor in Business (see Table 3).

Three high-level IS capabilities are described in in the IS2010 curriculum model (Topi et al., 2010, pp 16) IS Specific, Foundational, and Domain Fundamentals. It is possible that some of the courses intersect more than one capability sector. Also, in order to stay within the overall credit-hour requirements, courses are designed in such a way that they are shared among the Foundational and Core requirements.

Table 5: Business Minor (Business Track)

Course	Name
AC221	Principles of Accounting I
AC222	Principles of Accounting II
EC225	Principles of Macroeconomics
FI361	Financial Management
MG301	Principles of Management
MK301	Principles of Marketing

Figure 2 shows the course mapping by capability sectors for the CIS-Business Track. Excluding the IS003 (zero-credit hour), all 41 courses (indicated in Figure 2) are required to complete the CIS-Business track major. Courses located within the "Foundational and University Requirements" circle address the requirements of the program as well as graduation requirements for the University. Within the Business Track circle, we indicate the courses required to obtain a minor in Business and the courses that intersect the IS and Business domains.

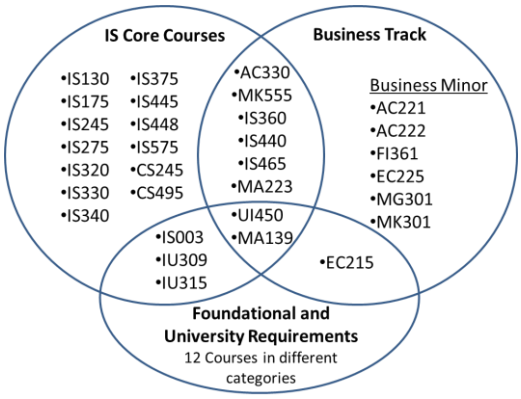


Figure 2: Courses for CIS Business Track

Figure 2 also shows the courses that intersect IS and the application domain which were not in the earlier CIS program. For reasons of credit-hour efficiency, two of the Foundational courses (IU 309 – Technical Writing and IU315 - Cyber Ethics) are shared with the IS Core. In addition, the Foundational course (EC215 Macroeconomics) is shared with the Business Domain while the capstone experience and Applied Calculus (MA139) intersect all three.

Verification

Our redesign process reformulated the program outcomes, while retaining the overall format of other programs in the department, to reflect the applied nature of CIS in a variety of domains. Verification of these outcomes depends on the learning outcomes of the courses. The courses in the redesigned CIS match, as shown in Table 2, the core and some of the elective courses of the IS2010 Model. Course descriptions in the IS2010 model guided the design of the courses in our CIS core as well as some electives.

While this is only a high-level verification, it sets the direction in describing the actual courses meeting the knowledge elements stated under the IS Body of Knowledge: General computing, IS specific, Foundational, and Domain-related (Topi et al., 2010 Appendix-4 pp 81-84). Table 6 maps the courses in the Business Track (Figure 2) having the potential to address the various knowledge elements in the above four knowledge areas. Similar mappings could be

administered for other tracks to assist in refining course descriptions.

Table 6: Knowledge Area Mapping

General Computing Knowledge Areas	
Programming Fundamentals	IS130, IS330, IS360
Algorithms & Complexity	CS245, MA139, MA223
Architecture & Organization	IS340
Operating Systems	IS340
Net Centric Computing	IS340, IS440, IS339
Programming Languages	IS130, IS330, IS360
Graphics & Visual Computing	IS130, IS330
Intelligent Systems	IS 465
Information Systems Specific Knowledge Areas	
IS Management & Leadership	IS275, IS575, IS440
Data & Information Management	IS575, IS375, IS465
Systems Analysis & Design	IS445
IS Project Management	IS448
Enterprise Architecture	IS275, IS175
User Experience	IS320
Professional Issues in Information Systems	IS439
Foundational Knowledge Areas	
Leadership & Communication	IU309, Literary & Oral Expression Categories
Individual & Organizational Knowledge Work Capabilities	UI450, CS495, IU315
Domain-related Knowledge Areas	
General models of the domain	AC221, MG301, MK301, FI361, EC215
Key specialization within the domain	AC222, AC330, MK555, EC225
Evaluation of performance with the domain	IS003

Depth of Knowledge Metrics

In addition to the knowledge areas, the depth of knowledge achieved through the various courses using appropriate assessments must also be addressed. Throughout the educational process, students are expected to progress through their courses of study, from that of acquiring factual knowledge and skills, to ultimately applying those resources to a given situation.

Several learning models have been designed to help faculty evaluate and design courses that will aid in assessment and progression. Such models are beneficial for evaluating courses in light of college, program and accreditation considerations. Bloom's Taxonomy was used in the development of the IS2010 model in addressing knowledge metrics (Topi et al., 2010 pp. 78-80). Anderson and Krathwohl furthered Bloom's model to not only assist with assessment, but to also help in the identification and classification of project objectives. In the next section, we apply Anderson and Krathwohl's model to the Business Track courses (Figure 2) from both learning and assessment perspectives. (See Appendix.)

Application of a Cognitive Taxonomy to Business Domain Courses

As described in Section 2 and for assessment purposes, Anderson and Krathwohl (2001) suggest that courses falling into the "Remember" Cognitive Process Dimension could be assessed through prompt-based recognition tools. Assessments for the "Knowledge Transfer" levels from "Understanding" through "Create" require the students to progressively apply their knowledge to new situations. At the "Apply" level, the assessments could require students to determine and apply the necessary procedure to solve a problem or situation. Assessments used at the "Analyze" level could require students to distinguish relevant from irrelevant facts before finding a solution. For the "Evaluate" level, students could be asked to make judgments based upon criteria and standards. Using these definitions, we examine the business track courses and then discuss the assessment options.

The junior level Accounting Information System course (AC330) focuses upon domain-specific fundamentals addressing data security and transaction cycle concepts. The course focuses upon the first three dimensions of Anderson and Krathwohl's learning taxonomy as students gain factual knowledge about the field, learn new applications, and then analyze and apply their

knowledge to projects within the course. (See Appendix.)

Mobile Application Development (IS360), currently under development, will correspond with the first three dimensions of the learning taxonomy. Students will first gain factual knowledge about designing and coding applications for mobile resources and then apply their knowledge throughout the course in the development of small mobile apps.

The Web Design for Electronic Commerce course (IS440) covers all of the dimensions of the Anderson and Krathwohl's taxonomy. Students learn techniques, languages, and tools for building Web pages and finally analyze a client's Web site needs and design and create a site to fulfill those needs.

The Management Support Systems course (IS465) focuses on the last three dimensions of the learning taxonomy. Students gain factual knowledge regarding system design and design tools, however, the focus of the course is in evaluating a business process and creating the design models associated with developing a system for that process.

The Internet Marketing course (MK555) introduces students to the strategic application of Internet technologies to a business' marketing plan. Students examine the characteristics and behaviors of Internet shoppers and the effect that web content has upon their buying behaviors. The course focuses on the first three dimensions of the Anderson and Krathwohl's learning taxonomy model as students gain factual marketing knowledge and then apply their knowledge through the analysis of Internet content and resources.

The capstone course (UI450) is taken by students in all CS/CIS tracks. In this experiential learning course, students apply their accumulated knowledge and skills as they work for a client to analyze, design and develop an IT solution for the client's specific need. The focus of this course is on the last dimensions of the learning model.

The Applied Calculus (MA139) and Elementary Probability and Statistics (MA223), provide a broad mathematical foundation applicable to multiple majors. Due to the general nature of these courses, they are not included in the analysis.

Each of the business domain courses, especially Internet Marketing, contains an element of gaining and remembering factual knowledge. Most of the courses conclude with the students applying their knowledge through a project-type assessment. This is especially true for the capstone experience course where students design and develop a project for an external client. Thus, in assessing students' levels of learning throughout the program, it appears that the assessment instruments should progress from that of fact-based definitional tools to those of development, evaluation, and application.

6. MULTIMEDIA TRACK: AN ALTERNATE DOMAIN

Since our objective is to consider domains beyond Business, we applied the same process for designing a CIS track to Multimedia. Students are required to minor in Graphics Communication Technology (See Table 6). In addition, five other courses are included in this track (see Table 4).

Table 6: Graphics Communication Technology Minor

Course	Name
GM180	Intro. To Industrial Graphics
GM200	Vector & Bitmapped Graphics
GM282	Vector and Text Graphics
GM380	3D Modeling and Animation
GM386	Interactive Multimedia & Animation
GM480	3D Animation Pipeline

Here, students take two Art courses for developing artistic design skills. The core CIS courses provide the necessary computing concepts that help prepare students for lifelong learning (as new technologies emerge) (Walker, 2010).

The courses shared between the three High-level IS capabilities are shown in Figure 3. Excluding the IS003 (zero-credit hour), all 41 courses (indicated in Figure 3) are required to complete the CIS-Multimedia track major. The Art & New Technology course is an intersecting course with Design Foundation as a prerequisite. For reasons of course load efficiency, Photography Fundamentals – PG284 is a shared Foundational course.

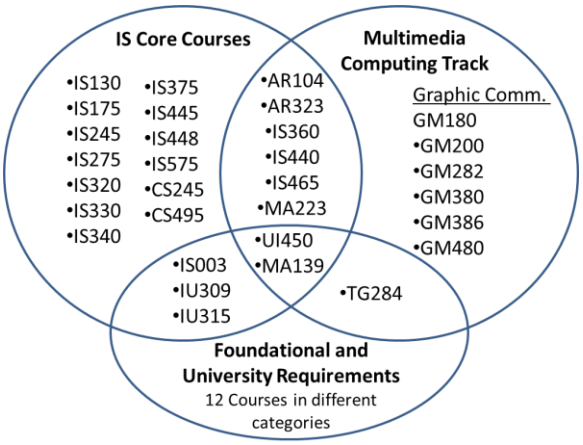


Figure 3: Courses for CIS Multimedia Track

The verification for this track will be similar to the one shown in Table -6, except for domain related knowledge areas. Here, the courses with GM and AR prefixes will map for general models of the domain and key specification areas within the domain. We can also apply the Anderson and Krathwohl's cognitive taxonomy to Multimedia domain courses.

7. CONCLUSION

The IS2010 model provides opportunity for designing a flexible IS program reaching out to all possible application domains. Walker (2010) noted that, although local resources, limitations, and priorities will influence programmatic elements, essentially all CS programs attempt to build problem-solving skills, from vision to implementation, to assist in IT solution development for people in diverse fields. Thus, institutions will adapt the IS2010 model to suit their local context.

In this case study, the authors describe a flexible CIS program at their institution that has been approved to start in Fall 2012. This program utilizes the IS2010 model to reach out to multiple knowledge domains. Four tracks were chosen to suit their local conditions. The redesign's architecture offers considerable flexibility in terms of adding new tracks. This is achieved through (1) having a supplemental component to the IS core that allows inclusion of appropriate intersecting courses to bridge computing with application domains and (2) requiring a minor in the application domain or having a mechanism for faculty to advise a set of relevant courses in an area of specialization.

The program's core courses were examined through the frameworks of the IS2010 model and Anderson and Krathwohl's. cognitive model. These mappings aid in choosing the appropriate topics for, and designing appropriate assessments in, program courses. The presented design process and the concept of tracks in application domains serves as a case study that is based on the IS 2010 Model. In addition, the mapping techniques, using a cognitive model, can be applied to courses in various tracks for matching course objectives with appropriate assessment techniques with consideration made for ABET accreditation.

Our general process can be replicated by other universities. We utilized a case study approach, explained in Section 3, for our redesign initiative. We addressed the important higher level issues -such as program objectives, accreditation intentions - at the very beginning. Creating a baseline program architecture that is agreed upon by all department members is crucial. Involving all of the faculty members and consulting all of the stakeholders (including the Registrar) helps in speeding program approval. Another key step is identifying domains that have intersecting courses with computing. If there are no constraints, it is possible to develop intersecting courses jointly with domain-specific departments. Context will dictate the choice of tracks.

One of the recent CIS revisions (Pauli et al., 2010) has five categories of specializations (Software Development, Web Development, Business Analysis, Infrastructure Analysis and Change Management). It is encouraging to note that they have realized growth in enrollment through their CIS revision. We expect similar results as three of their specializations are considered in our CIS redesign. Such aims to address, in part, the crisis through which the IS discipline is currently undergoing. Extending IS beyond the Business domain through additional IS minors should attract more students from other majors. However, the results of these program modifications are yet to be realized at the authors' institution as the foundation for change is being set into place.

8. ACKNOWLEDGEMENT

An exercise of this nature is not possible without the support of all the faculty members in the department of Computer Science at Southeast

Missouri State University. The program design presented here is a departmental effort.

9. REFERENCES

ABET(2009). http://www.abet.org/Linked%20Documents UPDATE/Criteria%20and%20PP/C001%2010-11%20CAC%20Criteria%2011-16-09.pdf Accessed on April 28, 6 & 13, 2011.)

ABET (2011). *Criteria for Accrediting Computing Programs* http://www.abet.org/uploadedFiles/Accreditation/Accreditation_Process/Accreditation_Documents/Current/cac-criteria-2012-2013.pdf (Accessed on January 3, 2012.)

Agresti, W. W. (2011). Toward an IT Agenda. *Communications of the Association for Information Systems, 28*(17), 255-276.

Anderson, L. W. & Krathwohl, D. R. (Eds.) (2001). *A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives*. Allyn & Bacon. Boston, MA: Pearson Education Group.

Apigian C. H., Gambill, S. E., (2010). Are We Teaching the IS2009* Model Curriculum?, *Journal of Information Systems Education*, *21*(4), 411-420.

Bloom, B.S. (Ed.) (1956). *Taxonomy of Educational Objectives, the classification of educational goals - Handbook I: Cognitive Domain* New York: McKay.

Bourque, P., and Dupuis, R. (Eds) (2001). *Guide to the Software Engineering Body of Knowledge.* IEEE CS Press, Los Alamitos, CA.

Dick, G., Granger, M., Jacobson, C., & van Slyke, C. (2007). Where Have All the Students Gone? Strategies for Tackling Falling Enrollments *AMCIS 2007 Proceedings.* Paper 334.

Duben, A. J, Naugler, D. R., & Surendran K. (2006). Agile Computing Curricula. *Information Systems Education Journal*, *4*(53), http://isedj.org/4/53/ ISSN: 1545-679X.

Firth, D., Lawrence, C., & Looney, C. A. (2008). Addressing the IS Enrollment Crisis: A 12-step Program to Bring about Change through the Introductory IS Course. *Communications of the Association for Information Systems, 23*(2), 17-36.

Firth, D., King, J., Koch, H., Looney, C. A., Pavlou, P., & Trauth, E. M. (2011). Addressing the Credibility Crisis in IS. *Communications of the Association for Information Systems, 28*(13), 199-212.

JTFCC (2005) —The Overview Report‖, Joint Task Force for Computing Curricula, Sept., Association for Computing Machinery, New York, NY.

Koch, H., Van Slyke, C., Watson, R., Wells, J.; and Wilson, R. (2010). Best Practices for Increasing IS Enrollment: A Program Perspective. *Communications of the Association for Information Systems, 26*(22). 477-492.

McGann, S. T., Frost, R. D., Matta, V., & Huang, W. (2007). Meeting the Challenge of IS Curriculum Modernization: A Guide to Overhaul, Integration, and Continuous Improvement. *Journal of Information Systems Education*, *18*(1), 49-62.

Pauli, W. E., Halverson, T., McKeown, J., (2010). The 2010 CIS Baccalaureate Degree Compared with IS2010 Guidelines. *ISECON 2010 Proceedings*, Paper 1396.

Topi. H., Valacich, J. S., Wright, R. T., Kaiser, K., Nunamaker, J. F., Sipior, J. C., & de Vreede, G. J. (2010). IS 2010: Curriculum Guidelines for Undergraduate Degree Programs in Information Systems. *Communications of the Association of Information Systems*, *26*(18), 360-429.

Walker, H. M. (2010). Eight Principles of an Undergraduate Curriculum. *ACM Inroads*. *1*(1), 18-20.

Appendix: Anderson and Krathwohl's Taxonomy Table

Course #	Course Name	The Cognitive Process Dimension				
		Remember	Apply	Analyze	Evaluate	Create
AC330	Accounting Information Systems	X	X	X		
IS360	Mobile Application Development	X	X	X		
IS440	Web Design for Electronic Commerce	X	X	X	X	X
IS465	Management Support Systems			X	X	X
MK555	Internet Marketing	X	X	X		
UI450	Capstone Experience			X	X	X

Problem Solving Frameworks for Mathematics and Software Development

Kirby McMaster
kmcmaster@weber.edu
CSIS Dept, Fort Lewis College
Durango, CO 81301, USA

Samuel Sambasivam
ssambasivam@apu.edu
CS Dept, Azusa Pacific University
Azusa, CA 91702, USA

Ashley Blake
ablaketx@hotmail.com
Scribblin' Sisters
Houston, TX 77094, USA

Abstract

In this research, we examine how problem solving frameworks differ between Mathematics and Software Development. Our methodology is based on the assumption that the words used frequently in a book indicate the mental framework of the author. We compared word frequencies in a sample of 139 books that discuss problem solving. The books were grouped into three categories: Traditional Math, Applied Math, and Software Development. We obtained a list of the most frequent words in each category, and used these lists to describe three problem solving frameworks. Applied Math uses models and algorithms to solve problems. Traditional Math is more concerned with proving theorems. In the Software Development framework, customers provide the problem, and models and algorithms are used to create a software solution. Our findings have relevance in the development of approaches for teaching problem solving in Mathematics and Software Development courses.

Keywords: problem, solution, framework, model, algorithm, mathematics, software.

1. INTRODUCTION

A monkey and a banana are placed in a room. The monkey desires the banana, but the banana is high overhead. The room also contains a box. If the monkey moves the box and climbs on it, the banana can be reached. This is one version of a classic *problem solving* situation in Artificial Intelligence (Bratko, 2001).

Scientific activities that demonstrate problem solving have been performed for centuries. Archimedes was able to determine if a king's crown was solid gold. Newton developed Calculus, anecdotally to explain why an apple fell on his head. Attributes of problem solving have been studied in fields such as Psychology, Medicine, Warfare, Management, and Engineering.

One issue often mentioned is whether problem solving can be expressed in terms of a single set of general principles, or if unique processes are required in different knowledge domains. In this paper, our primary focus is on problem solving in Mathematics (Math) and Software Development (SD). Does a single framework for problem solving apply to Math and SD, or do these academic disciplines solve problems in different ways?

Problem Solving in Math

Mathematics encompasses a large number of subject matter areas, such as Algebra, Calculus, Geometry, Differential Equations, and Number Theory. Within these areas, there are different levels of emphasis on problem solving and theorem proving. In his classic book *How to Solve It*, Polya (1945) promotes methods of solving problems in Math:

> Studying the methods of solving problems, we perceive another face of mathematics. Yes, mathematics has two faces; it is the rigorous science of Euclid, but it is also something else. Mathematics presented in the Euclidean way appears as a systematic, deductive science; but mathematics in the making appears as an experimental, inductive science. Both aspects are as old as the science of mathematics itself.

Almost fifty years later, Velleman (1994) wrote a book called *How to Prove It*, in which he discusses the same two faces of mathematics, but with a preference for constructing proofs:

> This textbook will prepare students to make the transition from solving problems to proving theorems by teaching them the techniques needed to read and write proofs.

The priority in each Math field can be on solving Math problems, or it can be on using Math to solve *real world* problems. Polya's book and Velleman's book spend most of their coverage on solving (or proving) Math problems. Several current Math books on problem solving provide students with techniques to help them compete in Math exams, such as the Mathematical Olympiads (Zeitz, 2006; Andreescu & Gelca, 2008). These books focus almost entirely on solving Math problems, not real world problems.

On the other hand, the recent book entitled *How to Solve It: Modern Heuristics* by Michalewicz and Fogel (2004) leans toward the use of Math to solve real world problems. Books on Statistics and Operations Research are often obligated to deal with real world problems. This is especially true for Applied Statistics, with its attention to the collection and analysis of real world data.

Given the diversity of content and form within Math, it seems reasonable to expect that more than one mathematical framework could be applicable to problem solving. A framework for solving problems is not equivalent to a framework for proving theorems. Also, a framework for solving Math problems might differ from a framework for solving real world problems.

Problem Solving in Software Development

Software Development has rapidly evolved into an extensive discipline that attempts to solve a variety of computation and communication problems. Coursework areas include Programming, Operating Systems, Databases, Networks, Software Engineering, and Electronic Commerce.

The earliest use of computers to perform repetitive calculations could be considered a form of problem solving. The field of Artificial Intelligence has specifically targeted problem solving in software. An early example is the General Problem Solver program for proving theorems, developed by Newell, Shaw, and Simon (1959). Current versions of Microsoft Excel have a Solver *add-in* that can search for solutions to a wide range of numerical problems.

Over a decade ago, IBM developed the Deep Blue computer system to play chess, and reached the Grand Master level. Recently, IBM's Watson computer competed on the TV game show *Jeopardy* and defeated two human champions.

Some areas of computing are more explicit about their desire to solve problems, especially topics which are heavily dependent on Math. Software Development areas such as Programming, Database, and Software Engineering solve problems with less reliance on Math. Most SD students prefer to be exposed to as little Math as possible in their courses.

We do not expect to find a single problem solving framework that is appropriate for all of Software Development. SD areas may share

some features of the Math frameworks, but each SD field contains domain-specific concepts that are difficult to combine into a common framework.

Plan of this Research

In this paper, we examine how problem solving frameworks differ between Math and Software Development. Measurement of mental concepts is always difficult. Our methodology is based on the assumption that the words people use are suggestive of their mental state. In particular, we assume that words used frequently in a book indicate the mental state, or framework, of the author.

Certainly, a framework is more than a list of words. A framework must provide a way to combine the words into a unified "whole". However, we need the individual words to describe the relevant concepts that form the overall framework.

In this study, we compare word frequencies in a sample of Math and SD books that discuss problem solving. After organizing the books into subject matter categories, we list the most frequent words in each category. We then synthesize these results to propose a problem solving framework for each book category. Our findings have relevance in the development of approaches for teaching problem solving in Math and SD courses.

2. METHODOLOGY

The methodology used to examine problem solving frameworks is described in this section. The methodology involved the following steps:

1. Choose a broad sample of Mathematics and Software Development books.
2. Record frequencies for words used often in the books.
3. Convert nouns, verbs, adjectives, and adverbs to a consistent form.
4. Transform the word frequencies to make data from different books comparable.
5. Combine synonyms into *word groups*.
6. Determine the most frequent word groups in each category of books.

Sampling

By design, a wide variety of Math and SD books were sought for our sample. We needed books for which we could determine word usage frequencies. Because we did not have full text files, we selected books from the Amazon web site that included a *concordance* (a list of frequently used words). Our need for a concordance hindered our ability to obtain a random sample of books. However, Amazon does provide a concordance for many of its books, so we were able to get a diverse sample. The majority, but not all, of our sample books are suitable for use as college textbooks.

Books were chosen from three broad categories:

1. *Traditional Math* (TRM) includes books in fields such as Algebra, Analysis, Geometry, Number Theory, and Topology, along with some Probability and Statistics books. Books with the word *Theory* in the title were usually placed in this category. For example, the book entitled "Course in Probability Theory" was classified as Traditional Math.

2. *Applied Math* (APM) includes books with the words *Applied*, *Computational*, *Numerical,* or *Engineering* in the title. For example, the book with the title "Applied Engineering Mathematics" was classified as Applied Math. This category also contains Operations Research and Simulation books, along with Probability and Statistics books that are more applied than theoretical.

3. *Software Development* (SD) includes books on Object-Oriented Programming (OOP), Database (DB), and Software Engineering (SE). These books are used in core SD courses that teach students how to design and implement software systems.

Our complete sample consisted of 53 Traditional Math books, 59 Applied Math books, and 110 Software Development books. The SD sample contained 36 OOP books, 37 DB books, and 37 SE books. The total number of books in the sample was 222.

Data Collection

The Amazon concordance for a book provides a list of the 100 most frequently used words. These concordances screen out many (but not all) common English words, such as "the" and "of". For each concordance word, we recorded

the book code, word, and frequency. Frequency is the actual number of times the word occurs in the book.

Convert Words to a Consistent Form

One problem with using words to infer an author's framework is that words can take more than one form. For example, nouns and verbs may be singular or plural. Verbs can have various tenses. Adjectives and adverbs can have related syntax. To alleviate this problem, we converted many words to a consistent form. We did not want the relative frequency of a word to depend on the particular form an author favored. The following types of word conversions were performed:

1. Convert plural nouns to singular form ("elements" becomes "element").
2. Make verbs refer to plural subjects ("exists" becomes "exist").
3. Change verbs to present tense ("defined" becomes "define", "solving" becomes "solve").
4. Remove endings such as "al" and "ly" from some adjectives and adverbs ("computational" becomes "computation", "finitely" becomes "finite").

Transform Frequencies

Word frequencies were rescaled (or standardized) to allow comparisons between books of different lengths. We rescaled word frequencies within a concordance as follows:

1. We removed all words that are in the list of Top 100 Common English Words (Fry, 1993). Fortunately, Amazon had already removed most of these Top 100 words. Otherwise, we would have had few words left to analyze.
2. For the remaining (approx. 90) words, we calculated the average word frequency for the concordance.
3. We then restated each individual word frequency (Freq) relative to the average frequency (avgFreq) using the formula:

   StdFreq = (Freq / avgFreq) * 100

   With this calculation, a standard frequency (StdFreq) score of 100 represents the transformed frequency for the "average word" in the reduced concordance. A word with a StdFreq value of 300 would appear three times as often as the average concordance word in the same book.

Combine Synonyms into Word Groups

A special complication with assembling words into frameworks is that different words can have similar meanings. When relevant, we combined two or more synonyms into a concatenated *word group*. For example, *algorithm* and *method* became *algorithm/method*. We applied this step *after* standardizing the word frequencies (StdFreq) because we wanted the average frequency for a concordance to be based on individual words. When synonyms are combined into word groups, the StdFreq score for the group is the sum of the StdFreq scores of the words in the group.

3. PROBLEM SOLVING BOOKS

The primary approach in this study of problem solving frameworks in Mathematics and Software Development was to examine frequently used words in our sample of books. But which books in the sample discuss problem solving? Polya's "How to Solve It" is certainly a candidate. However, only four books in our sample contain the words *problem* and/or *solve* in the title. Instead, we chose to focus on books that include *problem* or *solution/solve* in their concordances. We assumed that these books would be more likely to involve problem solving, even though these words are often used in other contexts.

Table 1: Math and SD Books by Category

Category	All Books	*Problem* Books	*Problem+ Solution* Books
APM	59	53	48
TRM	53	26	13
SD (OOP) (DB) (SE)	110 (36) (37) (37)	60 (12) (17) (31)	8 (2) (0) (6)
Total	222	139	69

Starting with a sample of 222 Math and SD books, the number of concordances containing the word *problem* is 139. This initial constraint removes one-third of our sample. If we then eliminate books that do not include *solution* or *solve* in their concordances, the remaining

sample has only 69 books--about 1/3 of our original sample.

The main difficulty with limiting our analysis to these 69 "problem + solution" books is that the reduction does not apply equally to all book categories. Table 1 summarizes how the book counts are reduced in each category as we successively apply the *problem* and *solution* filters.

Requiring Applied Math concordances to contain the *solution* keyword in addition to *problem* is not an issue. The resulting sample has 48 of the 53 *problem* books. The drop is more precipitous for Traditional Math (from 26 to 13 books) and Software Development (from 60 to 8 books). Note that none of the Database books and only 2 of the Programming books include both keywords in their concordances.

Many authors of Traditional Math are more concerned with *proofs* than with problem solving. This can partially explain the reduced number of books in this category that contain *solution* or *solve*, but it doesn't explain the remarkably small number of Software Development books that mention *solutions*.

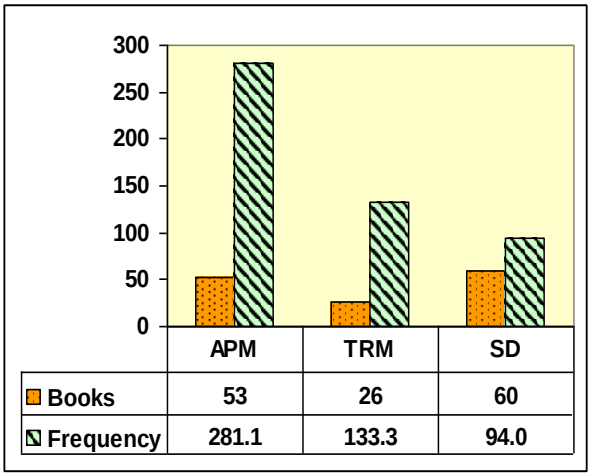
Problem solving is an important part of Software Development, as stated by McConnell (2004):

> Problem solving is the core activity in building computer software.

Programming, Database, and Software Engineering books use an alternative terminology for problem solving. From a Software Development perspective, *requirements* define the problem, and *software* is the solution. Programs and databases are essential components of the solution. The goal of Software Engineering is to effectively build software systems that meet customer requirements.

Because of the extreme sample size reduction that would result from requiring both *problem* and *solution* to be concordance words, we decided to impose the less restrictive constraint that only *problem* must be in the concordance. Figure 1 shows the resulting sample of 139 books used in the analysis that follows.

Across the three book categories, the average standard frequency (avgStdFreq) for the word *problem* varies widely. For the 53 Applied Math concordances that contain *problem*, the avgStdFreq of 281.1 indicates that this word occurs almost three times as often as an average concordance word. At the other extreme, in 60 Software Development books, *problem* occurs less often (94.0) than an average concordance word.



	APM	TRM	SD
Books	53	26	60
Frequency	281.1	133.3	94.0

Figure 1: "Problem" frequency by Category.

4. PROBLEM SOLVING FRAMEWORKS

This section describes how we obtained the words that form the problem solving frameworks for each category. We looked for words that are used *frequently* within each book and *consistently* across books in the same category. Given a category (e.g. Applied Math) and a word in at least one of the concordances, we calculated the number of books containing that word, plus the avgStdFreq for the word. We retained the words that appear in most of the category books and had a high avgStdFreq.

Our principal methodology decision was the choice of cutoff points for number of books and avgStdFreq. After some trial and error, we set the minimum number of books at 70% of the sample size. For the 53 APM books, 70% is 37 books (rounded). For avgStdFreq, we chose a cutoff point of approximately 150, with some judgment reserved for words near this point.

To qualify as a framework word, we wanted most of the books in the category to agree on the importance of the word. Some words had a high frequency, but appeared in only a few of the books. For example, the word *simulation* appears in 8 Applied Math books, with an

avgStdFreq of 219.7. This word is important in those 8 books, but is not used regularly throughout the Applied Math category.

Other words appear in most books, but with low frequencies. For example, the word *result* appears in 44 APM concordances, but the avgStdFreq value is a below-average 88.1. Of passing interest, this word could be considered a synonym for *solution*.

Applied Math Framework

Using the methodology explained in the preceding paragraphs, we generated a list of the 10 most frequent words and word groups for Applied Math books. This list is presented in Table 2. We include the word group *set/element*, even though its frequency value is slightly below 150.

Table 2: Most frequent words in Applied Math books (N=53).

Word Group	Books	Avg StdFreq
problem	53	281.1
algorithm/method	47	280.5
function	50	248.1
solution/solve	48	239.9
value/variable	52	250.9
model/modeling	38	227.7
equation/inequality	43	220.0
system	47	173.2
point/line	52	161.8
set/element	48	146.7

The most frequent word is *problem*. It is not surprising that this word is in all of the APM books, since this condition was used to generate the sample. What is unusual is that the frequency of *problem* and *solution/solve* is relatively high in this category. This suggests that the framework for Applied Math does emphasize problem solving.

The word groups *model/modeling* and *algorithm/method* describe this category's approach to solving real world problems. Models are used to abstract relevant aspects of the real world problem. Algorithms describe the computational effort needed to obtain a solution.

The Applied Math framework includes several widely-used mathematical objects--*function*, *variable*, *equation*, *point*, *line*, and *set*. These words appear frequently in most of the Applied Math books. However, other familiar Math concepts, such as *matrix*, *polynomial*, and *vector* do not appear in this general framework. These domain-specific words are in the concordances of some Applied Math books, but absent from many others.

Traditional Math Framework

Repeating the same methodology used for Applied Math, we obtained a list of the 12 most frequent words and word groups for Traditional Math books. The list is shown in Table 3. We include the word group *definition/define* in this list, since its avgStdFreq value is almost 150.

Table 3: Most frequent words in Traditional Math books (N=26).

Word Group	Books	Avg StdFreq
point/line	19	411.6
theorem/lemma/corollary	25	326.8
function	25	278.5
proof/prove	23	258.7
let	26	228.9
set/element	26	225.1
value/variable	20	197.3
show/shown	24	181.9
hence/thus/therefore	26	172.1
follow/following	24	163.3
equation/inequality	21	163.1
definition/define	23	149.3

The most frequent word group is *point/line*, which appears in 19 (73%) of the Traditional Math books. Points and lines--along with functions, sets, variables, and equations--are Math objects that are also in the Applied Math framework, but with different frequencies.

Two high-frequency word groups are *theorem/ lemma/corollary* and *proof/prove*. This reveals that the primary goal of Traditional Math is proving theorems. The words *model* and *algorithm* are not part of this framework.

Instead, this framework prefers the use of logic to solve Math problems.

Not everyone agrees that theorem proving is equivalent to solving problems. Concerning the "problem of proving things", Michalewicz and Fogel (2004) state:

> ... if you ask someone to *find* some solution to a problem, they'll typically find this much easier than if you had asked them to *prove* something about the solution, even when the two tasks are exactly the same mathematically.

For example, which of the following statements require problem solving?

1. Find the largest prime number less than 100.
2. Prove that 97 is the largest prime number less than 100.

Which task is more difficult? Are the tasks equivalent mathematically?

We could take the view that, in Traditional Math, the *problem* is to verify or refute a theorem. The proof or counterexample is the *solution*.

The remaining words in the Traditional Math framework--such as let, show, hence, follow, and define--are common terminology used in stating theorems and expressing proofs.

Software Development Framework

The Software Development category includes books on Object-Oriented Programming, Database, and Software Engineering. Table 4 lists 9 of the most frequent word groups for this category.

The top four word groups are *object/class*, *system*, d*ata*, and *program/code*. The framework formed by these words is very different from the frameworks for the two preceding categories of books. The word groups *problem* and *solution/solve* do not appear on this list. However, *model/modeling*, *algorithm/method*, and *system* are shared with Applied Math.

This is the only framework that includes the real world concept *data*. Note that no Math objects are on this list.

Table 4: Most frequent words in Software Development books (N=60).

Word Group	Books	Avg StdFreq
object/class	50	412.6
system	54	293.1
data	57	243.5
program/code	54	219.2
process/processing	48	211.9
user/client/customer	48	200.1
model/modeling	49	190.9
algorithm/method	44	182.7
design	46	151.1

Several common Software Development concepts that almost made the list include *software*, *requirement*, and *development*. These words have high frequencies, but are not in enough books (< 42) to qualify for this framework.

The Software Development framework uses models and algorithms to design software systems that integrate programs, data, and users.

5. COMPARING FRAMEWORKS

In the previous section, we presented problem solving frameworks for three book categories in Mathematics and SD. The frameworks are described by of lists of words used frequently in Applied Math, Traditional Math, and Software Development books. The frameworks are not independent, since some words appear on more than one list.

Mathematical Frameworks

The Applied Math and Traditional Math frameworks share 5 word groups that represent widely used mathematical objects--*set*, *function*, *variable*, *equation*, and *point/line*. The remaining words indicate the different nature of the two frameworks. The Applied Math list includes the words *problem*, *solution*, *model*, *algorithm*, and *system*, which describe an approach for solving problems in real world systems. The Traditional Math list includes the words *definition*, *theorem*, and *proof*, along with several common terms used in presenting

theorems and proofs. The emphasis in this framework is on solving mathematical problems through the use of logic.

Software Development and Applied Math

The Software Development framework presents a different approach to problem solving. This list includes *model*, *algorithm*, and *system* from Applied Math, and adds terms that are used in the software development process. In particular, *design*, *class*, *program* and *data* are highlighted. In this framework, users supply the problem, and the completed software product represents the solution. Thus, Software Development combines important Applied Math methods with specific components of the final system.

The relationships between the three frameworks are summarized visually in the Appendix. This figure is a Venn diagram that displays the frameworks as overlapping sets of word groups. No word group appears in all three sets. Moreover, Traditional Math and Software Development have no words in common.

6. DOMAIN-SPECIFIC FRAMEWORKS

We have described the commonalities and differences in the problem solving frameworks for the three book categories. Within each category, several subfields, or domains, are represented. Each of the main frameworks are based on concepts that apply to most of the books in the category. Domain-specific concepts are masked at this level of analysis.

The number of books in each area of Applied Math and Traditional Math is relatively small, so the ability to make domain-specific comparisons is limited. We do highlight the Operations Research framework within Applied Math.

The Software Development book sample covers three domains--Programming, Database, and Software Engineering. Word lists for each of these domains are presented below.

Operations Research Domain

The Applied Math (APM) sample includes 7 books on Operations Research (OR). The top word groups for the OR books, including domain-specific (New) words, are listed in Table 5.

The OR books present a classic variation of the Applied Math framework. OR includes six essential Applied Math word groups--*problem*, *solution/solve*, *model/modeling*, *algorithm/method, value/variable*, and *system*. All but *algorithm/method* have higher average frequencies in the OR domain than in the larger sample of Applied Math books.

The OR framework shares *program/code* with Software Development. It also adds *condition/constraint* and *cost*, which are important in optimization problems.

Table 5: Most frequent words in Operations Research books (N=7).

Word Group	Books	Avg StdFreq	vs. APM
problem	7	443.1	281.1
model/modeling	7	376.5	227.7
value/variable	7	357.9	250.9
solution/solve	7	324.3	239.9
system	6	272.1	173.2
algorithm/method	7	223.7	280.5
program/code	6	198.4	SD
condition/constraint	7	191.1	New
cost	7	178.4	New

Programming Domain

The Software Development sample includes 12 books on Programming (OOP). The top word groups for the OOP books, including one domain-specific word, are presented in Table 6.

Table 6: Most frequent words in Programming books (N=12).

Word Group	Books	Avg StdFreq	vs. SD
object/class	12	674.9	412.6
program/code	12	340.7	219.2
algorithm/method	11	330.9	182.7
value/variable	11	192.4	Math
type	10	186.1	New
set/element	11	170.3	Math
function	9	164.0	Math

The top three Programming word groups--*object/class*, *program/code*, and *algorithm/ method*--are shared with Software Development. These word groups have much higher frequencies in this domain than for the general SD framework.

The single new Programming word is (data) *type*. Three other word groups are borrowed from the Math frameworks. The word *model* is not included in this list because it appeared in the concordances of only 4 OOP books.

Database Domain

The Software Development sample includes 17 Database (DB) books. The top word groups for the DB books are shown in Table 7.

Table 7: Most frequent words in Database books (N=17).

Word Group	Books	Avg StdFreq	vs. SD
data	17	473.7	243.5
object/class	14	445.0	412.6
relation/table	17	360.6	New
database	17	334.9	New
system	17	213.0	293.1
user/client/customer	14	208.4	200.1
query	14	201.8	New
model/modeling	17	201.7	190.9
attribute/column	16	177.0	New
set/element	13	161.2	Math

The Database framework includes four new concepts--*database*, *relation/table*, *query*, and *attribute/column*. Not surprisingly, these words indicate an emphasis on relational databases. The word *data* has a much higher frequency in the DB domain than in the Software Development framework. Also, exactly one Math word group (*set/element*) is on this list.

Software Engineering Domain

The Software Development sample includes 31 Software Engineering (SE) books. The top word groups for the SE books are shown in Table 8.

New SE domain-specific words include *software*, *project*, *requirement*, *development*, and *product*. For most of the word groups shared with the Software Development framework, the average frequency for the SE books is close to the value for the larger SD sample. The likely reason for this similarity is that SE books comprise over half the sample of SD books.

Two exceptions are *object/class* and *system*. In the Programming and Database books, the frequency of *object/class* is substantially higher than in the SE books. This pattern is reversed for *system*.

Table 8: Most frequent words in Software Engineering books (N=31).

Word Group	Books	Avg StdFreq	vs. SD
software	31	402.4	New
system	30	363.8	293.1
process/processing	31	277.8	211.9
object/class	24	262.5	412.6
project	27	243.0	New
requirement	28	242.6	New
program/code	28	219.4	219.2
user/client/customer	27	218.9	200.1
development	31	208.6	New
model/modeling	28	189.1	190.9
product	23	174.0	New
design	30	168.8	151.1
data	28	157.4	243.5

7. SUMMARY AND CONCLUSIONS

The general objective of this study was to examine how problem solving frameworks differ between Mathematics and Software Development. Our approach assumes that words used frequently in a book indicate the mental framework of the author.

We started with a sample of 222 books drawn from three categories: Traditional Math, Applied Math, and Software Development. We chose books that had an Amazon concordance that lists the 100 most frequently used words. Because this research involved problem solving, we eliminated books that did not include *problem* in their concordances, leaving us with 139 books for further study.

We modified the concordance words to compensate for syntactic differences in nouns, verbs, adjectives, and adverbs. We also standardized the word frequencies in each book to make books of various lengths comparable. Finally, we collected words having similar meanings into word groups. Then we generated a list of the most frequent words and word groups in each book category. Based on these lists, we described problem solving frameworks for the categories.

Our results indicate that the frameworks for Traditional Math and Applied Math are fundamentally different. Applied Math uses *models* and *algorithms* to solve real world problems. Traditional Math is more concerned with *theorems* and *proofs*, with the application of logic to solve Math problems.

The Software Development framework shares an emphasis on *models* and *algorithms* with Applied Math, but includes many domain-specific features. Problem solving in Software Development is aimed at creating a successful software product. The methodology involves the design of models and algorithms for *programs* and *data*. Often these models and algorithms are represented visually rather than mathematically, before being implemented in software.

Our findings suggest ways to teach problem solving in Traditional Math, Applied Math, and Software Development courses. In Traditional Math courses, the instructor should introduce an appropriate amount of rigor in theorem proving, consistent with the level of the course. Math majors eventually acquire the mental fortitude to appreciate well-crafted theorems and proofs.

For Applied Math (e.g. Engineering) courses, students prefer to solve real world problems ("story problems") using abstract models and computational algorithms. When presented, proofs can be more informal and descriptive.

For Software Development courses, problems are expressed in terms of models and algorithms that can be used to create software solutions. Here, the nature of the problem and the solution depend on the application. Programming courses involve models for software architecture, as well as ways to specify algorithms. Database courses spend more time on data models, along with query algorithms written in non-procedural SQL. The framework for Software Engineering courses must include the entire life cycle of programming, database, and management activities that lead to the final system.

Computational thinking enthusiasts (Wing, 2006) seem to promote algorithms and computation at the expense of modeling. Conversely, in a recent article on abstract thinking, Kramer (2007) gives greater emphasis to modeling and abstraction:

> Modeling is the most important engineering technique; models help us to understand and analyze large and complex problems.

Teachers of Software Development courses should provide students with substantial exposure to both models and algorithms during the journey from user problems to the eventual software destination.

8. REFERENCES

Andreescu, T., and Gelca (2008), R. Mathematical Olympiad Challenges (2nd ed). Birkhäuser Boston.

Bratko, I. (2001). Prolog Programming for Artificial Intelligence (3rd ed). Addison-Wesley.

Fry, E., et al (1993). The Reading Teacher's Book of Lists (3rd ed). Center for Applied Research in Education.

Kramer, J. (2007). Is abstraction the key to computing? *CACM*, Vol 50, No 4.

McConnell, S. (2004). Code Complete (2nd ed). Microsoft Press.

Michalewicz, Z., and Fogel, D. (2004). How to Solve It: Modern Heuristics (2nd ed). Springer.

Newell, A.; Shaw, J.C.; Simon, H.A. (1959). Report on a general problem-solving program. *Proceedings of the International Conference on Information Processing*.

Polya, G. (1945). How To Solve It. Princeton University Press.

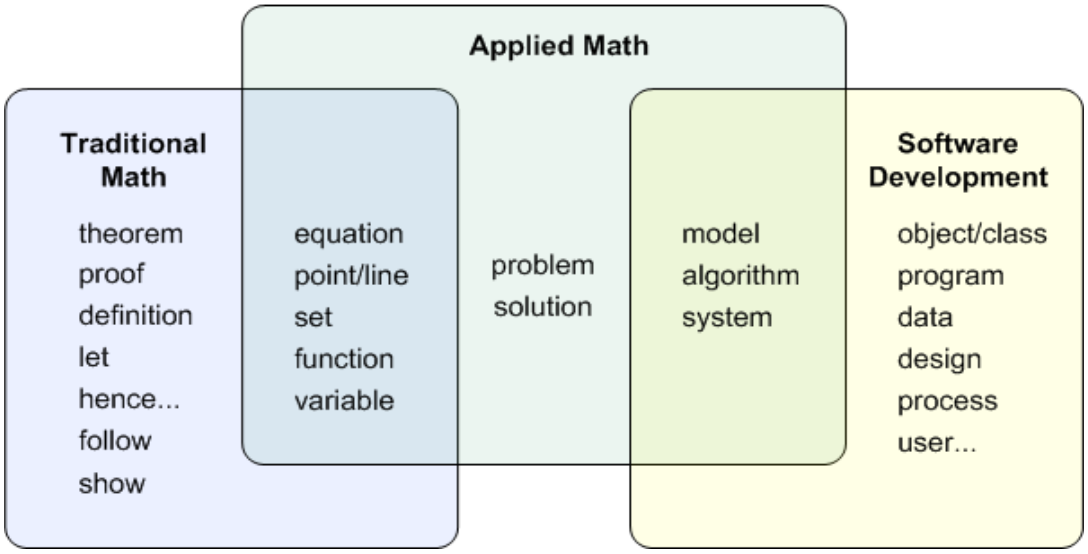
Velleman, D. (1994). How to Prove It: A Structured Approach. Cambridge University Press.

Wing, J. (2006). Computational thinking. *CACM*, Volume 49, No. 3.

Zeitz, P. (2006). The Art and Craft of Problem Solving (2nd ed). Wiley.

9. APPENDIX

Mathematics and Software Development Frameworks



The Learning and Productivity Benefits to Student Programmers from Real-World Development Environments

Justin C. W. Debuse
jdebuse@usc.edu.au

Meredith Lawley
mlawley1@usc.edu.au

Faculty of Business,
University of the Sunshine Coast
Maroochydore DC, Australia

Abstract

Existing research and practice in software development environments shows no clear consensus on the most appropriate development tools to use; these may range from simple text editors through teaching-oriented examples to full commercial integrated development environments (IDEs). This study addresses this gap by examining student perceptions of two development environments at opposite ends of the complexity spectrum. The results, gathered over several years using students at a range of experience levels, suggest that complex commercial IDEs are appropriate for programming education, even for entry-level students. Indeed, they offer a range of features that may improve the understanding and productivity of students. However, given the greater simplicity of simple text editors and potential for students to become overly dependent upon the support mechanisms provided by IDEs, teaching IDEs in combination with simple text editors appears to offer an ideal combination to maximize learning opportunities and student employability.

Keywords: integrated development environment, IDE, programming, learning, teaching

1. INTRODUCTION

A key challenge for ICT educators is to teach underlying concepts, such as structured analysis and data modelling (Tastle & Russell, 2003), so that students have transferable skills and deep understanding. However, the employment market demands specific skills such as ASP (Colomb, Death, Brown, & Clarkson, 2001) or Java (Liu, Liu, Lu, & Koong, 2003), and a compromise must therefore be found between technology-specific details and fundamental principles. Programming courses must strike this balance not only for the language but also the development environment. For example, the popular Java language can be taught using a range of environments, from a command line interface and text editor through a simple teaching-oriented integrated development environment (IDE) to a complex commercial IDE. The selected environment must fulfil a number of different and potentially conflicting criteria: employment market demand, learning support and ease of use.

The demand by employers appears highest for text editors (Russell, 2005a), although users show preference for using IDEs (Computerworld,

2005; Russell, 2005a). Learning support is high in teaching-oriented systems such as BlueJ, which have been found to assist student understanding of the object-oriented paradigm (Van Haaster & Hagan, 2004) and allow unusual topic orderings to be used (Murray, Heines, Moore, Trono, Kolling, Schaller, & Wagner, 2003). However, commercial IDEs such as JBuilder can also offer considerable teaching and learning support (Liang, 2005).

The ease of use of a development environment is likely to be affected by its complexity. Indeed, the complex nature of commercial IDEs has been used to justify using teaching-oriented alternatives (Kölling, Quig, Patterson, & Rosenberg, 2003), and may explain the high popularity of text editors for education (Russell, 2005a) and low usage of CASE tools for application development at both undergraduate and postgraduate levels (Chinn, Lloyd, & Kyper, 2005). However, there appears to be no clear consensus on whether IDE usability should be criticised (Kline, Seffah, Javahery, Donayee, & Rilling, 2002; Murray et al., 2003; Reis & Cartwright, 2004; Seffah & Rilling, 2001) or praised (Dujmovic & Nagashima; Murray et al., 2003), and students have not shown a preference for specific development environments (Russell, 2005a). Programming textbooks show similar dissent; examples exist that use commercial IDEs such as JBuilder (Liang, 2004), teaching-oriented IDEs (Barnes & Kolling, 2008), text editors (Farrell, 2003), or allow educators to choose between a text editor and IDE (Liang, 2009).

The impact of development environments upon student learning and understanding from a formative perspective is one of the least studied areas of IDE research (Gross & Powers, 2005). Existing studies, particularly those which measure student performance directly (Kordaki, 2010; Vogts, Calitz, & Greyling, 2008), have examined the educational suitability of IDEs to only a limited level of granularity; for example, Kordaki (2010) examines different development environments across broad areas such as the quality of students' code, rather than the features of the environments in detail. Further, although educational IDEs appear to yield improvements in student understanding (Rigby & Thompson, 2005; Van Haaster & Hagan, 2004; Xinogalos, Satratzemi, & Dagdilelis, 2006) and programming performance (Kordaki, 2010; Vogts et al., 2008), they require room in the syllabus to be found for students to convert to real world environments, which is unlikely to prove easy (Xinogalos et al., 2006).

This study therefore attempts to extend existing work to a finer level of granularity, in order to clarify the selection of development environments for programming education by determining whether the learning support and ease of use of an environment for which significant employment market demand exists are sufficiently strong for it to be successfully used without going through the intermediate step of using a teaching-oriented IDE. The environment used is Borland's JBuilder/Together, one of the object-oriented analysis and design market leaders (Blechar, 2004) and now incorporated into the popular Eclipse environment. The students examined are from a single regional university and thus likely to have greater requirements for learning support and ease of use than their metropolitan equivalents. Moreover, the students are examined at three stages in their programming education to determine the performance of the environment across a range of experience levels.

2. METHOD

The commercial IDE examined within this study was JBuilder from Borland; this was supplied within the teaching laboratories and used to deliver lectures and tutorials. Programming students were studied from 2005 to 2008; during 2007 JBuilder was incorporated into the Eclipse system, which offered very similar functionality. Students were surveyed across the three groups described below, to allow differences between programmers across a range of experience levels to be investigated.

Group 1: Introductory Java Programming

The first group of students took an introductory course in Java programming, held during semester 2 each year from 2005 to 2008. Students' attitudes to the IDE were surveyed using an instrument adapted from (Hede, 2005); the 2008 version is presented within Appendix 2. The first section established their prior knowledge of programming and IDEs, using questions adapted from (Russell, 2005b). The section included items determining whether JBuilder and Java were the most commonly used development environment and language, to confirm that students met the requirements of the study.

The second section investigated the complexity of JBuilder, since this drawback of commercial IDEs has been used to justify using teaching-oriented IDEs for education (Kölling et al., 2003). The statements in Table 1, labelled JBA, were used to assess how this affected students, both when they began learning to use JBuilder and once they had become proficient in using it; the available responses ranged from five (strongly agree) through three (neutral) to one (strongly disagree). Statement JBA1 is similar to the learnability scale from the Software Usability Measurement Inventory (SUMI) (Kirakowski & Corbett, 1993), described by (Kline et al., 2002). However, Kline et al. (2002) also cite a minimum sample size of 50 subjects to be confident of the results (Nunnally & Bernstein, 1994). This exceeds the current student numbers for the courses examined within this paper, and so SUMI scales were not used. Statements JBA3 to JBA6 were instead added to the questionnaire to cover the missing SUMI scales of affect, helpfulness, efficiency and control respectively.

The instrument also included questions, labelled JBB, to measure how aspects of the IDE improved or impaired understanding of the course concepts and productivity in producing its required deliverables. Five point Likert scales were again used, with one set (labelled A) determining the effect on understanding and a second (labelled B) measuring the effect on productivity; however, unlike the previous scales, the values ranged from 5 (strong improvement) through 3 (no effect) to 1 (strong impairment), together with 0 (if they have never used the feature or respond that this is not applicable). The IDE aspects investigated are shown in Table 2; some were adapted from (Dujmovic & Nagashima; Russell, 2005a; Storey, Michaud, Mindel, Sanseverino, Damian, Myers, German, & Hargreaves, 2003), and features absent from the university Java programming courses were excluded. The instrument has some overlap with Russell's (2005b) survey, although it examines the IDE aspects at a greater level of detail.

Preliminary results from this study suggested that students may become over-reliant upon the support mechanisms offered by JBuilder. The course examined was therefore revised after its 2005 intake to use a text editor (Programmers Notepad) initially, followed by JBuilder, rather than using JBuilder throughout. The second survey and its successors thus contained additional questions: PNA, which applied the complexity statements in Table 1 to the text editor rather than JBuilder; PNB, which investigated similar IDE aspects to those listed within Table 2, but aimed at the text editor rather than JBuilder (with a corresponding reduction in the number of aspects due to the more limited functionality of the text editor); and JBPN, adapted from (Russell, 2005b), which determines which environment students would have preferred to use to learn programming, together with which environment they would rather currently program with (the JBPN questions were administered in a separate survey during 2006 but incorporated into the main survey from 2007 onwards).

Group 2: Intermediate Java Programming

The second group of students took an intermediate level follow-on from the introductory Java programming course taken by group 1, held during semester 1 from 2006 to 2008. The course taken by group 1 was a prerequisite for the course taken by this group; a number of students from group 1 would therefore subsequently join group 2. For example, 68% of the students who took the intermediate course during 2006 had previously taken the introductory course during 2005. The group 1 instrument was applied for the group 2 students with minor modifications corresponding to their differing course enrolments.

Group 3: Architecture & Systems Integration

The third group of students took a capstone architecture and systems integration course in semester 2 2005, where programming skills were applied to systems integration tasks, using JavaScript and the Notepad text editor. The survey was only administered in 2005, and used an adaptation of the group 1 instrument which was modified to reflect different course enrolments and the use of Notepad in place of Programmers Notepad; further, section PNB (IDE aspects) was omitted due to the limited functionality of Notepad.

Analysis of Results

Missing values were identified as such when the data was entered and excluded from calculations on a pairwise basis; this means that the response for a student was only excluded from a calculation if data required by that calculation

was missing. Responses of zero for the IDE aspect statements were also treated as missing, as this value represented that the statement was not applicable or that the respondent had never used the feature. If a respondent indicated that they had not used a development environment but then proceeded to respond to items regarding the environment then these responses were excluded and treated as missing. Similarly, for each aspect statement there are two questions, covering understanding and productivity; if a response to either of these questions indicated that the aspect was never used or was not applicable then both were treated as missing data.

Hypothesis Testing

The results of the survey within this study were used for hypothesis testing, using a similar approach to that described in (Debuse, Lawley, & Shibl, 2007, 2008; Stevens & Jamieson, 2002). For the complexity assessment statements (labelled JBA and PNA), two hypotheses were formed; the first was that respondents agreed with the statement and the second was that respondents disagreed. Such an approach was used in place of a single hypothesis since responses could indicate agreement, neutrality or disagreement; thus, a hypothesis based on agreement may fail to hold, but this does not necessarily indicate disagreement. Specifically, for the first hypothesis to hold, the response must be greater than three; this equates to a response above 'Neutral', which may be high enough to equate to 'Tend to Agree' or 'Strongly Agree'. For the second hypothesis to hold, the response must be three or less; this equates to a response of 'Neutral', 'Tend to Disagree' or 'Strongly Disagree'. The 95% confidence interval for the mean response value was computed, and its lower and upper bounds were used to test the first and second hypotheses respectively. For example, consider lower and upper bounds for the 95% confidence interval for the mean response to statement JBA1 of 3.4 and 5.1 respectively. Such values would cause the first hypothesis for JBA1 to be accepted, since 3.4 is greater than three, and the second to be rejected, since 5.1 is greater than three. Such a result would lead to the conclusion that respondents agreed with JBA1.

Similar hypotheses were formed and tested for the aspect statements (labelled JBB and PNB). For each aspect, two hypotheses were again formed; the first was that it had improved respondents' understanding of the course concepts and the second was that it had impaired them. A third and fourth hypothesis were similarly formed for each aspect; these concerned its improvement or impairment respectively to respondents' productivity. The lower end of the 95% confidence interval of the mean response to the understanding scale had to exceed three for the first hypothesis to hold; for the second, the upper end had to be three or less. Similarly, lower and upper ends of the 95% confidence interval of the mean response to the productivity scale were calculated. If the lower exceeded three then the first hypothesis held; an upper value of three or less caused the second hypothesis to hold.

3. RESULTS

The total responses received across all surveys totalled 167; the breakdown of these, together with key demographic information, hypotheses and preferred development environments are presented in the following sections.

Demographics

Table 3 shows that all groups represent junior programmers, with at most one to three years' experience. Group 1 are the most junior, with responses being mainly less than one year rather than the one to three years for groups 2 and 3. All groups apart from 3 report JBuilder as the IDE used; the majority of groups used JBuilder the most. The JBuilder environment is thus very familiar to the students, and all groups had the most programming experience in Java.

The demographics thus suggest that the students examined meet the requirements of this study, namely junior programmers at differing points in their programming education, with experience in Java and JBuilder.

Hypotheses

The left half of Table 4 shows the hypotheses that held for each group; hypothesis 1 (H1) holding is denoted by 1, and hypothesis 2 (H2) holding is denoted by 2. An empty cell shows that neither hypothesis held; nor does NA show that the hypothesis was tested for the specified year.

The right half of Table 4 summarises the total number of times that hypotheses 1 and 2 held

across all groups, along with the percentage of non-NA groups for which hypotheses 1 and 2 held; these summaries are also given for group 1 across all years together with group 2 across all years. Grey denotes rows for which hypothesis 1 holds for every group examined, and bold denotes rows for which hypothesis 1 holds for no group examined.

The Notepad results for group 3 are not included in the table as they were only recorded for a single group. For these PNA statements, hypothesis 1 held for statements PNA1 and PNA2; hypothesis 2 held for PNA4.

Table 4 suggests that, for some JBuilder aspects (denoted by grey rows, starting at aspect JBB1A), every group examined found them to yield understanding and/or productivity benefits. Of these aspects, the following yielded both understanding and productivity benefits:

- Automatic bracket/brace matching
- Automatic code formatting
- Automatic completion of words in programs
- Display of parameter lists
- Automatic code colouring
- Automatic syntax error reporting
- Code audit warnings
- Breakpoint / line by line execution in debugging
- Variable value viewing / modification in debugging

The remainder of the aspects for which every group examined reported benefits yielded productivity but not understanding improvements:

- Automatic creation of program code
- Automatic generation of Javadoc comments
- Display of line numbers

There was no universal agreement on any other area of JBuilder or Programmers Notepad, but one area of Programmers Notepad (PNB4A – the benefit of case conversion within Programmers Notepad to understanding) was not perceived to give understanding/productivity benefits within any group. Further, respondents disagreed with one of the Programmers Notepad utility / ease of use statements (PNA4 – Programmers Notepad gives me assistance in its use) in one group, although they agreed with this within another.

The results can also be analysed in terms of totals over all group 1 students compared to totals over all group 2 students. In addition to the grey cells noted above (which will have 100% hypothesis 1 coverage for both these groups), these groups have 100% hypothesis 1 agreement for the following statements relating to JBuilder:

- Automatic program code creation improves understanding (group 1 and group 2)
- Code creation wizards improve productivity (group 1 only)
- Sync edit tool, which allows all instances of a variable name to be changed by editing a single instance of the name, improves understanding and productivity (group 1 only)
- Line number display improves understanding (group 1 and group 2)
- Automatic Javadoc creation improves understanding (group 1 and group 2)
- Javadoc integration improves understanding and productivity (group 2 only)
- The automatic link between Java and UML improves understanding (group 1 only)

Further, group 1 has 100% agreement with hypothesis 1 for the following statements relating to Programmers Notepad:

- Learning to use Programmers Notepad is straightforward
- Programmers Notepad automatic code colouring improves productivity
- Programmers Notepad display of line numbers improves understanding and productivity

Items for which no agreement was shown over the group 1 and/or group 2 groups, in addition to PNB4A described above, are:

- I feel I am in control of JBuilder when I use it (group 2 only)
- The automatic Java/UML link improves understanding and productivity (group 2 only)
- Once you have learned to use Programmers Notepad then producing Java software with it is straightforward (group 2 only)
- Using Programmers Notepad is enjoyable (group 2 only)

- Programmers Notepad gives me assistance in its use (group 2 only)
- The amount of time and effort required to perform tasks in PN is low (group 2 only)
- Programmers Notepad bookmarks improve understanding and productivity (group 2 only)
- Programmers Notepad display of line numbers improves productivity (group 2 only)

Summarising the differences between groups 1 and 2, it appears that the two have similar views regarding JBuilder, with differences in terms of 100% agreement only occurring for a small number of items. Differences are more extreme for Programmers Notepad, with only group 1 having some items which were agreed with across all years, and only group 2 having some items for which agreement was not found for any year. The group 1 students therefore appear much more positively disposed towards Programmers Notepad than group 2.

The most experienced programmers (group 3) were more positively disposed towards text editors than group 2, finding Notepad easy to use and produce Java software with, although unsurprisingly they did not find it supportive. However, they also found JBuilder to be easy to produce software with, and found it enjoyable and supportive to use. Further, the majority of the features of JBuilder proved to be useful to both their understanding and productivity.

Development Environment Preferences

Table 5 shows that there is no consensus across all groups in terms of the preferred environment to learn programming, although all but one prefer a combination of JBuilder and Programmers Notepad. All groups preferred to use JBuilder to do programming now (one of these was multi-modal).

4. DISCUSSION

The results suggest that students perceive considerable benefits from a real-world integrated development environment (IDE) such as JBuilder, which represents their preferred option for programming; however, a combination of text editor and IDE appear to be preferable for learning purposes. Students' responses overall are very positive for almost all areas examined within this study; the only negative responses were for case conversion and support within the text editor. All three groups of students appeared not to believe that any of the surveyed JBuilder IDE aspects impaired their understanding of course concepts or productivity. Indeed, the majority of the JBuilder aspects examined were found to improve productivity and/or understanding by all groups, and every item was present in at least one group.

A text editor appears particularly appealing to the group 1 students, particularly in terms of its reduced complexity; the more experienced group 2 students appear to be less positively inclined towards it, although the most experienced group (3) appeared to view such systems more favourably. However, the groups have similar views regarding JBuilder, and a number of its features in areas such as debugging and simple code writing support appear to yield understanding and productivity benefits across all groups and years. Further, features such as more sophisticated code writing support appear to have universal benefit, but only in terms of productivity; this is unsurprising given the potential for such support to deny students the opportunity to learn how to create code.

The most experienced programmers (group 3) were more positively disposed towards text editors than group 2, finding Notepad easy to use and produce Java software with, although unsurprisingly they did not find it supportive. However, they also found JBuilder to be easy to produce software with, and found it enjoyable and supportive to use.

The preference for simplicity by entry level students is unsurprising given the documented unsuitability of professional IDEs for teaching given their complexity (Reis & Cartwright, 2004). Complicated aspects of the Java language may also prove distracting (Reis & Cartwright, 2004); this may explain why many students in this find automated code creation to improve their understanding, since at a conceptual level the language complexities may impair learning.

The results contain a number of points of interest. Firstly, despite the study being held at a regional university, at which student quality is unlikely to be higher than at metropolitan centres, the respondents did not find the JBuilder IDE complex; indeed, both the novice

and experienced programmers found it straightforward to produce software with, and the novices found JBuilder easy to learn. This may be partially explained by the approach used to teach the programming courses at the university, with JBuilder being covered extensively throughout lectures, tutorials and the course text.

The overall status of JBuilder as the preferred environment for students to use currently matches existing research (Russell, 2005a). The results also overlap with those testing a visual programming language, where most students perceived improvements in their understanding and found the environment helpful (Collins & Fung, 2002). The results of Kline et al. (2002), who found that experienced programmers viewed their IDE as helpful and efficient, also support this study. However, unlike this study their programmers did not find the IDE easy to learn. Further, the preferred option identified within this study of combining a text editor and IDE for learning is unsurprising given the lack of consensus in existing research on whether text editors or IDEs would be preferred for training purposes (Russell, 2005a).

It is surprising that the majority of the IDE aspects examined were found to improve understanding and/or productivity, with debugging support being particularly useful; this contradicts existing research suggesting that integrated debugging is the least useful feature for both learning and production programming (Russell, 2005a). Other popular features such as automated code completion and Javadoc integration also proved unpopular (Russell, 2005a), although the popularity of areas such as bracket matching and syntax highlighting is supported by existing research (Russell, 2005a). Further, these results are supported by studies indicating that over 85% of user requirements are satisfied by current IDEs, with JBuilder offering the best performance (Dujmovic & Nagashima), and that the JBuilder debugging support is useful for teaching (Liang, 2005; Murray et al., 2003). Similarly, the BlueJ development environment has also improved students' understanding of object-oriented concepts (Van Haaster & Hagan, 2004); correspondingly, the educational IDE objectKarel yielded improvements in students' perceptions of their understanding (Xinogalos et al., 2006), and students using the LECGO for C educational IDE programmed more successfully than using a non-teaching environment or pencil and paper (Kordaki, 2010). Students' performance using the SimplifIDE educational plug IDE improved the programming performance of students compared to a professional IDE; their understanding, measured by assessment grades, was only superior using the educational IDE for weaker students (Vogts et al., 2008). The Gild educational plug in for Eclipse, when compared to Eclipse, appears to improve students' perceptions of their understanding but not their programming performance or productivity (Rigby & Thompson, 2005). Improvements in students' perceptions of their understanding have also been attributed to the Eclipse IDE (Hanks, 2006).

A study examining actual usage data for the Eclipse IDE across 41 Java software developers using the Mylar Monitor plug-in (Murphy, Kersten, & Findlater, 2006) gave strong support for the automatic program word completion that was found to be so important to understanding and productivity; the developers used such completion as often as popular editing commands such as copy and paste. The importance of debugging identified within this study was also supported (Murphy et al., 2006). Further, the sync edit tool, which was particularly popular with entry-level programmers within this study and allows all instances of a variable name to be changed by editing a single instance of the name, was part of the most popular refactoring command (rename), which was used by all respondents (Murphy et al., 2006).

The study has a number of limitations. Firstly, although students are surveyed at three different points in their education, the longer term effects of the IDE are not examined. Secondly, the most experienced group of students do not use JBuilder within their course; however, over half of them have used JBuilder, although many of these will be relying on memories of past courses. Thirdly, the study is restricted to a single organisation and single example of each tool. This approach, though used in a number of existing studies (Collins & Fung, 2002; Kordaki, 2010; Xinogalos et al., 2006), restricts the extent to which the results can be generalised, since specific details such as courses, tools and student demographics may contribute to the results, particularly as IDEs are presented very positively to students within the programming courses of this study; this does however offer the advantage of limiting potential confounding effects from areas such as

instructor or syllabus variations. Fourthly, any development environment that is successfully used by students will have a positive effect on their understanding, and the sequential usage of the two environments examined means that they will impact upon students at different learning stages. Finally, the study examines only students' perceptions rather than actual usage data. Although student perceptions of software usability and its effects on their own productivity would be unlikely to give inaccurate responses, understanding has the potential to be more problematic. This is because students' perceptions of their own understanding may not correlate well with their actual levels; also, the surveys query students' understanding of course concepts without giving details of the specific learning outcomes and course objectives to which such concepts relate, which gives the potential for weaker students to not realise that they have missed certain concepts; the objectives will also not be the same across all of the courses. However, the overall approach is not unusual, with a number of existing studies measuring student perceptions of understanding (Collins & Fung, 2002; Hanks, 2006; Rigby & Thompson, 2005; Xinogalos et al., 2006). Further, although weaker students have demonstrated a tendency to overrate themselves compared to educators, no consistent over or underrating has been found (Boud & Falchikov, 1989); indeed, a weak positive correlation has been found between student self assessment and educator assessment (Falchikov & Boud, 1989), and a review of existing work suggests that in the majority of studies the number of cases where student and staff marks agreed outnumbered those where they disagreed (Boud & Falchikov, 1989). Moreover, the number of development environment features for which understanding is examined is too large to feasibly investigate directly.

5. CONCLUSIONS

This study has highlighted a number of areas of importance for software development education. It appears that university students can learn introductory programming using a complex commercial IDE, without requiring the intermediate step of using an educational environment. Moreover, most of the IDE aspects improve their understanding and/or productivity. However, some of these mechanisms can deny students the opportunity to learn key programming skills that environments with limited support require. This suggests that the use of a text editor in addition to a complex IDE would be an ideal combination to maximize learning and future employment opportunities. However, institutional constraints such as the availability of IT service department support clearly need to be taken into account if such approaches are to be adopted.

Future research may determine how students' perceptions of the utility of IDE features correlate with their actual usage data, and where the perceived benefits translate into actual performance enhancements.

6. REFERENCES

Barnes, D. J., & Kolling, M. (2008). Objects First with Java (4th ed.). Prentice Hall / Pearson.

Blechar, M. (2004). Market Details for OOA&D Tools, Update for 2005. *Gartner Research*.

Boud, D., & Falchikov, N. (1989). Quantitative studies of student self-assessment in higher education: a critical analysis of findings. *Higher education, 18*(5), 529-549.

Chinn, S. J., Lloyd, S. J., & Kyper, E. (2005). Contemporary Usage of CASE Tools in U. S. Colleges and Universities. *Journal of Information Systems Education, 16*(4), 429-436.

Collins, T. D., & Fung, P. (2002). A visual programming approach for teaching cognitive modelling. *Computers & Education, 39*(1), 1-18.

Colomb, R., Death, B., Brown, A., & Clarkson, A. (2001). Trends in Computing Jobs - 2001. Retrieved 13 May, 2003, from www.itee.uq.edu.au/~colomb/Jobs-Anal-2001.html

Computerworld (2005). Computerworld Development Survey gives nod to C#. Retrieved 30 August, 2005, from http://www.computerworld.com/developmenttopics/development/story/0,10801,100542,00.html

Debuse, J., Lawley, M., & Shibl, R. (2007). The Implementation of an Automated Assessment Feedback and Quality Assurance

System for ICT Courses. *Journal of Information Systems Education, 18*(4), 491-502.

Debuse, J., Lawley, M., & Shibl, R. (2008). Educators' perceptions of automated feedback systems. *Australasian Journal of Educational Technology, 24*(4), 374-386.

Dujmovic, J., & Nagashima, H. Evaluation of IDE's for Java Enterprise Applications. Retrieved 15 August, 2005, from http://www.seas.com/downloadUNReg/IDE6p.pdf

Falchikov, N., & Boud, D. (1989). Student self-assessment in higher education: A meta-analysis. *Review of Educational Research, 59*(4), 395-430.

Farrell, J. (2003). Java Programming (2nd ed.). Course Technology, Boston, Massachusetts.

Gross, P., & Powers, K. (2005). *Evaluating assessments of novice programming environments.* Paper presented at the First international workshop on Computing education research.

Hanks, B. (2006). Using Eclipse in the classroom. *Journal of Computing Sciences in Colleges, 21*(3), 118-127.

Hede, A. (2005). Personal Communication.

Kirakowski, J., & Corbett, M. (1993). SUMI: The Software Measurement Inventory. *British Journal of Educational Technology, 24*(5), 210-212.

Kline, R., Seffah, A., Javahery, H., Donayee, M., & Rilling, J. (2002, September 3-6). *Quantifying Developer Experiences via Heuristic and Psychometric Evaluation.* Paper presented at the IEEE Symposia on Human Centric Computing Languages and Environments, Arlington, VA.

Kölling, M., Quig, B., Patterson, A., & Rosenberg, J. (2003). The BlueJ system and its pedagogy. *Journal of Computer Science Education, Special issue on Learning and Teaching Object Technology, 13*(4).

Kordaki, M. (2010). A drawing and multi-representational computer environment for beginners' learning of programming using C: Design and pilot formative evaluation. *Computers & Education, 54*(1), 69-87.

Liang, Y. (2004). Introduction to Java Programming with JBuilder (3rd ed.). Prentice Hall.

Liang, Y. (2005). Learning Java Effectively with JBuilder. In *Introduction to Java Programming* (7th ed.).

Liang, Y. (2009). Introduction to Java Programming, Comprehensive (7th ed.). Prentice Hall.

Liu, X., Liu, L., Lu, J., & Koong, K. (2003). An Examination of Job Skills Posted on Internet Databases: Implications for Information Systems Degree Programs. *Journal of Education for Business, 78*(4), 191-196.

Murphy, G. C., Kersten, M., & Findlater, L. (2006). How Are Java Software Developers Using the Eclipse IDE? *IEEE Software, 23*(4), 76-83.

Murray, K., Heines, J., Moore, T., Trono, J., Kolling, M., Schaller, N., & Wagner, P. (2003). *Panel on Experiences with IDEs and Java Teaching: What Works and What Doesn't.* Paper presented at the ACM SIG CSE 8th International Conference on Innovation and Technology in Computer Science Education, Thessaloniki, Greece.

Nunnally, J., & Bernstein, I. (1994). Psychometric Theory (3rd ed.). McGraw-Hill, New York.

Reis, C., & Cartwright, R. (2004). *Taming a professional IDE for the classroom.* Paper presented at the SIGCSE technical symposium on Computer Science Education.

Rigby, P. C., & Thompson, S. (2005). *Study of novice programmers using Eclipse and Gild.* Paper presented at the OOPSLA workshop on Eclipse technology eXchange.

Russell, J. (2005a). *Do the benefits of learning Java using an IDE outweigh the in-depth understanding gained by learning with a text editor only?* MSc Thesis, University of Liverpool.

Russell, J. (2005b). MSc Student Survey: Do the benefits of using an IDE to learn Java outweigh the understanding gained by learning with a text editor only. Retrieved 30 August (from Google cache dated 27 February), 2005, from http://www.jeremyrussell.co.uk/studentsurveyquestion.jsp

Seffah, A., & Rilling, J. (2001). *Investigating the Relationship between Usability and Conceptual Gaps for Human-Centric CASE Tools.* Paper presented at the IEEE Symposium on Human-Centric Computing Languages and Environments, Stresa, Italy.

Stevens, K., & Jamieson, R. (2002). The Introduction and Assessment of Three Teaching Tools (WebCT, MindTrail, EVE) into a Post Graduate Course. *Journal of Information Technology Education, 1*(4), 233-252.

Storey, M., Michaud, J., Mindel, M., Sanseverino, M., Damian, D., Myers, D., German, D., & Hargreaves, E. (2003). *Improving the Usability of Eclipse for Novice Programmers.* Paper presented at the Object-Oriented Programming, Systems, Languages and Applications (OOPSLA), Anaheim, California, USA.

Tastle, W., & Russell, J. (2003). Analysis and Design: Assessing Actual and Desired Course Content. *Journal of Information Systems Education, 14*(1), 77-90.

Van Haaster, K., & Hagan, D. (2004, June). *Teaching and Learning with BlueJ: an Evaluation of a Pedagogical Tool.* Paper presented at the Information Science + Information Technology Education Joint Conference, Rockhampton, QLD, Australia.

Vogts, D., Calitz, A., & Greyling, J. (2008). *Comparison of the effects of professional and pedagogical program development environments on novice programmers.* Paper presented at the Annual research conference of the South African Institute of Computer Scientists and Information Technologists on IT research in developing countries.

Xinogalos, S., Satratzemi, M., & Dagdilelis, V. (2006). An introduction to object-oriented programming with a didactic microworld: objectKarel. *Computers & Education, 47*(2), 148-171.

Appendix 1: Tables

Table 1. Complexity assessment statements

Statement	Description
JBA1	Learning how to use JBuilder is straightforward.
JBA2	Once you have learned how to use JBuilder then producing Java software with it is straightforward.
JBA3	Using JBuilder is enjoyable.
JBA4	JBuilder gives me assistance in its use.
JBA5	The amount of time and effort required to perform tasks using JBuilder is low.
JBA6	I feel I am in control of JBuilder when I use it.

Table 2. IDE Aspects

Aspect	Description
JBB1	Automatic code formatting.
JBB2	Automatic completion of words within programs.
JBB3	Parameter list display
JBB4	Automatic creation of code such as missing curly brackets.
JBB5	Code creation wizards for tasks such as class creation.
JBB6	An editing mode that allows all instances of a variable name to be changed by editing a single instance of the name.
JBB7	Integrated help system.
JBB8	Automatic code colouring.
JBB9	Line numbering.
JBB10	Automatic syntax error reporting.
JBB11	Code audit warnings.
JBB12	Deprecation warnings
JBB13	Debugging support through breakpoints and line-by-line execution.
JBB14	Debugging support through viewing and modifying variable values.
JBB15	Automatic bracket matching.
JBB16	Automatic generation of Javadoc comments.
JBB17	Javadoc integration through automatic creation and view of HTML associated with Javadoc comments
JBB18	Automatic two-way links between UML diagrams and their associated program code.

Table 3. Mode responses to demographics (percentage giving mode response in brackets)

Year	2005		2006			2007		2008	
Group	1	3	2	1	1[a]	2	1	2	1
N	10	16	10	31	24	19	22	8	27
How much programming experience do you currently have (years)?	<1 (50%)	1-3 (69%)	1-3 (60%)	1-3 (35.5%)	<1 (33.3%)	1-3 (52.6)	<1 (50%)	<1 (37.5%), 1-3 (37.5%) ie bimodal	<1 (48.1%)
Which Integrated Development Environments (IDEs) have you used?	JBuilder (90%)	A text editor (75%)	JBuilder (100%)	JBuilder (93.5%)	JBuilder (100%)	JBuilder (94.7%)	JBuilder (90.9%)	JBuilder (87.5%)	JBuilder (96.3%)
Which Integrated Development Environment (IDE) do you use the most?	JBuilder (80%)	JBuilder (50%)	JBuilder (100%)	JBuilder (77.4%)	JBuilder (87.5%)	JBuilder (94.7%)	JBuilder (63.6%)	JBuilder (37.5%), Eclipse (37.5%) ie bimodal	Eclipse (51.9%)
Which programming language do you have the most experience in?	Java (70%)	Java (75%)	Java (100%)	Java (80.6%)	Java (75%)	Java (78.9%)	Java (72.7%)	Java (75%)	Java (77.8%)

[a]This survey of the preferred environment was run separately to the rest of the survey during 2006

Table 4. Results of hypotheses (grey denotes rows for which hypothesis 1 holds for every group examined and bold denotes rows for which hypothesis 1 holds for no group examined)

Hypotheses holding for each aspect (1 & 2 denote hypothesis number; empty cells and NA denote no hypothesis holding and no testing respectively)										Summary data for hypotheses holding for each aspect						
Year	2005		2006			2007		2008								
Group	1	3	2	1	1[a]	2	1	2	1	H1total	H2 total	H1%[b]	H1 total (1)[c]	H1 total (2)[c]	H1% (1)[c]	H1% (2)[c]
JBA1	1			1	NA	1			1	4	0	50	3	1	75	33.33
JBA2	1	1		1	NA	1			1	5	0	62.5	3	1	75	33.33
JBA3		1		1	NA	1			1	4	0	50	2	1	50	33.33
JBA4		1		1	NA	1	1	1	1	6	0	75	3	2	75	66.67
JBA5					NA	1			1	2	0	25	1	1	25	33.33
JBA6		1			NA				1	2	0	25	1	0	25	0
JBB1A	1	1	1	1	NA	1	1	1	1	8	0	100	4	3	100	100
JBB1B	1	1	1	1	NA	1	1	1	1	8	0	100	4	3	100	100
JBB2A	1	1	1	1	NA	1	1	1	1	8	0	100	4	3	100	100
JBB2B	1	1	1	1	NA	1	1	1	1	8	0	100	4	3	100	100
JBB3A	1	1	1	1	NA	1	1	1	1	8	0	100	4	3	100	100
JBB3B	1	1	1	1	NA	1	1	1	1	8	0	100	4	3	100	100
JBB4A	1		1	1	NA	1	1	1	1	7	0	87.5	4	3	100	100
JBB4B	1	1	1	1	NA	1	1	1	1	8	0	100	4	3	100	100
JBB5A			1	1	NA	1	1		1	5	0	62.5	3	2	75	66.67
JBB5B	1	1	1	1	NA	1	1		1	7	0	87.5	4	2	100	66.67
JBB6A	1	1	1	1	NA	1	1		1	7	0	87.5	4	2	100	66.67
JBB6B	1	1	1	1	NA	1	1		1	7	0	87.5	4	2	100	66.67
JBB7A		1		1	NA	1	1		1	5	0	62.5	3	1	75	33.33
JBB7B		1		1	NA	1	1		1	5	0	62.5	3	1	75	33.33
JBB8A	1	1	1	1	NA	1	1	1	1	8	0	100	4	3	100	100
JBB8B	1	1	1	1	NA	1	1	1	1	8	0	100	4	3	100	100
JBB9A	1		1	1	NA	1	1	1	1	7	0	87.5	4	3	100	100
JBB9B	1	1	1	1	NA	1	1	1	1	8	0	100	4	3	100	100
JBB10A	1	1	1	1	NA	1	1	1	1	8	0	100	4	3	100	100
JBB10B	1	1	1	1	NA	1	1	1	1	8	0	100	4	3	100	100
JBB11A	1	1	1	1	NA	1	1	1	1	8	0	100	4	3	100	100
JBB11B	1	1	1	1	NA	1	1	1	1	8	0	100	4	3	100	100
JBB12A		1		1	NA	1	1	1	1	6	0	75	3	2	75	66.67
JBB12B		1		1	NA	1	1	1	1	6	0	75	3	2	75	66.67
JBB13A	1	1	1	1	NA	1	1	1	1	8	0	100	4	3	100	100
JBB13B	1	1	1	1	NA	1	1	1	1	8	0	100	4	3	100	100
JBB14A	1	1	1	1	NA	1	1	1	1	8	0	100	4	3	100	100
JBB14B	1	1	1	1	NA	1	1	1	1	8	0	100	4	3	100	100
JBB15A	1	1	1	1	NA	1	1	1	1	8	0	100	4	3	100	100
JBB15B	1	1	1	1	NA	1	1	1	1	8	0	100	4	3	100	100
JBB16A	1		1	1	NA	1	1	1	1	7	0	87.5	4	3	100	100
JBB16B	1	1	1	1	NA	1	1	1	1	8	0	100	4	3	100	100
JBB17A			1	1	NA	1	1	1	1	6	0	75	3	3	75	100
JBB17B		1	1	1	NA	1		1	1	6	0	75	2	3	50	100
JBB18A	1			1	NA		NA		NA	2	0	33.33	2	0	100	0
JBB18B		1		1	NA		NA		NA	2	0	33.33	1	0	50	0
PNA1	NA	NA	NA	1	NA	1	1		1	4	0	80	3	1	100	50
PNA2	NA	NA	NA	1	NA				1	2	0	40	2	0	66.67	0
PNA3	NA	NA	NA		NA				1	1	0	20	1	0	33.33	0
PNA4	NA	NA	NA		NA	2			1	1	1	20	1	0	33.33	0
PNA5	NA	NA	NA		NA				1	1	0	20	1	0	33.33	0
PNA6	NA	NA	NA	1	NA	1			1	3	0	60	2	1	66.67	50
PNB1A	NA	NA	NA	1	NA	1			1	3	0	60	2	1	66.67	50
PNB1B	NA	NA	NA	1	NA	1			1	3	0	60	2	1	66.67	50
PNB2A	NA	NA	NA	1	NA					1	0	20	1	0	33.33	0
PNB2B	NA	NA	NA	1	NA				1	2	0	40	2	0	66.67	0
PNB3A	NA	NA	NA	1	NA	1			1	3	0	60	2	1	66.67	50
PNB3B	NA	NA	NA	1	NA	1			1	3	0	60	2	1	66.67	50
PNB4A	**NA**	**NA**	**NA**		**NA**					**0**	**0**	**0**	**0**	**0**	**0**	**0**
PNB4B	NA	NA	NA	1	NA	1			1	3	0	60	2	1	66.67	50
PNB5A	NA	NA	NA	1	NA	1			1	3	0	60	2	1	66.67	50
PNB5B	NA	NA	NA	1	NA	1			1	3	0	60	2	1	66.67	50
PNB6A	NA	NA	NA	1	NA	1			1	3	0	60	2	1	66.67	50
PNB6B	NA	NA	NA	1	NA	1			1	3	0	60	2	1	66.67	50
PNB7A	NA	NA	NA	1	NA	1			1	3	0	60	2	1	66.67	50
PNB7B	NA	NA	NA	1	NA	1			1	3	0	60	2	1	66.67	50
PNB8A	NA	NA	NA	1	NA	1			1	3	0	60	2	1	66.67	50
PNB8B	NA	NA	NA	1	NA	1	1		1	4	0	80	3	1	100	50
PNB9A	NA	NA	NA	1	NA		1		1	3	0	60	3	0	100	0
PNB9B	NA	NA	NA	1	NA	1	1		1	4	0	80	3	1	100	50

[a] This survey of the preferred environment was run separately to the rest of the survey during 2006. [b] The percentage of non-NA groups for which hypothesis 1 holds. [c] Groups 1 and 2 are denoted (1) and (2) respectively.

Table 5. Mode responses to system preference questions (percentage giving mode response in brackets)

Year	**2006**	**2007**		**2008**	
Group	**1**	**2**	**1**	**2**	**1**
If you had free choice, which development environment would you prefer to have used to learn programming?	Both JBuilder and Programmers Notepad (45.8%)	JBuilder only (42.1%)	Both JBuilder and Programmers Notepad (18.2%)	Both JBuilder and Programmers Notepad (25%)	Both JBuilder and Programmers Notepad (29.6%)
If you had free choice, which development environment would you prefer to use to do programming now?	JBuilder only (66.7%)	JBuilder only (52.6%)	JBuilder only (22.7%)	JBuilder only (12.5%), Both JBuilder and Programmers Notepad (12.5%), Textmate (12.5%)	JBuilder only (37%)

Appendix 2: Group 1 Survey Instrument (2008)

Note: the labeling used in this survey has been modified within the paper to improve readability; for example, B1 corresponds to JBA1 within the paper

SURVEY ON INTEGRATED DEVELOPMENT ENVIRONMENTS

A - ICT 221/521 STUDENT INFORMATION

A1 Which degree programme are you enrolled in? *(Please tick one box only)*

- BICT ❑1
- BBus (Information Systems) ❑2
- Grad Dip (IS) ❑3
- Other (please specify below) ❑4

.....

A2 Are you enrolled in ICT321/621 Architecture & Systems Integration? *(Please tick one box only)*

- Yes, I am enrolled in ICT321/621 this semester ❑1
- No, but I have already taken ICT321/621 ❑2
- No, but I have been given credit for ICT321/621 through prior learning ❑3
- No, but I will take ICT321/621 in the future ❑4
- No, and I will not take ICT321/621 in the future ❑5

A3 Have you taken INF311/611 Advanced Business Programming? *(Please tick one box only)*

- Yes ❑1
- No, but I have already taken ICT311/611 ❑2
- No, and I have not taken ICT311/611 ❑3

A4 How much programming experience do you currently have? *(Please tick one box only)*

- None ❑0
- Less than one year ❑1
- Between one and three years ❑2
- Between four and six years ❑4
- Between seven and ten years ❑5
- More than ten years ❑6

A5 Which Integrated Development Environments (IDEs) have you used? *(Please tick all boxes that apply)*

AnyJ	❑1	NetBeans	❑9
BlueJ	❑2	OptimalJ	❑10
JBuilder	❑3	Oracle JDeveloper	❑11
DrJava	❑4	Programmers Notepad	❑12
Eclipse	❑5	Visual Cafe	❑13
IdeaJ	❑6	Websphere Studio	❑14
JCreator	❑7	A text editor	❑15
JGrasp	❑8	Other (please specify below)	❑16

.....

A6 Which Integrated Development Environment (IDE) do you use the most? *(Please tick one box only)*

AnyJ	❑1	NetBeans	❑9
BlueJ	❑2	OptimalJ	❑10
JBuilder	❑3	Oracle JDeveloper	❑11
DrJava	❑4	Programmers Notepad	❑12
Eclipse	❑5	Visual Cafe	❑13
IdeaJ	❑6	Websphere Studio	❑14
JCreator	❑7	A text editor	❑15
JGrasp	❑8	Other (please specify below)	❑16

.....

A7 Which programming language do you have the most experience in? *(Please tick one box only)*

- Java ❑1
- Other (please specify below) ❑2

.....

B – JBUILDER UTILITY AND EASE OF USE

The following set of questions asks about the usefulness and usability of JBuilder used within ICT221/ICT521. *Please indicate your agreement or disagreement with each statement by ticking the appropriate response on the 1-to-5 point scale:*		1 = Strongly Disagree	2 = Tend to Disagree	3 = Neutral	4 = Tend to Agree	5 = Strongly Agree
B1	Learning how to use JBuilder is straightforward	❑1	❑2	❑3	❑4	❑5
B2	Once you have learned how to use JBuilder then producing Java software with it is straightforward	❑1	❑2	❑3	❑4	❑5
B3	Using JBuilder is enjoyable	❑1	❑2	❑3	❑4	❑5
B4	JBuilder gives me assistance in its use	❑1	❑2	❑3	❑4	❑5
B5	The amount of time and effort required to perform tasks using JBuilder is low	❑1	❑2	❑3	❑4	❑5
B6	I feel I am in control of JBuilder when I use it	❑1	❑2	❑3	❑4	❑5

Please continue on the following page

C – UNDERSTANDING / PRODUCTIVITY MEASURES

The following is a list of aspects of JBuilder used within ICT221/521 with two different sets of responses required.

In the 1st set of responses – (a) Rate how each aspect of JBuilder used in ICT221/521 improves or impairs your understanding of the course concepts: *(please indicate your response on the 1-to-5 point scale of improvement/impairment, or give a response of 0 if you have never used the feature)*

In the 2nd set of responses – (b) Rate how each aspect of JBuilder used in ICT22/521 improves or impairs your productivity in producing software: *(please indicate your response on the 1-to-5 point scale of improvement/impairment, or give a response of 0 if you have never used the feature)*

		Improves/Impairs Understanding (a)						*Improves/Impairs Productivity (b)*					
		0 = Never used this feature	*1 = Strong impairment*	*2 = Moderate impairment*	*3 = No effect*	*4 = Moderate improvement*	*5 = Strong improvement*	*0 = Never used this feature*	*1 = Strong impairment*	*2 = Moderate impairment*	*3 = No effect*	*4 = Moderate improvement*	*5 = Strong improvement*
C1	JBuilder's automatic code formatting (this indents code such as contents of methods or loops)	❑0	❑1	❑2	❑3	❑4	❑5	❑0	❑1	❑2	❑3	❑4	❑5
C2	JBuilder's automatic completion of words in programs (for example, entering "System." presents a list of all items in the System class, and "out" can be selected from this)	❑0	❑1	❑2	❑3	❑4	❑5	❑0	❑1	❑2	❑3	❑4	❑5
C3	JBuilder's display of parameter lists (when a method name is entered, JBuilder shows a list of the parameters that it requires)	❑0	❑1	❑2	❑3	❑4	❑5	❑0	❑1	❑2	❑3	❑4	❑5
C4	JBuilder's automatic creation of program code (this performs tasks such as adding a missing closing curly bracket after enter is pressed)	❑0	❑1	❑2	❑3	❑4	❑5	❑0	❑1	❑2	❑3	❑4	❑5
C5	JBuilder's code creation wizards (these perform tasks such as creating classes or methods)	❑0	❑1	❑2	❑3	❑4	❑5	❑0	❑1	❑2	❑3	❑4	❑5
C6	JBuilder's Refactor…Rename tool (where changing a variable name results in all other copies of that name being changed as well)	❑0	❑1	❑2	❑3	❑4	❑5	❑0	❑1	❑2	❑3	❑4	❑5
C7	JBuilder's integrated help system (this allows help to be viewed for example by highlighting an item and pressing F1 or going to the Help menu)	❑0	❑1	❑2	❑3	❑4	❑5	❑0	❑1	❑2	❑3	❑4	❑5
C8	JBuilder's automatic code colouring (this makes comments green, reserved words red, etc)	❑0	❑1	❑2	❑3	❑4	❑5	❑0	❑1	❑2	❑3	❑4	❑5

Please continue on the following page

Please continue to rate your improvements/impairments in understanding and productivity by ticking the appropriate responses to each statement on the 1-to-5 point scales:		*Improves/Impairs Understanding (a)*						*Improves/Impairs Productivity (b)*					
		0 = Not applicable	*1 = Strong impairment*	*2 = Moderate impairment*	*3 = No effect*	*4 = Moderate improvement*	*5 = Strong improvement*	*0 = Not applicable*	*1 = Strong impairment*	*2 = Moderate impairment*	*3 = No effect*	*4 = Moderate improvement*	*5 = Strong improvement*
C9	JBuilder's display of line numbers within program code	❑0	❑1	❑2	❑3	❑4	❑5	❑0	❑1	❑2	❑3	❑4	❑5
C10	JBuilder's automatic syntax error reporting (this reports errors such as missing semicolons)	❑0	❑1	❑2	❑3	❑4	❑5	❑0	❑1	❑2	❑3	❑4	❑5
C11	JBuilder's code audit warnings (this warns of problems such as variables that are not used)	❑0	❑1	❑2	❑3	❑4	❑5	❑0	❑1	❑2	❑3	❑4	❑5
C12	JBuilder's deprecation warnings (where warnings are given when parts of Java that should not be used any more are included in programs)	❑0	❑1	❑2	❑3	❑4	❑5	❑0	❑1	❑2	❑3	❑4	❑5
C13	JBuilder's support for debugging operations through breakpoints and line-by-line execution	❑0	❑1	❑2	❑3	❑4	❑5	❑0	❑1	❑2	❑3	❑4	❑5
C14	JBuilder's support for variable values to be viewed and modified when in debugging mode	❑0	❑1	❑2	❑3	❑4	❑5	❑0	❑1	❑2	❑3	❑4	❑5
C15	JBuilder's automatic matching of brackets and braces (when a regular or curly bracket is selected, the matching bracket is highlighted)	❑0	❑1	❑2	❑3	❑4	❑5	❑0	❑1	❑2	❑3	❑4	❑5
C16	JBuilder's automatic generation of Javadoc comments (entering '/**' and return creates the comment with tags for parameters etc included)	❑0	❑1	❑2	❑3	❑4	❑5	❑0	❑1	❑2	❑3	❑4	❑5
C17	JBuilder's Javadoc integration (when you are viewing source code you can click on the tab marked 'Javadoc' to see how all the Javadoc comments look when turned into HTML)	❑0	❑1	❑2	❑3	❑4	❑5	❑0	❑1	❑2	❑3	❑4	❑5

Please continue on the following page

D – JBUILDER COMMENTS

D1 *Please describe the best aspect(s) of the JBuilder integrated development environment used within ICT221/521:*

.....

.....

.....

.....

.....

.....

.....

.....

D2 *Please describe the aspect(s) of the JBuilder integrated development environment used within ICT221/521 that are most in need of improvement:*

.....

.....

.....

.....

.....

.....

.....

.....

D3 *Please describe the least useful aspect(s) of the JBuilder integrated development environment used within ICT221/521:*

.....

.....

.....

.....

.....

.....

.....

.....

Please continue on the following page

E – PROGRAMMERS NOTEPAD UTILITY AND EASE OF USE

	The following set of questions asks about the usefulness and usability of Programmers Notepad used within ICT221/ICT521. *Please indicate your agreement or disagreement with each statement by ticking the appropriate response on the 1-to-5 point scale:*	*1= Strongly Disagree*	*2= Tend to Disagree*	*3= Neutral*	*4 = Tend to Agree*	*5 = Strongly Agree*
E1	Learning how to use Programmers Notepad is straightforward	❑1	❑2	❑3	❑4	❑5
E2	Once you have learned how to use Programmers Notepad then producing Java software with it is straightforward	❑1	❑2	❑3	❑4	❑5
E3	Using Programmers Notepad is enjoyable	❑1	❑2	❑3	❑4	❑5
E4	Programmers Notepad gives me assistance in its use	❑1	❑2	❑3	❑4	❑5
E5	The amount of time and effort required to perform tasks using Programmers Notepad is low	❑1	❑2	❑3	❑4	❑5
E6	I feel I am in control of Programmers Notepad when I use it	❑1	❑2	❑3	❑4	❑5

Please continue on the following page

F – UNDERSTANDING / PRODUCTIVITY MEASURES

The following is a list of aspects of Programmers Notepad used within ICT221/521 with two different sets of responses required.

In the 1st set of responses – (a) Rate how each aspect of Programmers Notepad used in ICT221/521 improves or impairs your understanding of the course concepts: *(please indicate your response on the 1-to-5 point scale of improvement/impairment, or give a response of 0 if you have never used the feature)*

In the 2nd set of responses – (b) Rate how each aspect of Programmers Notepad used in ICT22/521 improves or impairs your productivity in producing software: *(please indicate your response on the 1-to-5 point scale of improvement/impairment, or give a response of 0 if you have never used the feature)*

		Improves/Impairs Understanding (a)						*Improves/Impairs Productivity (b)*					
		0 = Never used this feature	*1 = Strong impairment*	*2 = Moderate impairment*	*3 = No effect*	*4 = Moderate improvement*	*5 = Strong improvement*	*0 = Never used this feature*	*1 = Strong impairment*	*2 = Moderate impairment*	*3 = No effect*	*4 = Moderate improvement*	*5 = Strong improvement*
F1	Programmers Notepad's code indenting (when lines of code are selected, pressing tab moves them all one tab space to the right)	❑0	❑1	❑2	❑3	❑4	❑5	❑0	❑1	❑2	❑3	❑4	❑5
F2	Programmers Notepad's bookmarks (bookmarks can be defined on selected lines of program code; these can then be navigated to)	❑0	❑1	❑2	❑3	❑4	❑5	❑0	❑1	❑2	❑3	❑4	❑5
F3	Programmers Notepad's code folding (each block of code can be 'folded' into a single line and then later unfolded to its original state)	❑0	❑1	❑2	❑3	❑4	❑5	❑0	❑1	❑2	❑3	❑4	❑5
F4	Programmers Notepad's case conversion (allowing selected code to be made upper or lower case)	❑0	❑1	❑2	❑3	❑4	❑5	❑0	❑1	❑2	❑3	❑4	❑5
F5	Programmers Notepad's automatic matching of brackets and braces (when a regular or curly bracket is selected, the matching bracket is highlighted)	❑0	❑1	❑2	❑3	❑4	❑5	❑0	❑1	❑2	❑3	❑4	❑5
F6	Programmers Notepad's search and replace	❑0	❑1	❑2	❑3	❑4	❑5	❑0	❑1	❑2	❑3	❑4	❑5
F7	Programmers Notepad's compiler integration (allowing the Java compiler to be accessed within Programmers Notepad)	❑0	❑1	❑2	❑3	❑4	❑5	❑0	❑1	❑2	❑3	❑4	❑5
F8	Programmers Notepad's automatic code colouring (this makes comments green, reserved words blue, etc)	❑0	❑1	❑2	❑3	❑4	❑5	❑0	❑1	❑2	❑3	❑4	❑5
F9	Programmers Notepad's display of line numbers within program code	❑0	❑1	❑2	❑3	❑4	❑5	❑0	❑1	❑2	❑3	❑4	❑5

Please continue on the following page

G – PROGRAMMERS NOTEPAD COMMENTS

G1 *Please describe the best aspect(s) of Programmers Notepad used within ICT221/521:*

.....

.....

.....

.....

.....

G2 *Please describe the aspect(s) of Programmers Notepad used within ICT221/521 that are most in need of improvement:*

.....

.....

.....

.....

.....

G3 *Please describe the least useful aspect(s) of Programmers Notepad used within ICT221/521:*

.....

.....

.....

.....

.....

H– JBUILDER AND PROGRAMMERS NOTEPAD

H1 If you had free choice, which development environment would you prefer to have used to learn programming? *(Please tick one box only)*

JBuilder only ❑1
Programmers Notepad only ❑2
Both JBuilder and Programmers Notepad ❑3
Other (please specify below) ❑4
.....

H2 If you had free choice, which development environment would you prefer to use to do programming now? *(Please tick one box only)*

JBuilder only ❑1
Programmers Notepad only ❑2
Both JBuilder and Programmers Notepad ❑3
Other (please specify below) ❑4
.....

Thankyou for completing this survey

Systems Analysis and Design: Know your Audience

Bryan A. Reinicke
reinickeb@uncw.edu
Information Systems and Operations Management
University of North Carolina Wilmington
Wilmington, North Carolina 28403, USA

Abstract

Systems analysis and design (SAD) classes are required in both Information Systems and Accounting programs, but these audiences have very different needs for these skills. This article will review the requirements for SAD within each of these disciplines and compare and contrast the different requirements for teaching systems analysis and design to both audiences. These observations are based on both literature on the subject, and the authors personal experience with teaching SAD to these two audiences.

Keywords: Systems Analysis and Design, Curriculum, Education

1. INTRODUCTION

Based on the IS model curriculum, systems analysis and design is a core course in the Information Systems curriculum (Topi, Valacich, Wright, Kaiser, Nunamaker, Sipior & de Vreede, 2010). SAD is also a required course in many Accounting programs, particularly for Accounting Information Systems or Audit concentrations (Badua, 2008; Daigle, Hayes & Hughes, 2007). While these courses could nominally be the same, and may be taught from the same textbook, there are distinct differences in the needs of these two audiences on the subject of SAD. Making this more difficult, SAD is frequently hard to convey as a subject to information systems students (Clyde & Crane, 2003; Chen, 2006), and attempting it with two different audiences compounds this problem.

The observations and comparisons in this paper are based both on the authors' experience with teaching SAD in both curriculums as well as research into the area. It is the intention of this paper to assist other faculty in avoiding some of the problems encountered by the author when teaching what is nominally the same material to different audiences.

The paper is structured as follows: First, the presentation of SAD concepts in the IS curriculum is examined. Then, the same is done for SAD in the accounting curriculum. This is followed by a discussion of the commonalities between the two curriculums. Next a discussion of the differences and potential problems created by these differences is examined. Finally, some concluding thoughts are presented.

2. SYSTEMS ANALYSIS AND DESIGN IN INFORMATION SYSTEMS

Systems Analysis and design courses are required for Information systems majors, based on the model IS curriculum (Topi, et al., 2010). The IS model curriculum notes that the SAD course "…discusses the processes, methods, techniques and tolls that organizations use to determine how they should conduct their business, with a particular focus on how computer-based technologies can most effectively contribute to the way business is organized" (p 51, Topi, et al., 2010).

The 2010 IS model curriculum lists 13 specific learning objectives for SAD courses within the undergraduate IS curriculum (p51, Topi, et al., 2010). There is a great deal of latitude given to schools on how to meet these learning objectives to allow flexibility on how the goals are met and which tools are used in classes.

The guidelines do note that the SAD course should focus on the process of analyzing and documenting business processes and then converting these into systems requirements and design specifications. The methods and approaches used are left up to the individual institutions, but the guidelines state that it is important for students to be exposed not only to the Systems Development Life Cycle (SDLC), but to Object Oriented (OO) design using the unified process and Unified Modeling Language (UML) and to agile development methodologies as well.

The core SAD course is recommended for Application developers, business process analysis, project managers, User interface designers and web content managers (Topi, Valacich, Kaiser, Nunamaker, Sipior, deVreede & Wright, 2007). These jobs cover a wide range of professional areas that IS students may find themselves working in, particularly immediately after graduation.

The key concerns for IS students in this course is to master the skills required for them to become competent in the requisite skills to prepare them for the jobs listed above and, of course, to pass the course so that they can graduate.

3. SYSTEMS ANALYSIS AND DESIGN FOR ACCOUNTING

SAD courses for accounting have a slightly different set of standards. First, it is not a required course in all undergraduate Accounting curriculums. Rather, it is addressed in Accounting Information Systems programs (Badua, 2008) or in Masters programs (Masters of Science in Accounting or MSA), which many students take to meet the requirements of the Certified Public Accountant (CPA) exam for education (http://www.aicpa.org/BecomeACPA/Licensure/Requirements/Pages/default.aspx). In addition, MSA students are not being trained as developers, nor will they necessarily have any development training or experience. Instead, these students are generally training for careers in auditing and control. Because of this, the focus of the course will be slightly different for these students. However, there are a number of studies that have pointed to the importance of increasing the IS skills of accounting majors (Daigle et al., 2007).

Because of these differences, the key concerns of accounting students in SAD courses differ from those of the IS students in similar courses.

Pass the CPA exam

The primary concern for most accounting students, and virtually all MSA students, is to pass the CPA exam. This is not an easy task, and the focus of the CPA exam does not help with the course content for SAD.

There is very little on the CPA exam that would cover the concepts in a SAD course (Gleim, 2009). Accountants, after all, are not developers, but they are likely to act as business analysts and, of course, as systems auditors.

Some of the commonalities and differences caused by the differences in expectations between the two programs are discussed in the following sections.

4. COMMONALITIES

Clearly, despite the differences between the IS and Accounting majors, there are a number of similarities in the requirements between these two. There is also an overlap in the types of jobs that the students could be looking into, as MSA students who have followed a system/audit style track could very easily find themselves in the role of a business process analyst or systems consultant.

The discussion of the commonalities is structured based on the learning objectives from the IS model curriculum (Topi, et al., 2010), the American Institute of CPAs (AICPA) core competencies (Daigle et al., 2007) and the authors' observations having taught courses in both curriculums. Even though these areas are of common concern, there may be differences in the way they need to be addressed to the different student groups. Those differences are addressed in the next section of the paper.

The first area of common concern between the curriculums is understanding the needs of the business and how these might be addressed by information systems. This is a skill required by

business analysts, and these are positions that could be filled by students from either area. In fact, the ability to leverage technology is listed as one of the AICPA core competencies for accounting students (Daigle et al., 2007).

The second common area is the process of initiating, prioritizing and assessing the feasibility of information systems projects. Each group of students would bring different strengths to this process based on their training, but it is an area that is focused on in both curriculums.

The third common area is utilizing a methodology for analyzing a business problem and modeling it using a given technique. While this is very open in the IS model curriculum to give schools flexibility on which methodologies and techniques are used, there is significantly less flexibility on the accounting side. This is largely driven by the fact that the accounting students need to be concerned with both what is expected of them on the CPA exam and what is expected within the accounting profession. This is discussed in more detail in the next section.

The forth area that both disciplines are concerned with is project management. This has actually been an area of expanding concern within the IS profession for a number of years, and it is certainly one within the accounting profession for at least one of the same reasons: the cost of IS projects.

The fifth area of overlap is the examination of articulation of various systems alternatives to solve a given business problem. This could include assessing whether to use a packaged or custom solution for a given system. Again, students in each area bring different strengths to this area based on their training.

Related to the previous area, the sixth are is the comparison of acquisition alternatives. This would involve creating an assessment metric and the applying that metric to the various alternatives solutions that the company has selected for that problem.

The seventh area, based on the IS model curriculum, deals with system security. This is certainly a primary concern for system auditors (Walters, 2007), and is something that is emphasized at multiple points in an accounting curriculum in the form of audit controls, which are then coded into the system.

The final area of overlap is that of analyzing and articulated ethical, cultural and legal issues for the system and how these impact the feasibility of the system. Ethical behavior and the regulations surrounding financial reporting are two areas that are focused on in the CPA exam, and therefore in accounting curriculums. With the advent of legislation such as Sarbanes Oxley, these concerns are quite directly translated into systems concerns.

By reviewing this list, it can be seen that there is at least partial overlap for 8 of the 13 learning objectives for SAD between IS and accounting. While this is fairly extensive, it's also significantly less than 100%, which can lead to some issues between the disciplines, and certainly leads to a different focus when teaching these classes.

5. DIFFERENCES AND PROBLEMS

While there are a large number of common areas of learning within the two curriculums, there are a number of differences as well. This is where the potential disconnects, and potentially some problems exist. However, it is not just the disconnects that can cause problems. It is also the differences in the overlap that can create problems as well.

In this section, I explore some of the areas that are most likely to cause problems. The purpose of this discussion is to highlight those areas where disconnects can occur, and help instructors working with either group (or both groups) of students identify the topics in their curriculum that may need to be adjusted.

Financials and the importance thereof

Certainly, financial considerations for new systems are covered in IS courses on SAD, but this is frequently not given extensive consideration. After all, this topic is a subset of one of the 13 primary learning objectives for the course, so it is difficult for many instructors to spend an extensive amount of class time on it.

While this may not be a primary concern for many IT professionals and professors, it is the primary concern of accountants and auditors. These are students who have spent and extensive amount of class time on considerations of cost and cash flow. This could lead to a disconnect between accounting and IS students, and will certainly change the amount of time spent on a topic in class.

Scheduling, and the problems associated with same
Project management is an important topic to cover, at least in part, within an SAD course. One of the problems with planning for systems projects is the inherent uncertainty that can surround development time for a new project. This is particularly true if the technology being utilized is relatively new, or if the problem being addressed is one that the organization does not have extensive familiarity with.

Generally speaking, IS students can grasp this problem very quickly. They have all had to take programming courses, and they have all had a program take longer to code than they thought it would. The same cannot be said for accounting students, who are not trained as programmers. They have generally not had the experience of an "easy" programming problem occupying an entire weekend.

This can be somewhat addressed depending on the type of database course the students have had. MSA students generally have a DB course as a part of the curriculum, as everything they need to verify as an auditor is in a database somewhere, and if they have had to program in SQL, they can understand the difficulties of coding. If not, then it is an area that will need some additional attention in the class.

Differences in approaches to identification and roll out of new technology
Clearly, one of the functions of a systems analyst or IT consultant would be to identify new technologies that could be applied to the business. It should be expected that IS majors would have higher levels of technical skills, and likely a more technical bent, than accounting majors. It could also be assumed that the average accounting student will be more conservative when it comes to the application of technology, particularly new technology, than the average IS major. IS students do have a tendency to be enthusiastic about the use of technology, while accounting students are trained to be more focused on issues of cost and functionality. Thus, while both could be responsible for the identification of new technologies to apply to the business, it is entirely likely that they will have divergent views on which technologies are suitable for implementation.

This means that, when discussing this type of activity in class, the instructor may need to take a different tack with both groups of students.

OO vs. Business Process Diagrams
As more and more IS shops and programs move to OO design and build techniques (Satzinger, Batra & Topi, 2007), there is likely to be a larger disconnect, as the accounting programs do not tend to focus on these (Jones & Lancaster, 2001). Part of the reason is that the questions on this do not appear on the CPA exam, which tends to focus on much older technology. As an example of this, my MSA students have told me that practice questions on the CPA exam in the technology area include "What is the job title of the person who feeds the punch cards into the computer?", and there is still discussion in the CPA review books of the role of the Librarian in checking out code to developers (Gleim, 2009).

Business process design and documentation is one of the auditors primary focuses, which makes sense as it is their responsibility to audit these processes to ensure compliance with applicable regulations. While this is also an area which IS classes focus, this is an area that may get more attention in an MSA class simply because they will not be working on the development tasks that may be covered in an IS class. The disconnect lies with the fact that accounting classes generally do not focus on object oriented techniques, which capture business processes in a different way. They tend to focus on the "older" business process diagrams, rather than newer OO techniques.

Why do you care about the development environment?
Auditors can have a legitimate set of concerns regarding a development environment from a control perspective (Hall, 2011). From an audit standpoint, there is very little that is less desirable than people being able to make unrestricted changes to a system without a control in place. From an IS standpoint, this means that our students should be prepared for these types of questions from the auditors and understand that they have a legitimate interest. This would include potentially auditing which developers have access to which areas of the system.

This also means that, in an accounting class, this topic will need to be addressed. As noted in the previous section, the CPA exam has not exactly kept up with new developments in technology.

This means that the accounting students will need to be educated about what can be done to control code in a modern systems development environment. The same could be said for the information systems students, who would also need to understand the differences between a build and test and production environment, and why the two should be separate.

Why do you care about my systems documentation?

Auditors may be required to audit the documentation vs. the code in a production system (Hall, 2011). This means that every change in the production system must be logged and, more importantly, must match the changes that are actually in the documentation. This is a legitimate audit function, and one that could come up in a systems development project, particularly in the maintenance phase. This alone means that audit standards may not line up with some systems development methodologies that do not emphasize documentation of the system (i.e. Agile methods).

With regulations like Sarbanes-Oxley, it seems likely that this type of audit is likely to continue in the future. This emphasizes the importance of documentation practices for the information systems student's, but it means that we must also educate the accounting students about the types of documentation and how these are created. There is the possibility that accounting students would reject agile methods as a viable option out of hand because of the reduced documentation that can accompany such development techniques. It needs to be made clear that even using agile methods, it is possible to create complete systems documentation.

6. CONCLUSIONS

While both information systems and accounting programs have a need to teach systems analysis and design courses, the needs of the students in each of these classes can be distinctly different. It is certainly possible to teach to both of these groups, but it is best to do so using two different curriculums because of the differences. This paper has laid out some of the similarities and differences between these two groups as a reference point for faculty who need to teach the same course to these different audiences.

7. REFERENCES

Badua, F. (2008). Pedagogy and the PC: Trends in the AIS Curriculum. Journal of Education for Business, 83(5), 7.

Chen, B. (2006). Teaching Systems Analysis and Design: Bringing the Real World into the Classroom. Information Systems Education Journal, 4(84).

Clyde, S. W., & Crane, A. E. (2003). "Design-n-Code Fests" as Capstone Projects for an Object-Oriented Software Development Course. Computer Science Education, 13(4), 16.

Daigle, R. J., Hayes, D. C., & Hughes, K. E. I. (2007). Assessing Student Learning Outcomes in the Introductory Accounting Information Systems Course Using AICPA's Core Competency Framework. Journal of Information Systems, 21(1), 22.

Hall, J. A. (2011). Information Technology Auditing (3 ed.). Mason, OH: Cengage.

Jones, R. A., & Lancaster, K. A. S. (2001). Process Mapping and Scripting in the Accounting Information Systems (AIS) curriculum. Accounting Education, 10(3), 17.

Gleim, I. N. (2009). CPA Review: Business Environment and Concepts (2010 ed.). Gainesville, FL: Gleim Publishing, Inc.

Satzinger, J. W., Batra, D., & Topi, H. (2007). Analysis and Design in the IS Curriculum: Taking it to the Next Level. Communications of the Association for Information Systems, 20, 15.

Topi, H., Valacich, J. S., Kaiser, K. M., Nunamaker, J. F. J., Sipior, J. C., de Vreede, G. J., Wright, R. (2007). Revising the IS Model Curriculum: Rethinking the Approach and Process. Communications of the Association for Information Systems, 20, 14.

Topi, H., Valacich, J. S., Wright, R. T., Kaiser, K. M., Nunamaker, J. F. J., Sipior, J. C., de Vreede (2010). IS 2010: Curriculum Guidelines for Undergraduate Degree Programs in Information Systems. ACM / AIS.

Walters, L. M. (2007). A Draft of an Information Systems Security and Control Course. Journal of Information Systems, 21(1), 27.

Measuring Assurance of Learning Goals: Effectiveness of Computer Training and Assessment Tools

Marianne C. Murphy
mmurphy@nccu.edu

Aditya Sharma
asharma@nccu.edu

Mark Rosso
mrosso@nccu.edu

Computer Information Systems,
North Carolina Central University
Durham, NC, 27707, USA

ABSTRACT

Teaching office applications such as word processing, spreadsheet and presentation skills has been widely debated regarding its necessity, extent and delivery method. Training and Assessment applications such as MyITLab, SAM, etc. are popular tools for training students and are particularly useful in measuring Assurance of Learning (AOL) objectives. Meeting these assessment objectives has become a crucial issue in business schools as it now plays a major role in AACSB accreditation. It is our contention that these tools are fundamentally necessary to train and assess students to meet specific objectives that support a particular goal. In our experience, the simulation component of these tools is not enough to ensure all objectives. In this paper, we describe our experience with the use of in-the-application assignment projects to supplement the assessment and training simulation in order to improve final assessments and close the AOL loop.

Keywords: assessment, computer applications

1. INTRODUCTION

Teaching office applications such as word processing, spreadsheet and presentation skills has been widely debated regarding necessity, extent and delivery method. Some contend that entering freshman should have had exposure to these applications and require the passing of an assessment exam (Shannon, 2008). Others believe that high school exposure does not ensure necessary advanced skills in applications such as spreadsheets and require additional training (Hulick & Valentine, 2008). Traditional training in computer applications has generally included lecture and lab assignments in the particular application (Mykytyn, Pearson, Paul, & Mykytyn, 2008). In more recent years, many universities have turned to assessment and training tools such as MyITLab, SAM, SimNet and SNAP (Hill, 2011; Morris, 2010).

These tools require students to complete various tasks in a simulated application. The tools are also debated, as some wonder if students are really just learning to "click and point" to learn specific tasks but do not have the ability to actually apply these learned tasks to solve business problems (Coleman, Thrasher, & Atkinson, 2010).

However, many universities not only use these simulation tools for training but also for implementing assurance of learning (AOL) standards mandated by the AACSB. Meeting these standards has become crucial, as they now play a major role in the AACSB accreditation of business schools (AACSB, 2007). Program learning goals must be set, objectives must be measured across time, and the results used for continuous improvement (a.k.a "closing the loop" (Al-Mubaid, Abeysekera, Kim, Perkins-Hall, & Yue, 2011; Hollister & Koppel, 2006)).

In this paper, we examine the extent to which these tools can be useful in attaining AOL objectives with regard to computer application skills. After providing a brief overview of the debate over teaching computer application skills, we look at how schools have responded with the use of automated training and assessment tools. We then relate our own school's experience with teaching computer applications, the use of these automated tools and how we supplemented their use in implementing the continuous improvement process necessary for our school's maintenance of AACSB accreditation.

2. BACKGROUND

Office Applications

One assessment goal in many business schools is that students have the ability to use technology (Hollister & Koppel, 2007). Computer application skills in word processing, spreadsheets and presentations are vital for all business students as they matriculate and in future employment (Wolk, 2008).

The need for business schools to teach these skills and/or assess a student's skill level has been a subject of discussion in many schools. One question usually discussed is "shouldn't incoming freshman have these skills?" The answer is that some do but many don't. Research indicates that a large percentage of students are not able to successfully pass a beginning assessment (Hulick & Valentine, 2008; Shannon, 2008; Kline & Strickland, 2004), even in states where competency in technology is required for high school graduation (Grant, Malloy, & Murphy, 2009). This research also shows that students may overestimate their ability in office productivity tools. Students have a much higher perception of their level of skill in these applications than their actual performance on assessments (Grant, et al., 2009). Their study particularly indicated that students did not possess an adequate set of spreadsheet skills (as did (Kline & Strickland, 2004)). Thus, without curricular intervention of some sort, many students will not take a computer applications course and therefore continue to lack critical skills such as spreadsheets.

Assessment Tools

In order to ascertain that students obtain or have these computer application skills, universities have turned to training and assessment tools for test-out and instruction (Morris, 2010). Assessment and training tools have become quite popular in business programs to ensure that students have adequate skills in office production software, and to assess skill level and determine placement (Coleman, Thrasher, & Atkinson, 2010; Tesch, Murphy, & Crable, 2006). Currently, the most popular tools include MyITLab, SAM, SimNet and SNAP (Hill, 2011).

These tools offer many benefits:

- Individualized instruction – students can work on modules that focus on skills in which they are deficient (Morris, 2010).
- Consistent content across sections in multi-section courses – this also encourages consistency of results across sections (Kline & Strickland, 2004).
- Automated grading is quick, and speeds the gathering of assessment data (Merhout, Benamati, Rajkumar, Anderson, & Marado, 2008).
- Distance learning - automated tools can be incorporated in online courses relatively easily (Huan, Shehane, & Ali, 2011).

However, as mentioned earlier, some question the effectiveness of these tools, and what few results have been reported have been mixed (Morris, 2010; Coleman, Thrasher, & Atkinson, 2010; Paranto, Neumann, & Zhang, 2008; Kline & Strickland, 2004).

Assurance of Learning

The importance of assessment in business schools has increased significantly since 2003

when the AACSB adopted new standards for accreditation and reaccreditation. Prior to 2003, the AACSB had only 10% of the criteria related to assessment. Currently, one third of the standards are assessment-related (Pringle & Michel, 2007).

Assessment has played such an important role in accreditation because stakeholders in universities such as state legislators, taxpayers, parents, donors and the federal government are requiring direct evidence of student learning (Bollag, 2006; Suskie, 2004). Computer application simulation tools can be used to easily measure relevant AOL objectives. The model in Figure 1 shows the loop that is referred to by the phrase "closing the loop", with regard to assurance of learning. Simulation tools can fill the assessment role depicted in Al-Mubaid, et al.'s (2011) model. See their paper for a complete description of the assessment process.

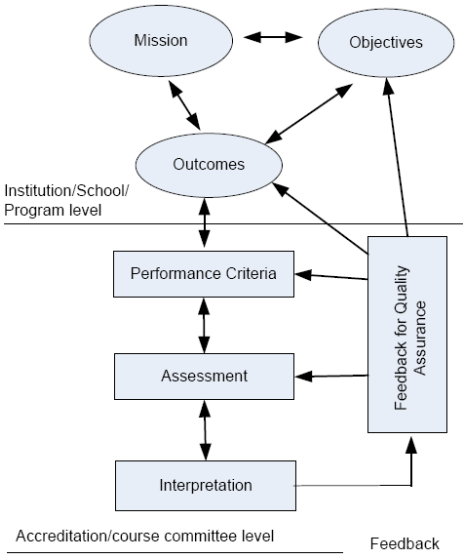


Figure 1. Conceptual Model of Assessment (reprinted from Al-Mubaid, et al., 2011)

3. ASSESSMENT STRATEGY

Teaching Business Applications

At our university, most faculty members in the school of business agree that a sound curriculum include a student's mastery of fundamental computer applications such as word, presentation, spreadsheet and database. However, delivery of training for these tools continues to be widely debated.

We informally surveyed nine of the largest schools by enrollment in the North Carolina state university system. Results show that schools address this delivery issue in a variety of ways:

- Require all students to take a course.
- Pass an assessment or take a course.
- Incorporate computer application skills with a Management of Information Systems (MIS) course.
- Pass an assessment initially or use a self-study application tool until passing the assessment.

Additionally, the course and or courses have a variety of content including:

- One course or assessment that includes word processing, spreadsheet and presentation applications (sometimes with an office database application such as Microsoft Access).
- Separate courses for word processing/presentation and spreadsheet/database.
- An MIS course that includes spreadsheets only.
- An MIS course that includes spreadsheets and database applications.

Although this data is limited in scope and size, it can reasonably be assumed that other business schools debate the best way to ascertain the delivery of application skills. Over the past several years our university has used a variety of delivery modes. In Fall 2008 and Spring 2009, we offered one business computer applications course that included word processing, spreadsheets and presentation skills and a separate course that includes office database applications.

At this writing, our first business computer applications course includes only spreadsheet skills. The decision to not teach word processing and presentation is largely based on student's requirement to have these skills in other courses and their ability to learn these skills on their own. Additionally, incoming freshman do not have the ability to complete even basic spreadsheet tasks (Grant, et al., 2009; Kline & Strickland, 2004) and these skills are deemed vital for matriculation and post-graduation employment.

Students can test out of the first business computer applications course (Microsoft Excel). The database application course is an elective. Our teaching and assessment tool is Pearson's MyITLab. All business students are required to obtain a score of 70% or better on an

assessment or take the business applications course. This percentage is based on the business school policy of requiring students to matriculate with a C or better in all their coursework. All students who take the course are required to take the same assessment as a post-test.

In the Fall of 2008 and Spring of 2009 this assessment included the testing of 10 MS-Excel skills, 5 MS-Word skills and 5 MS-PowerPoint skills. In the Fall of 2009 the pre- and post-test assessed 20 MS-Excel skills. Seven MS-Excel skills are persistent during the entire test period of Fall 2008 to Spring 2011. Each of these tested skills includes 2-5 tasks. All of the tasks for each skill must be completed successfully.

AOL Goal

We use this pre- and post-test of all business students to measure our technology AOL goal. The criterion for meeting this goal is that 70% of the students correctly complete each skill tested. A summary of our AOL report is included in Appendix A.

Each semester's post-tests are reviewed and a strategy to reach our goal of 70% on all skills is determined. In spite of several strategies, our post-test results in Fall 2008 through Spring 2009 indicated that, on average, half of the original 10 tested skills were below standard.

Project Implementation

A criticism of the assessment and application tools is that students only learn to click and point in a simulated environment and these skills do not always translate to "in-the-application" skills. Project-based courses in business applications may be more successful (Murray, Hooper, & Perez, 2007) but are not always practical in terms of training large numbers of students.

After the initial introduction of the MyITLab tool, Pearson Education received numerous requests for a built-in grader for problem solving projects that could be performed in the actual application. In Fall 2009, MyITLab offered an applications enhancement called Project Grader. This enhancement offered in-the-application projects. Students would download a beginning spreadsheet and perform a variety of tasks in MS-Excel, upload the completed spreadsheet and receive a grade based on the correct completion of those tasks.

In order to determine what effect the projects would have on the overall performance of students on the final assessment, we implemented projects in one section over two consecutive semesters (Fall 2009, Spring 2010). Projects were implemented in all sections in the Fall 2010 semester. Instructors determined how many projects to include in their section. In Spring 2011, all sections included 7 projects in addition to the simulation training. See Appendix B for the results of these sections.

4. WHAT WE LEARNED

Teaching and/or assessing students in computer applications skills and measuring our AOL objectives remain an ongoing process. However, our experience has taught us that:

- Incoming freshmen do not always have the necessary computer application skills, in particular spreadsheet skills.
- Training and Assessment simulation tools have proven to be an effective method for training students and measuring AOL objectives.
- Augmenting simulation training with projects that require the use of the actual spreadsheet application improves AOL measured objectives.

The average compliance improved each semester except one. However the most dramatic increases in the percentage of correct tasks were in the one section using 7 projects in Fall 2009 (see Table 2). When 3 projects were used in one section, improvement was noted in some skills but not in others. When instructors determined how many projects to implement in Fall 2010, results were mixed. In Spring 2011, all sections implemented the 7 projects used in Fall 2009 and all persistent skills (of the original 10) tested met the standard. Although our data is not scientific proof that adding projects, in particularly these 7 projects, increases a student's overall skill level, it gives us a base for improvement. Additionally, we show continued improvement over time.

Simulation tools are extremely useful especially in assessing and training computer applications to large numbers of students. Additionally, students' acceptance of this type of training is high (Baker, 2004). However, simulation training may not completely prepare students to successfully apply the skills learned to later tasks and projects using computer application skills. Project-based training in-the-application only is not practical in terms of time and

resources for large numbers of students. In our experience, a combination of simulation training and in-the-application training increases the likelihood that students will be able to complete any given task in that application.

5. MOVING FORWARD

Our experience supports previous research that project-based courses in computer application increases the skill level of the students. Specifically, the addition of application-based projects in our courses increased the percentage of students who could successfully complete the tasks tested and closed the loop for our technology AOL goal. Meeting AOL goals for AACSB accreditation is vital for business schools. Evaluating our assessment goals every semester and supplementing simulation training with live application projects significantly increased our ability to "close the loop."

We plan to continue using these projects and measure the student's success with the additional 13 skills in the pre- and post-tests (please contact author for a list of these skills). Additionally, based on the pre- and post-tests, we will adjust the project focus to tasks that specifically address the desired skill.

6. REFERENCES

AACSB. (2007, November 20). *AACSB Assurance of Learning Standards: An Interpretation.* Retrieved June 25, 2011, from The Association to Advance Collegiate Schools of Business: http://www.aacsb.edu/accreditation/papers/aolpaper-final-11-20-07.pdf

Al-Mubaid, H., Abeysekera, K., Kim, D., Perkins-Hall, S., & Yue, K. (2011). A Model for Long Term Assessment of Computing and Information Systems Programs. Information Systems Education Journal, 9(3), 59-67.

Atkinson, J. K., Thrasher, E., & Coleman, P. (2010). Simulation software's effect on college students spreadsheet project scores. Academic and Business Research Institute Conference. Orlando.

Baker, J. (2004). Spreadsheet Applications: Prototyping an Innovative Blended Course. Turkish Online Journal of Distance Education, 5(1), 1-9.

Bollag, B. (2006, December 6). Fears of Possible Federal Learning Standards Grow as Liberal-Arts Accreditor Is Penalized. Retrieved 7 6, 2011, from The Chronicle of Higher Education: http://chronicle.com/article/Fears-of-Possible-Federal/119555/

Coleman, P. D., Thrasher, E. H., & Atkinson, J. K. (2010). Simulation software's effect on college students' spreadsheet project scores. Academic and Business Research Institure Conference. Orlando: AABRI.

Grant, D., Malloy, A., & Murphy, M. (2009). A Comparison of Students Perception of their Computer Skills to their Actual Abilities. Journal of Information Technology Education, 8, 141-160.

Hill, T. (2011). Word Grader and Powerpoint Grader. ACM InRoads, 2(2), 34-36.

Hollister, K., & Koppel, N. (2006). Framework for meeting AACSB International's assurance of learning requirements: application to information technology. Journal of Informatics Education Research, 8(3), 1-14.

Hollister, K., & Koppel, N. (2007). Curricular Changes in Response to Assurance of Learning Results in Information Technology. IABE-2007 Annual Conference, III, pp. 152-158. Las Vegas.

Huan, T., Shehane, R., & Ali, a. (2011). Teaching computer science courses in distance learning. Journal of Instructional Pedagogies, 6, 1-14.

Hulick, F., & Valentine, D. (2008). Computer competency of incoming college students: yet more bad news. ISECON (pp. 1-7). Phoenix: EDSIG.

Kline, D., & Strickland, T. (2004). Skill Level Assessment and Multi-section Standarization for an Introductory Microcomputer Applications Course. Issues in Information Systems, 5(2), 572-578.

Merhout, J., Benamati, J., Rajkumar, T.,Anderson, P., & Marado, D. (2008). Implementing direct and indirect assessment in the MIS curriculum. Communications of the Association of Information Systems, 23(24), 419-436.

Morris, K. (2010). College and the digital generation: Assessing and training students for the technological demands of college by exploring relationships between computer self-efficacy and computer proficiency. The

University of Alabama: Unpublished Doctoral dissertation.

Murray, M., Hooper, J., & Perez, J. (2007). A project-based approach to assessing student competency with productivity software. SIGED-IAIM (pp. 1-14). Montreal, Quebec, Canada: AIS.

Mykytyn, J., Pearson, A., Paul, S., & Mykytyn, P. (2008). The use of problem-based learning to enhance MIS education. Decision Sciences Journal of Innovative Education, 6(1), 89-113.

Paranto, S., Neumann, H., & Zhang, L. (2008). Information systems: performing application-specific assessment of student performance. Issues in Information Systems, 9(1), 87-94.

Pringle, C., & Michel, M. (2007). Assessment Practices in AACSB-Accredited Business Schools. Journal of Education for Business, 82(4), 202-211.

Shannon, L. (2008). Information and Communication Technology Literacy Issues in Higher Education. Information Systems Education Journal, 6(23), 3-13.

Suskie, L. (2004). Assessing Student Learning: a common sense guide. Bolton, MA: JB-Anker.

Tesch, D., Murphy, M., & Crable, E. (2006). Implementation of a Basic Computer Skills Assessment Mechanism for Incoming Freshman. Information Systems Education Journal, 4(13), 1-11.

Wolk, R. (2008). How important is student computing ability? The role of information technology competence in business school accreditation. Information Systems Education Journal, 6(39), 1-16.

APPENDIX – A - AOL Summary Results

Date	# of students enrolled	# of students assessed	Below standard	Post Assessment Action (to close the loop)
Fall 2008	259			
Pre-test		201	90%	Current Term: 10 trainings, 4 exams.
Post-test		206	50%	Next term: demonstrate skills in class at least 3 times in areas where average is less than 70%.
Spring 2009	199			
Pre-test		110	90%	Current Term: 10 trainings, 4 exams.
Post-test		128	40%	Next term plan to demonstrate skills in class at least 3 times in areas where average is less than 70%. Introduce in-the-application projects in one section as a test.
Fall 2009	302			
Pre-test		142	80%	Current Term: 10 trainings, 4 exams, 7 projects in one section; 10 trainings, 4 exams in remaining sections.
Post-test		169	30%	Continue to roll out in-the-application projects.
Spring 2010	248			
Pre-test		158	90%	Current Term: 10 trainings, 4 exams, 3 projects in one section; 10 trainings, 4 exams in remaining sections.
Post-test		175	20%	Continue to roll out in-the-application projects.
Fall 2010	139*			
Pre-test		51	100%	Current Term: 10 trainings, 4 exams, 3 projects in one section; 10 trainings, 4 exams in remaining sections.
Post-test		111	10%	Projects in all sections. For comparison on this report only original 7 objectives are included.
Spring 2011	122*			
Pre-test		80	100%	Current Term: 10 trainings, 4 exams, 7 projects in all sections
Post-test		85	0%	Projects in all sections and determining that all objectives are covered in the project content. For comparison on this report only original 7 objectives are included.

***Not all business students were required to show competency in computer business applications and therefore enrollment dropped.**

Appendix B: Percentage of Compliance with assessment objectives by semester

Objectives (Skill Tested)	Fall 2008	Spring 2009	Fall 2009	Fall 2009 with 7 Projects	Spring 2010	Spring 2010 with 3 Projects	Fall 2010 with 3-7 Projects	Spring 2011 With 7 Projects
Compute the Gross Pay	**51.46%**	**60.23%**	**60.00%**	74.42%	**41.33%**	**68.00%**	**69.37%**	75.56%
Use the IF Function	**67.96%**	**63.16%**	**69.05%**	79.07%	**52.67%**	**40.00%**	**48.65%**	73.33%
Start Microsoft Office Excel 2007	96.12%	98.83%	100.00%	95.35%	100.00%	100.00%	98.20%	94.44%
Apply Number Formatting	90.29%	92.40%	89.05%	88.37%	94.40%	92.00%	92.79%	95.56%
Copy the Formulas with the Fill Handle	86.41%	88.30%	89.05%	93.02%	88.00%	92.00%	91.89%	91.11%
Insert a Row and Compute Totals	77.18%	80.12%	79.52%	76.74%	82.67%	80.00%	87.39%	86.67%
Change the Chart Type	**67.96%**	70.18%	75.24%	**69.77%**	**69.33%**	76.00%	76.58%	76.67%
Average Compliance	*76.77%*	*79.03%*	*80.27%*	*82.39%*	*75.49%*	*80.00%*	*80.69%*	*84.76%*
Objective not meeting at least 70% compliance	3	2	2	1	3	2	2	0