INFORMATION SYSTEMS EDUCATION JOURNAL

Volume 17, No. 5
October 2019
ISSN: 1545-679X

In this issue:

iscap
Information Systems & Computing Academic Professionals

The **Information Systems Education Journal** (ISEDJ) is a double-blind peer-reviewed academic journal published by **ISCAP** (Information Systems and Computing Academic Professionals). Publishing frequency is six times per year. The first year of publication was 2003.

ISEDJ is published online (http://isedj.org). Our sister publication, the Proceedings of EDSIGCON (http://www.edsigcon.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 EDSIGCON 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 under 40%.

Information Systems Education Journal is pleased to be listed in the 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. Special thanks to members of AITP-EDSIG who perform the editorial and review processes for ISEDJ.

INFORMATION SYSTEMS EDUCATION JOURNAL

Editors

2019 ISEDJ Editorial Board

Software Concepts Emphasized in Introductory Programming Textbooks

Kirby McMaster, Ret.
kmcmaster@weber.edu

Brian Rague
brague@weber.edu

School of Computing
Weber State University
Ogden, UT 84408

Samuel Sambasivam
ssambasivam@apu.edu
Department of Engineering and Computer Science
Azusa Pacific University
Azusa, CA 91702

Stuart L. Wolthuis
stuart.wolthuis@byuh.edu
Faculty of Mathematics and Computing
Brigham Young University – Hawaii
Laie, HI 96762

Abstract

In this research study, we performed a *content analysis* of selected introductory programming textbooks for three languages to examine which software development concepts are emphasized in these books. Our goal was to determine which concepts are considered to be most representative of software development based on the topics emphasized by the textbook authors. We counted how often programming words appeared in samples of C++, Java, and Python books. We discovered which concepts are consistently supported for all three languages. We also noted those concepts that are favored by just one or two languages. Our summarized results lead to several conclusions that are relevant to the choice of a language for an introductory programming course.

Keywords: Java; C++; Python; programming; CS1; CS2; content analysis;

1. INTRODUCTION

Two current questions in Computer Science are: (1) What concepts should be taught in an introductory programming course, and (2) What language should be taught in the course? Debate on these questions has continued for decades, with no clear resolution in sight (Brilliant &

Wiseman, 1996; Siegfried, Chays, & Herbert, 2008; CC 2001; CSC 2013). The two questions are related, in that various programming languages historically have been designed based on differing conceptual frameworks.

The early years of computing saw advances in programming from machine language to assembly language to higher-level languages (such as FORTRAN and COBOL). The ability to give instructions to a computer in a language closer to the problem domain is one of the greatest inventions in computing. When employees learned how to program within the work environment, little attention was paid to sound programming concepts and practices because of the coding flexibility afforded by higher-level languages.

As the next generation of higher-level languages was developed (e.g. Algol and PL/I), designers took advantage of previous experience to consider a wider range of language options. During this period, a few languages were developed specifically for teaching programming (e.g. Basic and Pascal). The availability of languages designed for a variety of purposes encouraged teachers to present programming concepts beyond simple language-specific syntax features.

Languages were developed using different computational models, including functional languages (e.g. LISP, Haskell, Scheme) and logical languages (e.g. Prolog). In the relational database world, procedural languages (e.g. relational algebra) and non-procedural languages (e.g. SQL) were considered and implemented. Structured programming concepts were promoted as best practices to develop and maintain evolving complex business applications.

Object-oriented languages C++, Java and Python evolved from C or special purpose web and scripting languages. In the current academic environment, the above three object-oriented languages are among the most popular candidates for teaching introductory programming (Guo, 2014).

The decision about which programming paradigm to teach beginning students influences the choice of introductory language. The paramount question for an effective introductory programming course remains "What concepts to teach?", followed by "Which language best supports these concepts?". The increased demand for programming courses for liberal arts students has led to the development of what are termed CS0 courses (Sooriamurthi, 2010). The preferred programming language for a CS1 or CS2 course for Computer Science majors is often different from the language taught to non-majors (Hertz, 2010).

1.1 Purpose of this Research

Many research studies have been performed in recent years on which language is best for an introductory programming course (de Raadt, Watson, & Toleman, 2002). In an effort to contribute to this discussion, our research focuses on C++, Java, and Python, which are common CS1 and CS2 languages. Rather than argue the merits of these languages for teaching programming, we performed a *content analysis* (Krippendorff, 2012) of C++, Java, and Python textbooks to determine how well they support teaching fundamental programming concepts.

Our primary assumptions are that the framework of the author is reflected by the words used frequently in the textbook, and that the framework of interest is one that is appropriate for an introductory programming course. From the author's choice of words, we can judge how well the textbook will contribute to the generally recognized objectives of an introductory programming course.

2. METHODOLOGY

This section of the paper describes the methodology used to collect word frequency data from selected C++, Java, and Python textbooks. The words we are searching for represent important concepts for an introductory programming course. In this study, we did not start with an initial list of concepts. We recorded all words we found in the books, and eliminated those that did not relate to computer programming.

2.1 Sample of Textbooks

We collected a sample of 5 C++ books, 5 Java books, and 7 Python books. We included more Python books because they tended to be shorter. We wanted our sample to include popular books in all three languages. To reduce research costs, we chose textbooks that were available on the Internet and could be downloaded as PDF files. For example, we obtained C++ books by Prata (2005) and Lafore (2002), Java books by Schildt (2007) and Wu (2010), and Python books by Lutz (2011) and Zelle (2002). Overall, we obtained a fairly representative sample of books, but some were older editions.

2.2 Convert PDF Files to Text Files

To perform word searching and counting, Adobe Reader provides a menu option to convert the contents of a PDF file into a text file. We used

Adobe Reader to create a text file for each of the textbooks in our study.

We noticed that the text file versions of the books included many character strings containing digits, punctuation, and other non-alphabetic symbols. To simplify our counting of concept words, we wrote a Python program that (after changing C++ to CPP) removed all non-letter symbols except apostrophes, and replaced them with blank characters.

We included apostrophes to allow contractions (e.g. *don't*, *g'day*) to be counted as words. We considered allowing hyphens, but they were not used consistently by the authors (e.g. *floating-point* vs. *floating point*). Our Python program also converted all letters to lower-case.

Since we were searching for words that represent programming concepts, our Python program included a function to remove most of the words on Fry's list of 100 most frequent English words (UEN, 2015). A few of Fry's top 100 words can be interpreted in a programming context (e.g. *number*, *long*), which we retained. Instead, we modified the frequent word list to include some non-programming words from Fry's second 100 words (e.g. *only*, *most*). The total number of distinct words on our common word list was 110. By screening out common words, we shrunk the number of original words by more than 40%.

In the Python program, we also added a second function to convert many plural nouns and verbs to singular form. This reduced the number of distinct words further, since only the singular forms appeared in the generated text files. Our Python program provided a filtered set of text files consisting only of letters (and apostrophes), blanks, and substantially fewer words.

2.3 Word Groups for Concepts

A single programming concept can be expressed in more than one form. For example, a noun concept can be presented in singular or plural form (e.g. variable, variables). Verbs can also be written in singular or plural form, as well as with various tenses (e.g. solve, solves, solved, solving). Often, the same concept is described by both a noun and a verb (e.g. inheritance, inherit). In some cases, synonyms representing similar ideas can be used to represent a concept (e.g. record, structure). Some concepts are written not as a single word but as a sequence of words (e.g. structured programming).

Our goal was to count how often an author referred to a programming concept, but our counting software was designed to count individual words. For this reason, we defined a word group for each concept. In this study, a word group consists of a set of nouns and verbs that represent the same concept. We occasionally included synonyms in the same word group. To get a textbook count for a concept, we summed the frequencies for each of the words in the word group.

2.4 Word Counts and Word Rates

We used a program called TextSTAT (Huning, 2007) to obtain word counts for all words in our modified text files. With TextSTAT, a "Corpus" is created to hold a list of text files to examine simultaneously. We defined a corpus for each programming language: C++, Java, and Python. We linked each corpus to the transformed textfiles for the language. The total word counts for the three languages were nearly the same, having about 900,000 words for each language. We recorded the frequencies for each word and combined them into counts for word groups.

Although total word counts were close for each language, the sets of textfiles for each language do contain different total numbers of words. The Java books have a slightly greater total word count than the Python and C++ books. To standardize the counts, we converted each word count for a concept to a *word rate*. The rate we chose was "per 100,000 words". That is, we divided the concept word count by the total number of words in the set of textfiles for the language, and then multiplied by 100,000.

For example, the 5 C++ textfiles contained a total of 868,902 words. The word count for *object* in these files is 10,264. This count is rescaled to a word rate as shown below:

word rate = (10,264/868,902) * 100,000 = 1,181.3

This indicates that the *object* concept is mentioned 1,181.3 times per 100,000 words in the C++ files. Word rates were calculated for each concept in each language.

3. ANALYSIS OF DATA

The purpose of this research is to distinguish the frequency in which programming concepts appear in textbooks for C++, Java, and Python. For every concept, we counted the number of occurrences of each word group member in the textbooks. Prior to obtaining the results presented below, our samples of textbook words were filtered by replacing non-letter characters with blanks, removing common English words, and converting plural nouns and verbs to singular form.

3.1 Word Frequency Distributions

Selected statistics for the word frequency distribution for each language are shown in Table 1 (all tables located in the appendices). The samples consisted of 868,902 C++ words, 939,851 Java words, and 902,702 Python words. Most of these words are repeated multiple times in the textbooks. For example, in the C++ sample, the maximum frequency word is *function*, which appears 18,073 times. The maximum frequency words are *class* (18,009 times) in the Java books and *python* (10,946 times) in the Python books.

The TextSTAT program uses the term *word form* to refer to a specific word string, such as *object*, that represents one word. The total number of word forms for each language are given in Table 1. Note that the Java sample has the greatest number (26,587) of word forms and also the greatest number of word forms (11,120) that appear just once.

A surprisingly large number of words have a frequency of 1. Many of these words were not actual words, but consisted of several words concatenated together into a single string. We suspect that this anomaly is due to an imperfect conversion of PDF files into text files and the extensive use of variable names in programming texts.

When we checked word counts for each of the 5 Java books separately, we observed that one of the books had a noticeably larger number of words having a frequency of 1. Since we are looking for frequent words that represent programming concepts, words that appear only once should have little effect on the word counts of interest. However, a large number of unduplicated words can slightly bias the word rates calculated from word counts. Rather than remove this Java book having the large number of distinct words, we chose to ignore all words having a frequency of 1 when performing our word rate calculations. This reduced the total word counts for C++, Java, and Python to the values shown on the bottom line of Table 1.

3.2 Word Rate Distribution

Since our focus in this paper is on frequent words in the textbooks, we need to provide a criterion for determining if a word is frequent. The actual word frequencies range from 1 up to a maximum for each language. In C++ the maximum frequency is 18,073 for *function*. Because the total word counts differ for each language, we rescaled word frequencies into *word rates* as described above. Our criterion for defining frequent words involves setting a threshold word rate for frequent words.

Table 2 describes the distributions for C++, Java, and Python in terms of word rate intervals. If a frequent word were defined to be one with a word rate above 800 (words per 100,000 words), then there would be 10 + 7 + 3 = 20 frequent words (not all distinct). These 20 frequent words are not uniform across languages. For example, the word object has word rates above 800 for C++ and Java, but not Python.

In this paper, we chose to define a *frequent* word as one with a word rate above 250. This gives us a reasonable number of words to study for each language and across languages.

Not all frequent words are *programming* words. The words *example*, *chapter*, *using*, and *same* are frequent for all three languages, but we do not interpret these words as programming concepts.

3.3 Consistently Frequent Concepts

We further define a word to be *consistently frequent* when it is frequent for all three languages. The consistently frequent programming words, together with their word rates for C++, Java, and Python, are listed in Table 3. The words are ordered by decreasing average word rate. Because these words are used frequently by authors for all three languages, they represent a measure of agreement on important programming concepts irrespective of language.

The most frequent programming word across all three languages is *class*, which is a keyword for each language (shown in **bold**) and also the most frequent Java programming word. The most frequent C++ programming word is *function*. The most frequent Python word is *python*. However, *function* and *python* are not consistently frequent. Of the 16 programming words in Table 3, the C++ word rates are highest for 7 words, 3 words have the highest rates for Java, and the remaining 6 words have the highest rates for Python.

The OOP words *class* and *object* have very high rates for C++ and Java. This suggests a substantial emphasis on OOP in the Java and C++ books. For most Table 3 words, the rates for C++ and Java are fairly similar.

The frequent word *type* has a lower word rate for Python, where data types are dynamic and are not explicitly defined. The frequent word *list* has a higher word rate in Python because (variable size) lists are used in place of (fixed size) arrays. *File* has a higher Python word rate, perhaps due to the emphasis on multimedia in some Python books.

Six of the Table 3 words (*value*, *string*, *type*, *number*, *data*, *list*) refer to data characteristics and data structures. Three of the words (*program*, *code*, *line*) represent program segments. *Name* can refer to data (e.g. variables) or program components (e.g. functions).

3.4 Language Dependent Concepts

A number of programming words are frequent in one or two languages but not the third. For example, *function* is a frequent word in C++ and Python, but not in Java. We refer to these words as *language-dependent* programming concepts. These words reflect variation between languages about words that are important. Table 4 lists 18 programming words that have a word rate range (high minus low) above 275 and at least one word rate below 150.

For example, the word *reference* has word rates of 213.4 for C++, 85.3 for Java, and 209.7 for Python. This word is not included in Table 4 because the range of word rates is below 275. The purpose of this constraint is to highlight words with language rate disparities that are meaningful.

Excluding language names *cpp* (representing C++), *java*, and *python*, the Table 4 words include 3 C++ keywords, 4 Java keywords, and 1 Python keyword. Being a keyword can have some effect on word rates, especially if the word is used in sample code (e.g. *public* in C++ and Java). The importance of some keywords (like *class*) extends throughout programming. We now direct our attention to Table 4 words that are not keywords.

In C++ books, *method* is often replaced by the two-word term *member function* to designate functions that are part of a class. This can explain the high C++ rates for *function* and *member*. C++ uses a *compiler*, while Java and Python use a run-time environment or interpreter.

In C++ and Java, an *array* is more frequent than a (linked) list. *Pointers* are common in C and C++ for indirect addressing. *Declaration* of variables is required in C++ and Java, but not in Python. *Threads* and *events* are built into the Java language, but not C++.

If the language in a programming course switches from C++ to Java, then some of the frequent C++ concepts will not be well-supported in the Java books. Similarly, if the language switch is made from Java to Python, more programming concepts will be lost.

3.5 Less Frequent Concepts

We have presented programming words that have a word rate above 250 for at least one language. In this section, we examine selected non-frequent words representing concepts from object-oriented programming, structured programming, and software engineering. We might expect a majority of these concepts to be included in the content of an introductory programming course.

Object-oriented programming concepts have appeared often in Table 3 and Table 4. The OOP words *class* and *object* have high word rates in all three languages. In Table 5A, we show word rates for 3 defining characteristics of OOP.

Encapsulation and polymorphism have low word rates for all three languages. Inheritance does get some respect from C++ authors, with a word rate above 100. Maybe there is more discussion of class hierarchies in the C++ books. Encapsulation certainly should have higher rates, since it is a critical concept in modular programming and especially for classes. Polymorphism is difficult enough to pronounce much less explain in a textbook.

Table 5B lists 10 **structured programming** concepts. The first four Table 5B words--*sequence*, *selection*, *iteration*, and *recursion*--are the formal names for classic control structures. The next two words, *branch* and *loop*, are informal terms for *selection* and *iteration*, respectively. In all three languages, *loop* is much more frequent than *iteration*, but *branch* is not a popular substitute for *selection*.

The *block* concept has been central to structured programming since the days of Algol. Word rates for *block* are near 100 for C++ and Java, but smaller for Python. Python uses indentation instead of special symbols (e.g. braces) to designate the start and end of a *block* (or paragraph). The words *argument* and *parameter* are closely related. *Argument* is a frequent word for C++, but *parameter* has word rates below 200 for all three languages.

Procedure is a forgotten term in current language textbooks, perhaps due to the residual effects of the decision by C language designers to implement only functions. This design decision persists in C++, Java, and Python for various reasons.

The 16 **software engineering** concepts in Table 5C include project stages, activities, and byproducts that do not directly involve writing code. This list includes the frequent Java word *implementation* and the frequent C++ and

Python word *error*. These words were not included in Table 4 because their range of word rates was below 200. We might expect some of these concepts to receive less emphasis in an introductory programming course.

The first four words--*analysis*, *design*, *implementation*, and *maintenance*--describe the stages of the traditional software development life cycle (SDLC). *Implementation* (which includes writing code) has word rates between 102.6 and 252.7 for all three languages. *Design* has a word rate above 100 in the Java books. *Maintenance* and *quality* are almost an afterthought in all textbooks. Based on these books, don't hire an introductory programmer to do maintenance.

Additional observations about the software development word rates include the following. In software development, *requirements* and *specifications* are usually discussed together, in response to a *problem* request from a client. One formal SDLC document that is often prepared is a Software Requirements Specification (IEEE, 1998).

The word *documentation* does not appear often in C++ books (rate just above 25), but it does in Python books (rate almost 200). What does this say about the mindset of the authors of these textbooks? From our experience, many computer programmers do not like to document their work.

The word rates for *abstraction* are very low. The term may be too general to be used frequently in introductory programming books. This thought ignores arguments presented in the article "Is Abstraction the Key to Computing?" (Kramer, 2007).

The *model* (and *modeling*) concept has rates below 100, which appears low considering that most design work requires some form of modeling for both code and data. Modeling is the realization of abstraction. In introductory courses, much of the design work is usually provided by the instructor. The students focus on writing the programs.

Algorithm has a C++ word rate of almost 160, indicating that C++ books spend a reasonable amount of time explaining the nature of algorithms. Maybe this is one reason why C++ has a reputation for being "harder" than Java and Python.

The word rates for *test* are above 100, but the rates for *debug* are near 0. One possible explanation for this difference is that *test* does not imply that the programmer made a mistake, whereas debug suggests that something needs to be fixed. On the other hand, *error* has word rates that almost qualify it as a consistently frequent word. In commercial software development organizations, initial debugging is usually performed by the developers who write the code. Formal testing is more likely to be performed by specialized test groups, especially when a suite of tests must be re-run whenever the code is changed.

As a special note, if you want to teach students about *functional decomposition* or *data decomposition*, don't use one of these books. Word rates can't get much lower than 0.3.

4. SUMMARY AND CONCLUSIONS

The choice of programming language for an introductory Computer Science course influences the concepts that will be emphasized in the course. Discussion about which concepts to teach in a first course and what language best supports these concepts continues among faculty and professional organizations. This discussion has often led to the conclusion that no language is best for all situations (CSC, 2013). Our work attempts to contribute to this dialog by revealing which programming concepts are supported in textbooks for C++, Java, and Python.

We gathered a sample of textbooks that were restricted to those available in PDF format, converted the contents into text files, and then screened the files to remove or transform unnecessary material. We counted how often words that represent programming concepts appeared in the books, and then converted the frequencies into word rates. From the transformed data, we draw several conclusions.

A word is defined to be *frequent* for a language if its word rate is at least 250 per 100,000 words in the textbooks for that language. We found 16 programming words that are frequent for all three languages. Two of the words with the highest rates are *class* and *object*, which are central concepts for object-oriented programming. This list of concepts that are supported across languages is a good start for an introductory programming course.

We next searched for words that were frequent in one or two languages, but not all three. These words highlight differences between the languages. The word *function* is very frequent in C++ and Python, but not in Java. Java prefers the term *method.* Java considers all functions (and all code) to occur within a class. C++ uses the combined term *member function* for functions within a class, but C++ (and Python) allow functions to be defined outside of a class.

With its history from C, C++ provides explicit indirect addressing using *pointers*. Java makes indirect addressing implicit through the use of *references*. C++ and Java provide fixed size *arrays* as a common data structure. Python uses variable size *lists* (without mentioning the word *linked*). C++ and Java have a *character* data type, whereas characters in Python are represented as strings of length 1. Each language provides support for the above concepts, using possibly a different name, and sometimes involving a different underlying implementation (e.g. arrays vs. lists).

Among the other concepts, Java supports *threads* and *events* for real-time programming. C++ and Java, but not Python, require a *declaration* (*name* and *type*) for variables before they can be used.in a program. For words that are frequent in two languages, many of the word rates for C++ and Java are comparable. C++ and Java books seem to provide similar support for most of the frequent programming concepts. Python provides less support.

We also examined a selection of object-oriented programming, structured programming, and software development words that did not appear on our most frequent word lists. On a word-by-word basis, many of the comparative word rates are interesting, with several results standing out. Longer technical words (e.g. *polymorphism*, *iteration*, *requirement*, and *decomposition*) tended to have lower word rates, but there are exceptions (e.g. *selection* vs. *branch*). Word rate differences for *test*, *debug*, and *error* are hard to explain. Hopefully, the extremely low rates for *abstraction*, *maintenance*, and *quality* do not persist into more advanced programming textbooks.

Finally, both C++ and Java books provide reasonable support for most of the frequent programming concepts. Python provides less support. The ultimate choice of language for an introductory programming course must be based on considerations beyond textbook coverage of important concepts.

4.1 Future Research

Planned future research activities include:

1. Replicate this study with a larger, more representative sample of textbooks.

2. Examine variation in word rates between books within the same language.

3. Perform a similar study comparing textbooks for other candidate languages for an introductory programming course (e.g. PERL, Ruby, Javascript, Ada, Scheme).

5. REFERENCES

Brilliant, S. S., & Wiseman, T. R. (1996, March). The first programming paradigm and language dilemma. In *ACM SIGCSE Bulletin* (Vol. 28, No. 1, pp. 338-342). ACM.

Curricula, C. (2001). Computer Science. *Final Report, December*, *15*, 2001.

Joint, A. C. M. (2013). IEEE-CS Task Force on Computing Curricula. *Computer science curricula*.

De Raadt, M., Watson, R., & Toleman, M. (2002). Language trends in introductory programming courses. In *Proceedings of the 2002 Informing Science+ Information Technology Education Joint Conference (InSITE 2002)* (pp. 229-337). Informing Science Institute.

Guo, P. (2014). Python is now the most popular introductory teaching language at top us universities. *BLOG@ CACM, July*, 47.

Hertz, M. (2010, March). What do CS1 and CS2 mean?: investigating differences in the early courses. In *Proceedings of the 41st ACM technical symposium on Computer science education* (pp. 199-203). ACM.

Huning, M. (2007). TextSTAT 2.7 User's Guide. *TextSTAT, created by Gena Bennett*.

IEEE Computer Society. Software Engineering Standards Committee, & IEEE-SA Standards Board. (1998). IEEE recommended practice for software requirements specifications. Institute of Electrical and Electronics Engineers.

Kramer, J. (2007). Is abstraction the key to computing?. *Communications of the ACM*, *50*(4), 36-42.

Krippendorff, K. (2012). Content Analysis: An Introduction to Its Methodology, 3rd Ed. SAGE Publications.

Lafore, R. (2002). Object-Oriented Programming in C++ (4th ed). Sams Publishing.

Lutz, M. (2011). Programming Python (4th ed). O'Reilly Media.

Prata, S. (2005). C++ Primer Plus (5th ed). Sams Publishing.

Schildt, H. (2007). Java The Complete Reference, 7th Ed. McGraw-Hill.

Siegfried, R. M., Chays, D., & Herbert, K. (2008, July). Will there ever be consensus on cs1?. In *FECS* (pp. 18-23).

Sooriamurthi, R. (2010). The essence of object orientation for CS0: concepts without code. *Journal of Computing Sciences in Colleges*, *25*(3), 67-74.

UEN (2015). High Frequency Words--Fry Instant Words. *Utah Education Network.* Retrieved Feb 21, 2017 from http://www.uen.org/

UEN (2018). k-2educator/word_lists.shtml *Utah Education Network.* Retrieved Feb 21, 2017 from http://www.uen.org/

Wu, C. T. (2010). An Introduction to Object-Oriented Programming with Java (5th ed). McGraw-Hill.

Zelle, J. (2002). Python Programming: An Introduction to Computer Science. Wartburg College Printing Services.

Appendices

Table 1: Word Frequency Distribution Summary

Statistic	**C++**	**Java**	**Python**
Textbooks	5	5	7
Authors	6	8	10
Total Words	868,902	939,851	902,702
Max Count	18,073 *function*	18,009 *class*	10,946 *python*
Min Count	1	1	1
Word Forms	17,328	26,587	21,644
Forms: count>1	11,716	15,467	14,620
Forms: count=1	5,612	11,120	7,024
PctForms:count=1	32.4%	41.8%	32.5%
*Words:count>1	863,286	928,749	895,678

* Used to calculate word rates

Table 2: Word Forms by Word Rate

Word Rate	**C++**	**Java**	**Python**
800.0+	10	7	3
400.0 - 799.9	18	15	19
200.0 - 399.9	49	40	43
100.0 - 199.9	97	121	113
50.0 - 99.9	190	218	228
25.0 - 49.9	326	325	372
* Words: count>1	863,286	928,749	895,678

* Used to calculate word rates

Table 3: Consistently Frequent Programming Concepts

(Rate > 250 for all 3 languages)

Rates for keywords are shown in **bold**

	Concept	C++ Rate	Java Rate	Python Rate	Mean
1	class	**1,929.0**	**1,939.1**	**641.9**	**1,503.33**
2	object	1,188.9	1,163.7	629.2	994.0
3	value	1,019.1	835.8	675.0	843.3
4	program	890.1	913.1	688.4	830.8
5	string	855.0	857.1	529.2	747.1
6	type	861.7	782.7	370.7	671.7
7	file	571.3	551.4	890.4	671.0
8	line	450.8	498.9	611.8	520.5
9	number	597.7	543.6	415.9	519.1
10	name	493.1	481.7	580.0	518.3
11	call	552.1	486.3	494.6	511.0
12	data	523.5	412.0	394.3	443.3
13	list	302.1	358.1	568.2	409.5
14	code	374.7	310.5	433.0	372.7
15	element	443.9	254.6	288.6	329.0
16	input	267.0	251.6	296.0	271.5

Table 4: Language-Dependent Concepts
at least 1 rate < 150, and range > 275
Rates for keywords are shown in **bold**

	Concept	C++ Rate	Java Rate	Python Rate	Range
1	function	2,093.5	58.4	696.8	2,035.1
2	python	3.5	0.1	1,222.1	1,222.0
3	cpp	1,192.2	0.0	0.2	1,192.2
4	java	11.8	1,072.5	61.0	1,060.7
5	member	719.8	119.7	24.5	695.4
6	operator	**776.6**	146.9	133.2	643.4
7	array	641.4	486.7	34.4	607.0
8	public	**197.7**	**621.0**	38.9	582.1
9	pointer	551.0	17.3	11.3	539.8
10	module	10.5	5.7	461.2	455.5
11	thread	0.8	414.9	210.0	414.1
12	constructor	395.8	268.8	49.1	346.7
13	event	8.2	336.7	132.7	328.5
14	declaration	333.0	213.2	19.1	313.9
15	static	**163.1**	**329.4**	17.9	311.5
16	compiler	300.6	72.5	8.0	292.6
17	import	1.3	**185.3**	**291.5**	290.2
18	interface	64.9	**341.0**	161.7	276.1

Table 5A: Object-Oriented Programming Concepts

	OOP Concepts	C++ Rate	Java Rate	Python Rate	Mean
1	encapsulation	6.3	5.4	5.9	5.9
2	inheritance	129.4	45.1	29.9	68.1
3	polymorphism	28.7	17.7	6.1	17.5

Table 5B: Structured Programming Concepts

	StructProg Concepts	C++ Rate	Java Rate	Python Rate	Mean
1	sequence	98.0	97.7	121.8	105.8
2	selection	38.3	45.8	44.4	42.8
3	iteration	22.5	18.8	18.8	20.0
4	recursion	24.6	30.9	17.1	24.2
5	branch	3.2	6.9	4.1	4.8
6	loop	215.8	174.0	165.9	185.2
7	block	95.6	100.7	48.3	81.5
8	argument	436.8	181.8	184.6	267.7
9	parameter	154.2	179.2	116.7	150.0
10	procedure	3.6	6.4	4.8	4.9

Table 5C: Software Engineering Concepts

	Software Dev Concepts	C++ Rate	Java Rate	Python Rate	Mean
1	analysis	10.8	11.4	16.7	13.0
2	design	74.6	112.1	45.7	77.5
3	implementation	147.3	252.7	102.6	167.5
4	maintenance	3.2	2.3	3.7	3.1
5	problem	128.2	123.9	94.3	115.5
6	requirement	24.1	11.5	14.6	16.7
7	specification	89.9	147.7	92.3	110.0
8	abstraction	7.4	6.2	4.8	6.2
9	model	41.7	80.8	50.8	57.8
10	algorithm	159.5	77.3	68.2	101.7
11	decomposition	0.1	0.3	0.1	0.2
12	test	122.1	136.1	122.1	155.8
13	debug	12.2	5.1	8.2	8.5
14	error	242.9	198.8	214.5	218.7
15	documentation	26.6	89.6	195.9	104.0
16	quality	4.3	2.9	3.1	3.4

IT Infrastructure Strategy in an Undergraduate Course

Ronald E. Pike
rpike@cpp.edu
Computer Information Systems
Cal Poly Pomona
Pomona, CA 91768

Brandon Brown
Bbrown118@coastline.edu
Coastline College
Fountain Valley CA 92708

Abstract

Cloud computing has grown immensely in the past decade and is becoming increasingly important to organizations. The Computer Information Systems (CIS) program at Cal Poly Pomona has gone through a re-visioning process that included the addition of a course on IT infrastructure to address the growth of cloud computing and resulting changes in IS/IT organizations. This paper reports on the newly created IT infrastructure course, which is in the beginning of two emphases (curriculum tracks) in the CIS department.

Keywords: IS Infrastructure, Cloud Computing, IS Curriculum, Kubernetes

1. INTRODUCTION

IT infrastructure has been viewed as a strategic asset in organizations for at least twenty years (Broadbent & Weill, 1998; Weill, Subramani, & Broadbent, 2002) and continues to grow increasingly relevant as changes in IT infrastructure offer organizations new avenues for differentiation with reduced cost and risk. Despite the importance of the integrated IT infrastructure, undergraduate Information Systems (IS) courses continue to offer distinct courses such as telecommunications and database that focus on the individual IT infrastructure components. This paper presents an overview of a holistic IT Infrastructure course within a CIS program and the inclusion of IT strategy as a component.

Courses on primary components of an IT infrastructure, such as database, provide insights into the strategic value of IT infrastructure. However, such coverage may not demonstrate the strategic synergies that underlie the integration of IT infrastructure components. This paper reports on a process to develop an IT Infrastructure course that will replace an introductory business telecommunications course in an undergraduate IS program. The IT Infrastructure course introduces the individual IT components while also highlighting the integrated nature of these technologies and synergistic relationships made possible through integrated and automated IT infrastructure. The course highlights the strategic role of IT infrastructure and includes technical topics including hybrid-cloud infrastructure, telecommunications and systems administration.

This paper is broken into four major components including hybrid-cloud infrastructure, telecommunications, systems administration and coverage of the strategic role that IT infrastructure can play within organizations. The paper also addresses an IT Infrastructure course platform selected to offer the course. Key

elements of the curriculum plan for the program are included to provide context for the IT Infrastructure course and to support arguments related to this course and the IS model curriculum.

While this manuscript and the related course are considered to be a minor contribution to the literature in terms of originality, the focus of the paper is to motivate a conversation on the inclusion of IT infrastructure in undergraduate IS/IT education and perhaps the consideration of a review of the Model IS Curriculum in light of recent innovations in IT infrastructure and automation capabilities made possible by current and emerging data science capabilities.

2. IT INFRASTRUCTURE & STRATEGY

The choice to present fundamentals of IT strategy as part of a course on IT infrastructure in an undergraduate program initially drew skepticism. An argument was made that this topic is beyond the preparation level of undergraduate students and is in the purview of a master's program. The conclusion; however, was that while IT Infrastructure strategy does belong in a master's program, an introduction to the topic is still appropriate in the undergraduate curriculum. The close integration and shared dependencies of IT infrastructure components in a cloud environment make IT infrastructure strategy more important than ever.

Not only was this course placed in the undergraduate curriculum, but also placed, along with an introduction to operating systems course, at the beginning of the information security and forensics emphasis in the CIS curriculum. The IT infrastructure course was placed at the beginning of the business intelligence emphasis as well. Figure 1 shows the CIS core required of students in each of the three CIS emphases. Figure 2 shows the specific classes required of students in the Information Security and Forensics emphasis and Figure 3 shows the classes required of students in the Business Intelligence emphasis.

General education, college of business core requirements, and mathematics preparation courses required of students in the CIS program are not presented as they are similar to course requirements for most information systems programs.

Computer Information Systems Core
CIS 3050 – Database Design and Development
CIS 3090 – Object-Oriented Programming for Business
CIS 3252 – Business Intelligence

Figure 1 Computer Information Systems Core

Information Security and Forensics Emphasis
Core
CIS 2650 – Contemporary Operating Systems
CIS 2670 – IT Infrastructure
Electives *(select 4)*
CIS 3470 – Telecommunications Networks
CIS 4333 – Information Systems Auditing
CIS 4670 – Network Security
CIS 4710 – Information Security
CIS 4810 – Computer Forensics

Figure 2 Information Security and Forensics Emphasis

Business Intelligence Emphasis
Core
CIS 2670 – IT Infrastructure
CIS 3150 – Systems Analysis and Design
Electives *(select 4)*
CIS 3454 – Data Warehousing
CIS 3650 – Digital Analytics
CIS 4321 – Data Mining
CIS 4567 – Big Data Analytics
CIS 4680 – Advanced Data Analytics

Figure 3 Business Intelligence Emphasis

Hybrid-Cloud Infrastructure
NIST Special Publication (SP) 800-145 defines cloud computing as "a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction" (Mell & Grance, 2011). Cloud computing offers benefits such as faster time-to-market and improved scalability due to on-demand provisioning of pooled and shared computing resources.

NIST SP 800-145 goes on to define four cloud computing deployment models including hybrid-

cloud. Hybrid-cloud is defined as "a composition of two or more distinct cloud infrastructures (private, community, or public) that remain unique entities, but are bound together by standardized or proprietary technology that enables data and application portability (e.g., cloud bursting for load balancing between clouds)."

The newly revised CIS curriculum focuses on public/private hybrid-cloud infrastructure given the benefits this deployment methodology offers organizations. Private-cloud infrastructure offers organizations the ability to run secure applications within their own control and with lower operational costs for applications with steady demand. Public-cloud infrastructure offers the ability to outsource common applications (i.e. email) as well as applications requiring significant resource scalability. Hybrid cloud creates a bond between the two environments offering the ability to transfer loads between the public and private clouds to meet organizational needs. The hybrid-cloud deployment models offers organizations the ability to take advantage of both public and private cloud computing environments and by binding the two environments together, organizations can enjoy nearly unlimited scalability, reduced costs, and the ability to optimize cost benefits with the ability to transfer workloads back and forth between public and private clouds (Mazhelis & Tyrväinen, 2012).

Telecommunications

Telecommunications is an important component in our IT infrastructure course in part because the introductory telecom course was sacrificed to make room for the IT infrastructure course. In addition, telecommunications is increasingly integral and embedded within the IT infrastructure given the emergence of Software Defined Networks (SDN) and Network Function Virtualization (NFV). In fact, network functionality is increasingly embedded into applications within the cloud infrastructure and less reliant on stand-alone hardware devices.

SDN separates the network control and forwarding planes allowing network operators an enhanced level of managerial control of the entire network (Shahzad, Mujtaba, & Elahi, 2015). NFV seeks to move network functions from hardware devices such as routers and firewalls to software applications. These two innovations offer significant promise when coupled together to provide unprecedented control and automation to computer networks (Duan, Ansari, & Toy, 2016). Automation and holistic managerial control of networks, coupled with machine learning, are required for networks to integrate with a dynamic hybrid-cloud environment and build needed operational network services in real time.

Systems Administration

Systems administration has not typically been a part of information systems programs. However, the integration of IT systems and services evident in the virtualization of computing resources, and now the virtualization of networks and network services, require IT/IS professionals to wield substantial systems administration skills to manage and optimize IT infrastructure components. Campbell and Cohen (2005) suggest the most critical tools to teach in a systems administration course are:

- Introduction to scripting
- A scripting language (such as Python)
- Text manipulation and regular expression pattern matching
- Interfacing with operating systems shells
- Operating systems namespaces (like DNS and NIS) and file systems (like NFS)
- Network concepts

While the list of needed skills for systems administration may change some in a cloud computing context, the list above from Campbell and Cohen remains relevant and will allow students to interact and provide operational control over the myriad applications distributed across the hybrid-cloud system. This is further reinforced by (Appiahene, Kesse, & Ninfaakang, 2016) who emphasized the importance of system administrators to reducing time barriers to market for businesses. A potential addition to the critical tools for system administration is an introduction to machine learning. Scripting languages such as Python offer significant power and flexibility for IS professionals in configuring and managing IT infrastructure. However, recent developments in AI and machine learning, coupled with the programmability features of modern IT infrastructure, offer new opportunities in automation but require fundamental knowledge and skills with machine learning.

IT Infrastructure Strategy

IT infrastructure has evolved from hardware to software and IT infrastructure strategy must adapt to leverage this powerful new platform. The importance of IT strategy has been well documented even before the advent of cloud computing (Byrd & Turner, 2001; Weill, Subramani, & Broadbent, 2003). Over the past

20 years, IT infrastructure components were virtualized. Recent changes to IT infrastructure include Software Defined Networks (SDN) and Network Function Virtualization (NFV) which allow for the completion of the transition to a fully virtualized infrastructure. The ability to control and manage IT infrastructure via software, means that perhaps IT strategy is best expressed in software via algorithms. The software platform selected to teach and provide "hands-on" learning in IT infrastructure and strategy is Kubernetes running in Red Hat OpenShift.

3. IT INFRASTRUCTURE CLOUD COURSE

The IT Infrastructure course is based on Kubernetes and specifically on the Red Hat OpenShift platform. Kubernetes is a portable, extensible open-source platform for managing containerized workloads and services ("What is Kubernetes?," n.d.). Kubernetes is based on a series of technology enhancements for hosting IT systems at scale including LXC and Docker Containers. David Bernstein stated "Containers, Docker, and Kubernetes seem to have sparked the hope of a universal cloud application and deployment technology" (Bernstein, 2014, p. 84). This hope of a universal cloud application and deployment technology underlies the reason for moving our program in this direction.

Red Hat OpenShift is an enterprise-grade application platform that hosts containers with Kubernetes. Kubernetes running on Red Hat OpenShift was determined to be a reasonable platform for our program given its use in industry and the prescribed computing infrastructure requirements in the IS 2010 Curriculum Guidelines for Undergraduate Programs in Information Systems. The IS 2010 Curriculum Guidelines state "Information Systems students and faculty must have access to computing facilities at least equivalent to those used in a typical organization operating within a program's domain" (Topi et al., 2010, p. 2). The IS 2010 Curriculum Guideline document points out that access to computing facilities at least equivalent to those used in organizations is needed to prepare students for their profession and for faculty to contribute to the creation of new knowledge in the field.

The Red Hat Openshift infrastructure was placed in the Mitchell C. Hill Student-run Data Center (SDC) at Cal Poly Pomona. As such, senior students in the program worked with engineers from Red Hat and other partner organizations to design, and now build and operate the infrastructure for this course.

4. THEORETICAL IMPLICATIONS

If Kubernetes becomes a universal cloud application and deployment technology as suggested by Bernstein (Bernstein, 2014, p. 84), then a new operating environment is made possible in which IS professionals will exercise direct control of IT infrastructure. Infrastructure designers and operators will operate the cloud environment (public and private) in much the same way that power companies control their infrastructures. IS professionals will consume services in a manner that best serves organizational needs from a set of cloud services run internally or through external providers.

In this scenario, IS professionals have the ability to exercise much greater control over their environments automating tasks via scripts and even using artificial intelligence where appropriate to manage variation in tasks designed to meet predefined criteria. In this case, a new class of IS/IT theories become possible. Mid-range theories with a limited scope leading to testable hypotheses are needed in any discipline (Merton, 1968). The IS/IT field is in greater need of such theories than most given the relatively scant theoretical foundation for much of the work in the field. A universal cloud application and deployment technology, such as Kubernetes, may also open the door to a more general (grand) theory of IS/IT which has so far eluded the discipline (Weber, 1997).

The need for the IS/IT community to integrate IT infrastructure more closely with business systems and processes is made clear in a recent paper on big data analytics in healthcare (Wang, Kung, & Byrd, 2018). This paper mentions IT infrastructure ten times with calls for aligning business value of information with appropriate IT infrastructure (p. 6), capturing benefits from big data analytics (p. 6) and creating shareable and reuseable IT resources that provide a foundation for business applications (p. 7). There is a significant, and growing, need to align IT infrastructure with business processes and IS researchers and practitioners must be at the forefront of this effort.

5. CONCLUSION

The newly revised curriculum is designed around the notion that hybrid (public/private) cloud technology is an important and growing platform for business computing. The success of such a platform relies upon a general computing platform that permits workloads to transition

easily between private and public cloud infrastructures. Such an infrastructure provides significant advantages in creating robust, cost effective, scalable and secure compute platforms. A new course titled IT Infrastructure is at the beginning of the CIS emphases in Information Security and Forensics as well as Business Intelligence, which offers students an introduction to IT infrastructure and the increasingly importance of customization and integration of IT infrastructure with business processes.

6. REFERENCES

Appiahene, P., Kesse, B. Y., & Ninfaakang, C. B. (2016). Cloud Computing Technology Model for Teaching and Learning of ICT. International Journal of Computer Applications, 143(5), 22–26.

Bernstein, D. (2014). Containers and Cloud: From LXC to Docker to Kubernetes. IEEE Cloud Computing, 1(3), 81–84.

Broadbent, M., & Weill, P. Leveraging the New Infrastructure: How Market Leaders Capitalize on Information Technology. (May 1998). Retrieved from http://proxy.library.cpp.edu/login?url=http://search.ebscohost.com/login.aspx?direct=true&AuthType=ip,uid&db=buh&AN=123511762&site=ehost-live&scope=site

Byrd, T. A., & Turner, D. E. (2001). The exploratory examination of the relationship between flexible IT infrastructure and competitive advantage . *Information & Management*, *39*(1), 41.

Campbell, W., & Cohen, R. (2005). Using system administrator education in developing an IT degree in a computer science department. In *Proceedings of the 6th conference on Information technology education* (pp. 319–321). Newark, NJ USA.

Duan, Q., Ansari, N., & Toy, M. (2016). Software-Defined Network Virtualization: An Architectural Framework for Integrating SDN and NFV for Service Provisioning in Future Networks. *IEEE Network*, *30*(5), 10–16.

Mazhelis, O., & Tyrväinen, P. (2012). Economic aspects of hybrid cloud infrastructure: User organization perspective. *Information Systems Frontiers*, *14*(4), 845–869. https://doi.org/10.1007/s10796-011-9326-9

Mell, P., & Grance, T. (2011). *The NIST Definition of Cloud Computing* (No. SP 800-145). National Institute of Standards and Technology. Retrieved from https://csrc.nist.gov/publications/detail/sp/800-145/final

Merton, R. (1968). *Social Theory Social Structure*. New York: Free Press.

Shahzad, N., Mujtaba, G., & Elahi, M. (2015). Benefits, Security and Issues in Software Defined Networking (SDN). *NUST Journal of Engineering Sciences*, *8*(1), 38–43.

Topi, H., Kaiser, K., Sipior, J., Valacich, J., Nunnamaker, J., de Vreede, G., & Wright, R. (2010). *Curriculum Guidelines for Undergraduate Degree Programs in Information Systems* (p. 97). ACM.

Wang, Y., Kung, L., & Byrd, T. A. (2018). Big data analytics: Understanding its capabilities and potential benefits for healthcare organizations. *Technological Forecasting and Social Change*, *126*, 3–13. https://doi.org/10.1016/j.techfore.2015.12.019

Weber, R. (1997). *Ontologica! Foundations of Information Systems*. Melbourne, Victoria, Australia: Coopers & Lybrand.

Weill, P., Subramani, M., & Broadbent, M. (2002). Building IT Infrastructure for Strategic Agility. *MIT Sloan Management Review*, *44*(1), 57–65.

Weill, P., Subramani, M., & Broadbent, M. (2003). IT Infrastructure for Strategic Agility. Retrieved from http://proxy.library.cpp.edu/login?url=http://search.ebscohost.com/login.aspx?direct=true&AuthType=ip,uid&db=ecn&AN=0840831&site=ehost-live&scope=site

What is Kubernetes? - Kubernetes. (n.d.). Retrieved July 29, 2018, from https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/

Building the Physical Web: A Campus Tour Using Bluetooth Low Energy Beacons

Jake OConnell
oconnel_jake@bentley.edu

Mark Frydenberg
mfrydenberg@bentley.edu

Computer Information Systems Department
Bentley University,
Waltham, Massachusetts

Abstract

Estimote Bluetooth Low Energy (BLE) beacons deployed throughout a university campus enabled the creation of interactive tours for prospective and accepted students at Fall and Spring open house events. By configuring their mobile devices with an app to detect the beacons' signals, participants could view webpages containing information based on their current physical locations from the web addresses pushed to their devices by nearby beacons.

Google Analytics and relational database reporting functions provided information related to the behavior of campus guests. This study presents lessons learned from deploying a Physical Web and results of analyzing user behavior based on the analytics data. The conclusions drawn from this data can be used to improve campus visits for prospective students and supplement decision making in university admissions offices. Participants in a follow-up survey shared their impressions of the Physical Web as an emerging technology. By joining in the project, participants said they learned more about the Physical Web and its application to real-life scenarios.

Keywords: Physical Web, Bluetooth, beacons, Internet of Things, mobile devices, campus admissions

1. INTRODUCTION

The Physical Web provides a platform for seamless interaction between the virtual and physical worlds. Coupled with the increasing prevalence of the Internet of Things (IoT) and mobile computing, the Physical Web allows individuals to interact with nearby physical objects through their mobile devices ("The Physical Web - Google"). Want, et. al. refer to "devices that are part of the IoT and directly accessed, monitored, or controlled by Web technologies as the Physical Web" (Want, Schilit, & Jenson, 2015, p. 28).

This emerging technology enhances the use and understanding of various locations and process-critical objects in a variety of applications. From a deployment standpoint, the Physical Web is indiscriminate, allowing any place or thing to broadcast useful data to nearby mobile devices. Bluetooth Low Energy (BLE) beacons, which are small devices that can be embedded or affixed to an object or building, often provide this broadcast capability. Case studies using beacons for Physical Web functionality include smart airports (Namiot and Sneps-Sneppe, 2015), remote tourist locations (Lodeiro-Santiago et al., 2017) and Physical Web retailing (Lazaris and Vrechopoulos, 2014). Beacons represent the

hardware component of the Physical Web, providing proximity-based access to IoT resources. Major manufacturers of BLE beacons include BlueUp, Gimbal, and Estimote. Estimote manufactured the BLE beacons, shown in Figure 1, used in this project. An Estimote Developer Kit with four proximity BLE beacons can be purchased for $99 on estimote.com (Estimote).

This paper describes the steps involved in building a Physical Web using a collection of BLE beacons to provide a self-guided campus tour experience for visitors at Fall 2017 Prospective Students day and Spring 2018 Accepted Student's Day events at Bentley University, a business university in Massachusetts. The authors were interested in participants' first impressions of the Physical Web as an emerging technology, and after experiencing it, what applications they might imagine that could make use of the Physical Web.

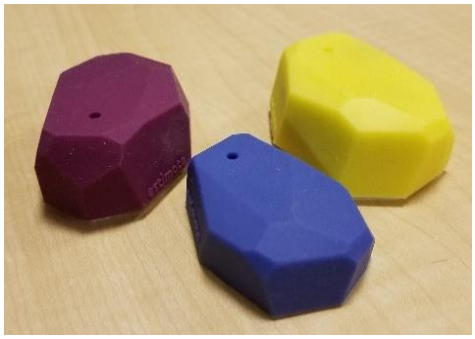

Figure 1. Estimote BLE beacons.

These research questions guided this study:

- How might experiencing beacon technology inspire participants to consider potential business applications of the Physical Web?
- What will participants learn about the Physical Web after participating in a demonstration of its capabilities?
- In what ways can the Physical Web enhance a visitor's campus tour experience?

2. APPLICATIONS OF THE PHYSICAL WEB

While the Physical Web is a new technology, its foundation rests in common and widely deployed tools and technologies like Bluetooth and web addresses (URLs). This allows for seamless integration and creates numerous areas of research and experimentation, addressing the specific use of the Physical Web alongside IoT devices and existing standards. First introduced by Google in 2014 as an improvement to existing physical-virtual interaction (i.e. QR codes), several industries have created use cases and proposed applications for the Physical Web.

Namiot et al. study an application of the Physical Web for smart cities, exploring the use of beacons for navigation and terminal information within airports (Namiot and Sneps-Sneppe, 2015, 2). In the retail industry, Lazaris and Vrechopoulos, (2014) investigate the technology as a disruption to traditional multichannel retailing, providing tailored selection and presentation of material for users' mobile devices. Lodeiro-Santiago et al. specifically harness the power of storing webpages directly on BLE beacons to present useful tourism information at remote locations without Internet access (Lodeiro-Santiago et al., 2017).

In addition to Physical Web deployments in industry, tourism, and business environments, educators have explored its applications as well. Apoorv and Mathur (2016) investigate the use of the Physical Web to enforce school attendance in India. Their research focuses on using BLE beacons attached to student ID cards, and an Android application for use on the mobile phones of school faculty. In another example, Uttarwar et al. describe an experiment with BLE beacons deployed in a library to present information regarding books and their locations within the library to proximate mobile devices (Uttarwar et al., 2017). A university library in Sweden uses BLE beacons to show information about nearby books based on the user's location among the shelves (Jergefelt, 2015).

Much of the existing literature includes the use of specific mobile applications designed to communicate with beacons. However, Google's original support for the Physical Web was based on a strictly pull-based browsing approach ("The Physical Web - Google"), where a user deliberately requests to view notifications from nearby beacons. This approach included Physical Web functionality through Google's Chrome Browser, available on both Android and iOS devices. This method contrasts with push-based notifications, implemented by Apple's iBeacon technology. Push-based notifications send messages to users' devices without their explicit request (Daradkeh and Namiot, 2015). Due to a wide array of potential abuse associated with push-based approaches, including phishing attacks, most current Physical Web platforms are pull-based.

3. BUILDING A PHYSICAL WEB ON CAMPUS

In preparation for this event, students placed 24 BLE beacons at various locations throughout campus, including academic buildings, student life centers, and athletic arenas. This section summarizes the process used to create webpages used for the open house campus tours and describes how to configure Estimote beacons to broadcast their web addresses.

Developing Campus Tour Webpages

Webpages for the Fall 2017 project were written in HTML and CSS to display static content about locations on campus, shown in Appendix 1, Figure A1. A campus web server hosted these pages.

For the Spring 2018 project, students created interactive webpages with Python and the Django web framework. Django facilitates the development of data-driven webpages. Hosted on a third-party cloud server, these dynamic pages contained short student-made videos describing each location and campus trivia questions pulled from a MySQL database. A short domain name was purchased to make webpage addresses more accessible for visitors. Examples of interactive Django pages are shown in Appendix 1, Figures A2, A3, and A4.

Configuring Beacons

Students used the Estimote app shown in Appendix 1, Figure A5 to configure each beacon with its associated address of a webpage to broadcast to nearby users' devices. Each beacon broadcast the address of a webpage using the Eddystone-URL format. Eddystone-URLs are short web addresses developed by Google, and compatible with both Android and iOS devices (Bhattacharya et al., 2016).

A Beacon-Enabled Campus Tour

At the opening programs for prospective and admitted students, an information card (shown in Appendix 1, Figure A6) placed at each seat contained instructions inviting visitors to configure their mobile devices to receive signals from BLE beacons deployed at various locations throughout campus. The University President and Director of Admissions both called attention to this project during their introductory remarks to encourage participation. Visitors used the Google Chrome mobile web browser's Physical Web feature to receive beacon signals. As an incentive, participants who, during the day, captured screenshots of any three webpages broadcast from the beacons received a free T-Shirt.

The Spring 2018 Admitted Student's Day Physical Web experience used the Phy mobile app (*The Phy Platform | We Reinvented the Universal QR Experience.*), as a replacement. Phy is a browser for the Physical Web and represents a pull-based system in which users must open the app to view nearby beacons. In Figure 2, the Library notification appears in the Phy app, shown at the left, when a visitor approaches the beacon located in the library. Tapping the Library notification card in the Phy app displays its campus tour webpage in the web browser of the visitor's mobile device, as shown in Figure 2 at the right.

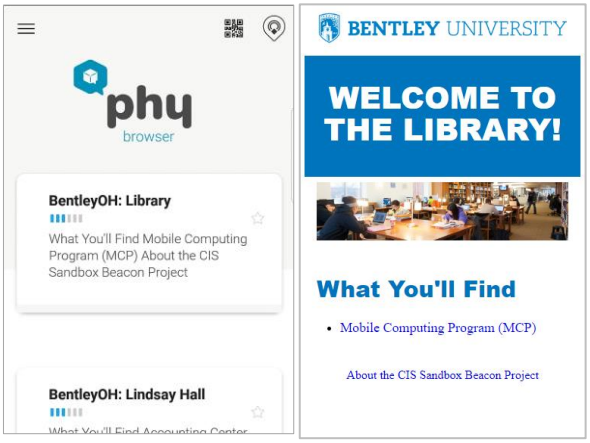

Figure 2. The Phy application running on an Android mobile device (left) and the webpage launched by tapping the link obtained from a beacon notification (right).

4. RESULTS

This section describes results of the deployment of BLE beacons at Fall 2017 Prospective Student's Day and Spring 2018 Admitted Student's Day. A limitation of this study is that a small percentage of visitors chose to participate in the activity at each open house event. Presumably, those who participated were already tech savvy, were interested in learning about the physical web, or wanted a free T-shirt. An expected outcome is that participants who claimed their T-shirts were interested in the activity and did not have difficulty setting up their mobile devices to interact with the beacons. The number of visitors who had difficulty setting up their mobile devices to receive signals from the beacons is unknown. These visitors were unable to claim their T-shirts because they could not complete the activity.

Fall 2017 Prospective Student's Day

During the Fall 2017 Open House, 700 students and their families visited campus throughout the day. Of these, 115 participants found beacons

placed at locations across campus and viewed their associated webpages. This represents a 16% participation rate among visiting families, assuming one person per family participated. The Google Analytics platform aggregated webpage activity, providing useful data points for analysis during and after the implementation. User behavior patterns, as exhibited in Appendix 1, Figure A7, provided insights into the order in which visitors encountered beacons that directed them to campus tour webpages.

This data shows that most participants started at campus locations adjacent to the welcome event (49 at Smith building and 40 at Registration) and then traveled to buildings further across campus (Smith, Quad, Library, CIS Sandbox) in their first stops. From there they went to the library, bookstore, student center, and campus center later. This is consistent with project expectations because many visitors first learned about the project during the welcome event and continued their tours at locations nearby. The 115 participants contributed to 194 unique web sessions. The data shows that 112 visitors dropped off, or closed their browser, after viewing the welcome page. From there, 36 visitors dropped off after viewing their first location's page, and 19 participants left the campus tour website after viewing the second location's page. Participants may have stopped viewing beacon pages at this point because they needed to capture only three screen shots of webpages to claim their T-Shirt prize at the end of the day.

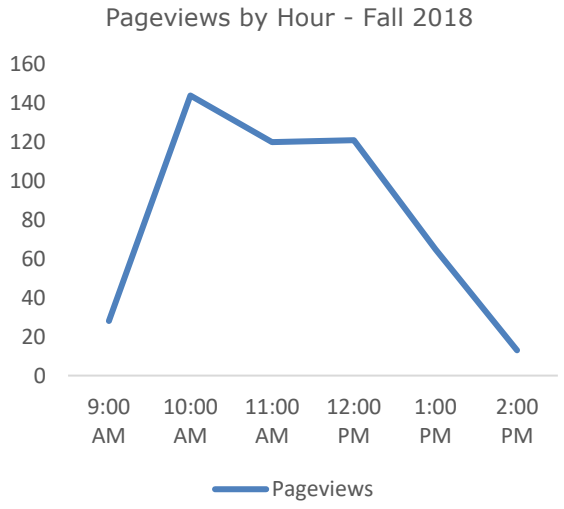

Figure 4. Pageviews by hour for Fall 2017, Prospective Student's Day.

The behavior data also provided information about page views throughout the entire open house event. This data, summarized in Figure 4, shows that peak beacon usage occurred between 10:00 AM and 12:00 PM. This usage aligns with the introductions to the project given by the University President and Admissions Director during the opening presentation beginning at 10:00 AM.

Of the 69 visitors who received T-shirts, 33 further participated in a Physical Web survey. Among respondents were 24 students, 5 parents, and 4 other campus guests. Only 18% (6/33) of survey respondents had heard of the Physical Web before participating in the beacon project. This response shows that many end users are not aware of Physical Web technology or its uses.

Survey results regarding users' first impressions of Physical Web technology included, "It is very interesting, and I believe it has the potential to grow and become amazing" and "Useful for marketing." Survey respondents also provided several ideas for real world applications of the Physical Web including parking meters, museums and statues, real-time marketing, and directions in cities.

Spring 2018 Admitted Student's Day

During the Spring 2018 Admitted Students' Day, 800 students and their families visited campus throughout the day. The Spring 2018 Physical Web implementation had 92 participants, with a total of 268 pageviews. This interaction represents an 11.5% participation rate, a 28% decrease in participation from the Fall 2017 project. Webpage viewing patterns for the Spring 2018 deployment are shown in Figure 5 and identify the same 10:00 AM to 12:00 PM timeframe as the peak beacon usage for the event.

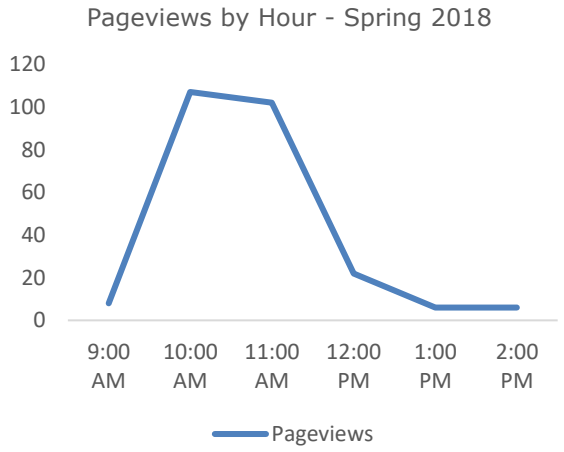

Figure 5. Pageviews by hour for Spring 2018, Admitted Student's Day.

Like the Fall 2017 project, this peak usage is correlated to the timing of the President's address and introduction to the project, as shown in Figure 4.

The Spring 2018 project featured specific paths (Academics, Campus Life, Campus Resources) to tour various university locations. Each path directed the visitor to several locations related to the path. For example, Campus Life was the most popular selection among visitors who chose a specific path. Those who followed that path visited the student center, dorms, cafeteria, and other related locations. A MySQL database connected to the tour webpages stored users' path choices. The breakdown in path choice is shown in Table 1.

Table 1. Tour path selections for Spring 2018.

Path	Number of Participants
Academics	14
Campus Life	26
Campus Resources	23
No specific path	29

Of the 92 project participants, 12 successfully completed their chosen campus tour paths and received a T-Shirt Prize. All 12 of these visitors, composed of 11 students and 1 parent, responded to a Physical Web survey for the Spring 2018 project. The survey data shows that a larger percentage had heard of the Physical Web before completing the project than had in the Fall 2017 implementation.

Attitudes and Ideas about the Physical Web

Based on the Spring 2018 event survey, 67% of respondents agree and 33% strongly agree that their participation in the beacon project enhanced their campus experience. In addition, 75% of respondents expect to use the Physical Web in their homes within five years, as shown in Figure 6. This result suggests that visitors saw the value of the Physical Web and can imagine its maturity as an emerging technology.

Survey results regarding users' first impressions of Physical Web technology included, "easily found the beacons on the phone" and "it is very useful and boosts productivity." Real world applications offered by respondents included virtual tour guides, stadiums and concerts, video games, tourism and school orientations.

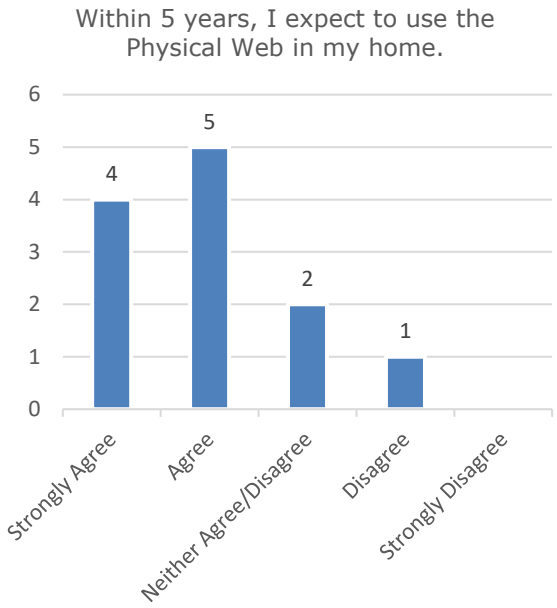

Figure 6. Participants who expect to use the Physical Web in their homes within 5 years (Spring 2018 project).

Respondents further provided feedback when asked what they learned about the Physical Web by participating in the Spring 2018 beacon project. Responses included:

- "It was my first time using it, so I learned how it can be applied to everyday events."
- "I learned about how the Physical Web network is actually set up."
- "The digital world can yield implications in the real world."

5. DISCUSSION AND CONCLUSIONS

The Physical Web was an experiment conducted by Google where beacons broadcast their web addresses to the Chrome browser. As the Physical Web evolved, its features have migrated into the Eddystone-URL protocol and were integrated to the Android platform through the Nearby and Notifications apps. As a result, Google discontinued support for the Physical Web within the Chrome mobile browser in October, 2017 (Levine).

Based on Google Analytics data and survey results, the authors compared the two Physical Web projects to answer guiding research questions and evaluate the Physical Web as an emerging technology.

The beacon-enabled campus tour inspired many creative ideas for potential business applications for the Physical Web. Survey respondents from both events suggested uses such as parking meters, real-time marketing, and virtual-physical tours, all of which are current real-world applications. Hands-on experience as end-users in a Physical Web environment successfully provided visitors with an opportunity to explore the effectiveness of this technology and to brainstorm future use cases.

The authors further considered the contexts of the two university events when analyzing the data collected at each. The Fall project was set up for Prospective Student's Day to orient students interested in attending the university. In contrast, the Spring project was designed for accepted students. Prospective students are likely to be more eager to participate in campus activities and learn as much about the campus as possible. This is a probable explanation for decreased Physical Web participation in the Spring 2018 project compared to Fall 2017. While students in the Fall were focused on academics and evaluating the university, students in the Spring had already decided Bentley was a likely fit and were more interested in campus life resources like dorms, athletic buildings, and student centers. This premise supports the "Campus Life" path being the most popular virtual tour selection among participants in the Spring 2018 project.

Using the Phy mobile application in the Spring project streamlined the participation process and is likely the reason for 0% of participants expressing difficulty setting up their devices compared to 27% in the Fall project.

Participants in the Spring 2018 project expressed several key takeaways. These included raising awareness about the Physical Web, its technology and applications; the implications of bringing together the physical and virtual worlds; and the Physical Web's application to everyday objects. Through their participation in this project, school visitors were able to learn about a new technology while also more effectively touring campus.

6. ACKNOWLEDGEMENTS

The authors acknowledge Bentley University's Office of Undergraduate Admissions for sponsoring this research project and supplying the Estimote beacons used for the Physical Web implementations.

The authors acknowledge the many Bentley University students who coded webpages, configured and placed the beacons on campus, and participated in this project.

7. REFERENCES

Apoorv, R., and P. Mathur. "Smart Attendance Management Using Bluetooth Low Energy and Android." *2016 IEEE Region 10 Conference (TENCON)*, 2016, pp. 1048–52. *IEEE Xplore*, doi:10.1109/TENCON.2016.7848166.

Bhattacharya, Debasis, et al. "Impact of the Physical Web and BLE Beacons." *Proceedings of the 5th Annual Conference on Research in Information Technology*, ACM, 2016, pp. 53–53. *ACM Digital Library*, doi:10.1145/2978178.2978179.

Daradkeh, Yousef Ibrahim, and Dmitry Namiot. *Network Proximity and Physical Web*. p. 6.

Estimote. http://estimote.com/. Accessed 6 June 2018.

Levine, Barry. *Google's "Physical Web" Loses the Chrome Browser - MarTech Today*. 13 Nov. 2017, https://martechtoday.com/physical-web-loses-chrome-browser-206732.

Jergefelt, Mikael. "An Internet of Pings: Enhancing the Web User Experience of Physically Present Patrons with Bluetooth Beacons." *Weave: Journal of Library User Experience*, vol. 1, no. 2, 2015, doi:http://dx.doi.org/10.3998/weave.12535642.0001.202.

Lazaris, Chris, and Adam Vrechopoulos. *From Multichannel to "Omnichannel" Retailing: Review of the Literature and Calls for Research*. 2014. *ResearchGate*, doi:10.13140/2.1.1802.4967.

Lodeiro-Santiago, Moisés, et al. "Improving Tourist Experience Through an IoT Application Based on FatBeacons." *Ubiquitous Computing and Ambient Intelligence*, Springer, Cham, 2017, pp. 149–60. *link.springer.com*, doi:10.1007/978-3-319-67585-5_16.

Namiot, D., and M. Sneps-Sneppe. "The Physical Web in Smart Cities." *2015 Advances in Wireless and Optical Communications (RTUWO)*, 2015, pp. 46–49. *IEEE Xplore*, doi:10.1109/RTUWO.2015.7365717.

The Phy Platform | We Reinvented the Universal QR Experience. 2018, https://www.phy.net/.

"The Physical Web - Google." *The Physical Web*.

Uttarwar, Monica Laxman, et al. "BeaLib: A Beacon Enabled Smart Library System." *Wireless Sensor Network*, vol. 09, no. 08, 2017, pp. 302–10. *Crossref*, doi:10.4236/wsn.2017.98017.

Want, R., et al. "Enabling the Internet of Things." *Computer*, vol. 48, no. 1, Jan. 2015, pp. 28–35. *IEEE Xplore*, doi:10.1109/MC.2015.12.

Editor's Note:

This paper was selected for inclusion in the journal as an EDSIGCON 2018 Meritorious Paper. The acceptance rate is typically 15% for this category of paper based on blind reviews from six or more peers including three or more former best papers authors who did not submit a paper in 2018.

Appendix 1. Additional Figures

Figure A1. Static webpage for a campus location in Fall 2017 Prospective Student's Day.

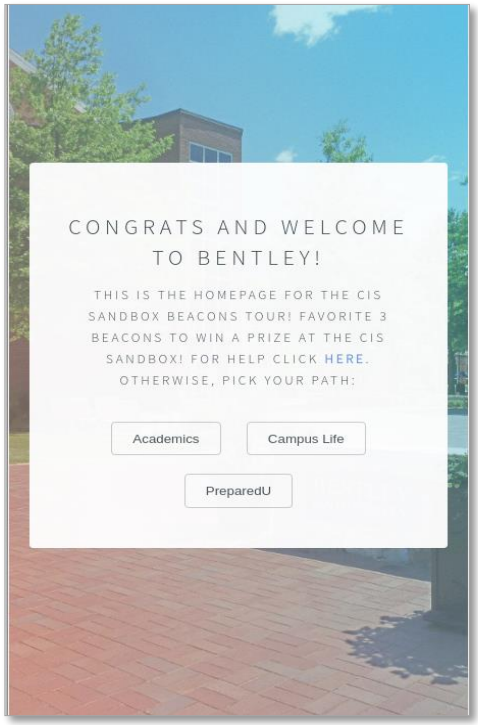

Figure A2. Path selection webpage from Spring 2018 Accepted Student's Day.

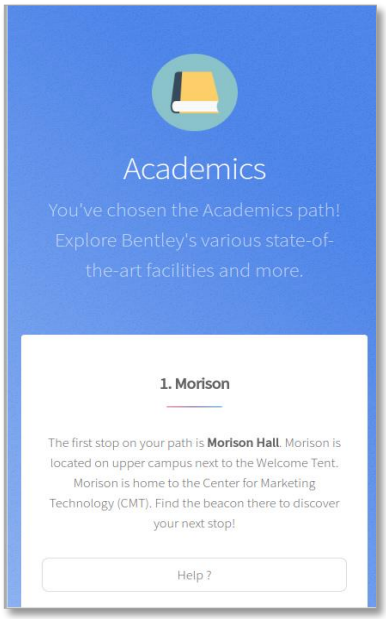

Figure A3. Sample Academics Path page. Used at Spring 2018 Accepted Student's Day.

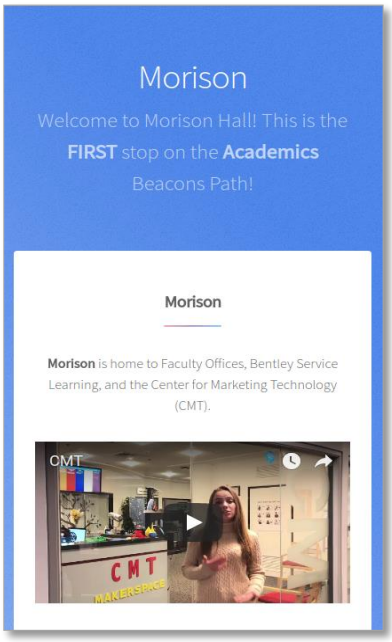

Figure A4. Sample location webpage with video. Used at Spring 2018 Accepted Student's Day.

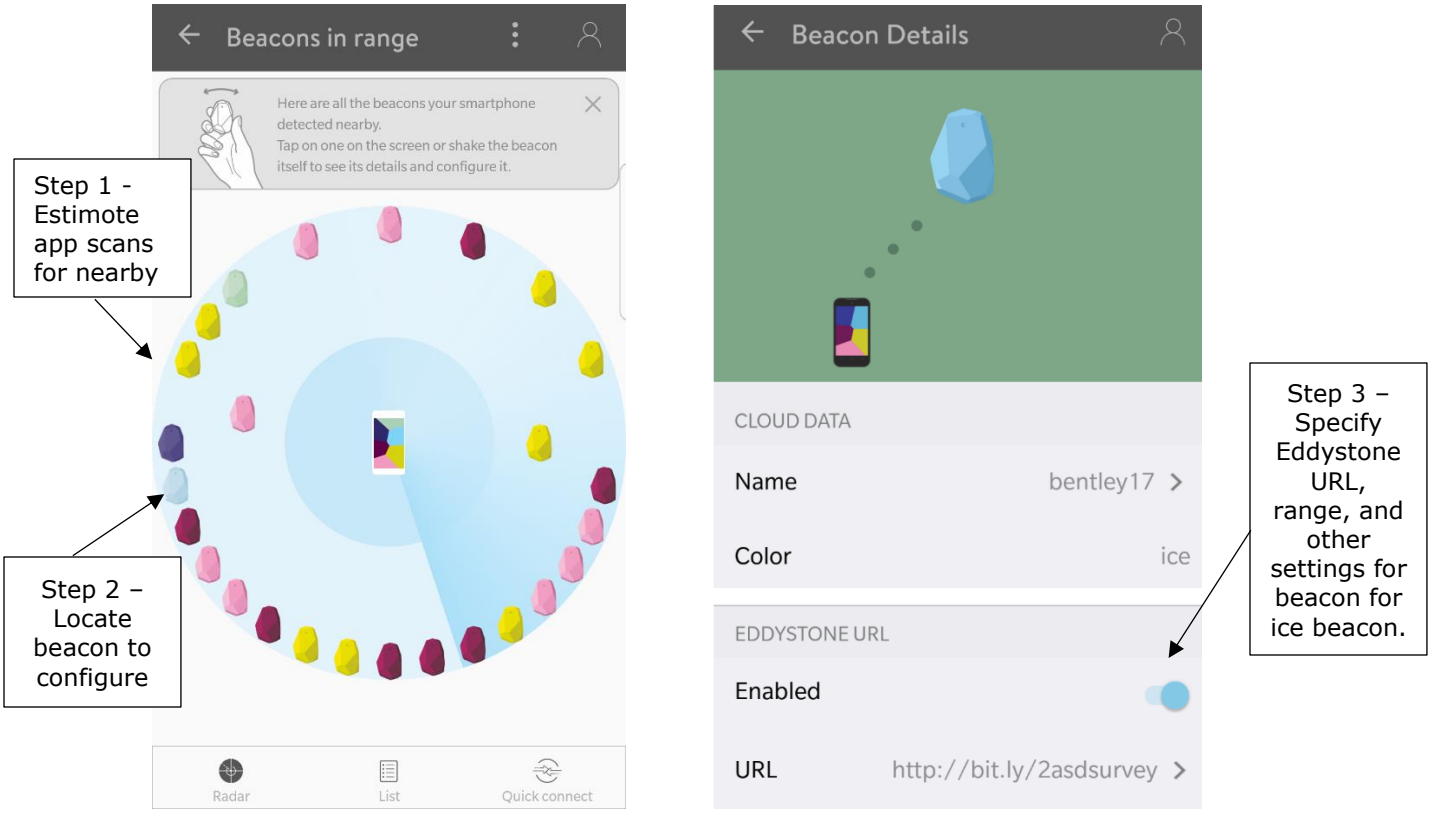

Figure A5. Configuring Beacons with the Estimote App.

Figure A6. Information card placed at each seat for Spring 2018 Admitted Student's Day, with instructions for participating in the Physical Web campus tour.

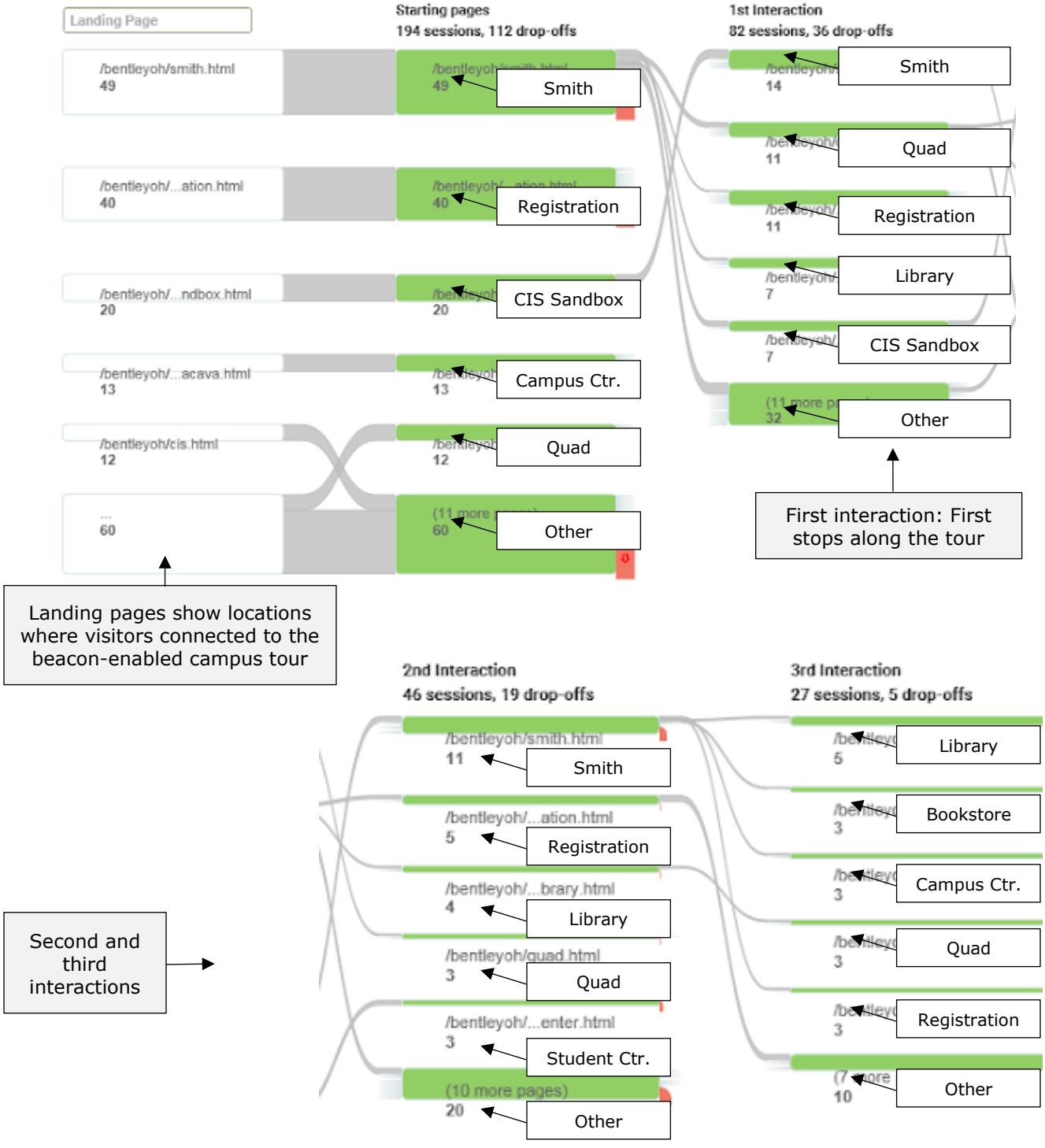

Figure A7. Google Analytics shows user behavior for the Fall 2017 Prospective Student's Day. The height of each green bar represents the number of visitors to its corresponding webpage.

Information System Curriculum versus Employer Needs: A Gap Analysis

Lori N. K. Leonard
Lori-leonard@utulsa.edu
School of Accounting and CIS
University of Tulsa
Tulsa, OK 74104

Kiku Jones
Kiku.jones@quinnipiac.edu

Guido Lang
Guido.lang@quinnipiac.edu

Computer Information Systems
Quinnipiac University
Hamden, T 06518

Abstract

Information systems (IS) curriculum review is a continuous process. Universities seek to offer content that they believe will be most beneficial to students as they begin their career. However, that content may or may not satisfy employer needs. This paper, which is part of a larger study, seeks to determine if required course content matches the desired skills from employers. Course descriptions from 221 IS curricula in AACSB accredited schools were content analyzed to determine knowledge areas and technical skills covered. The study finds that there are gaps between the current IS curriculum and the employer's desired skills. Among others, security and project management should be more prominent in the IS curriculum. A full discussion of findings is provided.

Keywords: IS curriculum, knowledge area, technical skills

1. INTRODUCTION

Information system (IS) curricula seems relatively standard to most; however, the specific courses that each institution requires can vary. This means a required course at one university is not required at another. The course may be an elective or not covered at all. Therefore, the result can be a more diverse curriculum than one might anticipate. Added to this diversity is the wants and needs of industry. Some employers may desire the curriculum that is being offered in universities for IS professionals or they may desire curriculum that will specifically help their new hires to hit the job at full speed. Programming, system methodologies and development, database design, and telecommunications come to mind when thinking of "standard" offerings; however, project management could be an unmet need at some universities as well as more specific application development.

In order to understand if IS curriculum is matching employer needs, this study is a continuation of two previous works – the first identified IS skill categories and the second identified of those skill categories which are desired most by employers. This study takes these previous findings and determines what

courses are being required by universities and how the previously identified desired skills match required course content. Ultimately, this paper identifies if there is a gap between required course content and desired employer needs. More specifically, this paper will assess whether certain "employer desired" IS topics are being neglected in the curriculum.

2. LITERATURE REVIEW

IS curriculum review has been done for many reasons. Veltri, et al. (2011) developed a framework for curriculum mapping to be used as a tool to help undergraduate IS programs determine if they are offering courses in the proper order to achieve the desired learning outcomes. The authors provide instruction on how to use the framework for continuous improvement of the program. In addition, the researchers discuss how mapping the IS curriculum can serve as an assessment tool for accreditation purposes.

Mills, et al. (2012) developed four types of IS curriculum profiles: independent, focused, adoptive, and flexible. These were developed by reviewing IS programs in AACSB schools and using the IS 2010 Model Curriculum Guidelines as a framework for their analysis. The independent profile represented programs that did not closely adhere to the 2010 model guidelines. The focused profile had programs with 30% to 70% adherence to the 2010 model guidelines. The adoptive profile had between 40% and 80% adherence. Finally, the flexible profile showed a 20% to 60% adherence to the 2010 model guidelines. The authors provided sample curriculum for each of the different profiles. Understanding the different types of profiles can help programs to see where they fit into the overall IS curriculum picture.

In addition to these reasons for studying and reviewing curriculum, researchers have also been studying the ability of IS programs to satisfy the needs of industry for quite some time. Researchers have used many methods. Some researchers have surveyed students to bring in their perspective, while others have worked with industry professionals. Each has added to the knowledge in this area. Here are just a few of those studies.

Plice and Reinig (2007) surveyed alumni of an IS program to determine what needed to be done to the program to better align it with their careers by balancing the technical vs. the business content. They found that the participants felt that bringing in heavier business content over technical content would only help them in the short-term. This type of change would actually hinder their advancement into managerial roles, they reported. The need for an emphasis in communication and teamwork skills was the result.

Aasheim, Shropshire, Li, & Kadlec surveyed Information technology (IT) managers nationwide in a longitudinal study about the skills and traits needed for entry-level IT workers (2012). The first survey was administered in 2006 and then a second in 2010. In the 2010 survey, the top 12 skills found were the soft skills - personal and interpersonal skills. While some of the soft skills were different, the top skills were also the soft skills in 2006. The top technical skills were very close as well. In 2010, they were operating systems, security, hardware, networking, and database.

Mardis, et al. (2017) surveyed course syllabi, job postings, and certifications from information technology prep programs at a state college and two universities. They were trying to understand to what extent students were prepared for technical careers. The researchers used text analysis to review these artifacts. They found that they were able to determine that the programs provided the students with the technical skills. However, the soft skills, such as critical thinking and teamwork, were a bit more difficult to determine from the learning outcomes of the syllabus.

Longenecker, Babb, Waguespack, Janicki, and Feinstein (2015) used CC2005 as a base for creating the model curricula for CIS programs. This included the knowledge areas and sub-areas for each. The authors combined knowledge areas from the Bodies of Knowledge recognized by computing professional societies. Next, they organized a group of experts with backgrounds in computing education. They surveyed this group in regards to the knowledge areas list. They found that the experts indicated Database, Information Systems Development, Systems Design, Software Requirements/Programming (including web), and Project Management based on Leadership, Team, and Interpersonal Skills were the most important to curriculum.

These previous studies illustrate that there is room to continue improving IS curriculum as well as the studies being used to assess said curriculum. The next section will provide explanation for how this study hopes to examine the employer needs versus the IS curriculum content.

3. METHODOLOGY

This study is part three of a larger, ongoing research project which intention is to determine current employer needs and align IS curriculum to better satisfy those needs. In the first part of the study, the authors conducted 10 telephone interviews with IS/IT professionals and identified a starting place for development of a questionnaire (Lang, Jones, & Leonard, 2015). They identified several skill categories using inductive category development: soft skills, consisting of intrapersonal skills and interpersonal skills; and hard skills, consisting of domain knowledge and technical skills. In the second part of the study, the authors surveyed IS professionals regarding desired entry-level skills (Jones, Leonard, & Lang, 2018). Appendix A shows the demographics information of participants. They found that employers deemed soft skills as significantly more important than hard skills for entry-level IS positions. Among the soft skills were critical thinking and willingness to learn. The most important hard skills were Microsoft Office, Security, and database.

The current study adopted a methodology used in a prior study to directly survey universities IS program curriculum as described on their websites (Yang & Wen, 2017). Starting with a list of AACSB-accredited schools, the schools reviewed were ones with programs that had computer information system (CIS)/management information system (MIS)/IS programs listed. Not assessed were schools with a focus of computer science or business or data analytics for their program.

This was a two-stage process: data collection and data mapping. Data collection took place from September of 2016 to May 2017. Data collected included: course name, number, description, credits, and required/elective. General information about each university was also collected.

Once all of the data was collected, the data mapping could begin. The authors determined only to review the required courses, as each student in the program would take these. Knowledge Areas and Technical Skills were the framework used for mapping the data (Jones et al., 2018). Each author individually went through the courses and coded them based on the framework item that best described that course. Table 1 and Table 2 provide a list of the Knowledge Areas and Technical Skills. The Rank column is the rank they had from the second part of the larger study (Jones et al., 2018). It was determined that it would be too difficult to identify if the Interpersonal/Intrapersonal Skills were being captured in the courses based on the course descriptions. Mardis, et al. (2017) had a similar issue found in their study. Therefore, only the Knowledge Areas and Technical Skills are in this study. Course descriptions for many courses identified more than one area or skill. In these cases, the authors selected the main area or skill. There were additional columns in the spreadsheet for secondary areas and skills, but those were minimal and are not a part of this paper. After each author went through the list individually, the lists were reconciled against each other.

Rank	Code	Area
1	K1	Security
2	K2	Programming
3	K3	Systems Development Methodologies
4	K4	Database Design
5	K5	Project Management
6	K6	Web Development
7	K7	IS Trends
8	K8	Enterprise Architecture
9	K9	Disaster Recovery
10	K10	Development Estimation Techniques
11	K11	Networking/Telecommunications
12	K12	E-Commerce
13	K13	Management
14	K14	Finance
15	K15	Accounting
16	K16	Marketing

Table 1: Knowledge Areas

Rank	Code	Skill
1	T1	Microsoft Office (Word, Excel, PowerPoint)
2	T2	Database/Data Warehouse/ Structured Query Language (SQL)
3	T3	Programming Languages
4	T4	Enterprise System Software
5	T5	Web Development Software
6	T6	Project Management Software
7	T7	Decision Support Systems
8	T8	Statistical Packages

Table 2: Technical Skills

4. RESULTS

At the time the research study began, there were 517 schools/colleges with an AACSB accreditation. Of these, 356 schools had some type of computing education program. After removing specialized programs and programs such as computer science, the number of schools was 249. Of these, 15% were private universities and 85% were public. Among these schools, 221 had clear designations of required versus elective courses on their websites.

Table 3 provides a list of the knowledge areas used as a main topic for the required courses sorted by frequency found in the courses across the schools. The table lists both the number of courses and the associated percentage. The main topic of 18% of the courses reviewed was programming. Database Design and Systems Development Methodologies were the main topics of 14% of the required courses. Management and Networking/Telecommunications was the main topic of 10% of the required courses. Table 4 provides a list of the knowledge areas and their frequencies of appearing as the main topic of at least one course in a program at the university. Database Design was the main topic of at least one course in 87.78% of the programs reviewed. Systems Development Methodologies was the main topic of at least one course in 85.97% of the programs reviewed. Programming was the main topic of at least one course in 75.57% of the programs reviewed. However, security was only the main topic of at least one course in 14.03% of the programs reviewed.

Code	Knowledge Area	Frequency	Percentage
K2	Programming	288	18%
K3	Systems Development Methodologies	228	14%
K4	Database Design	226	14%
K13	Management	166	10%
K11	Networking/ Telecommunications	161	10%
K5	Project Management	114	7%
K6	Web Development	59	4%
K8	Enterprise Architecture	53	3%
K7	IS Trends	41	3%
K1	Security	34	2%
K12	E-Commerce	21	1%

Table 3: Frequency of Knowledge Area as Main Topic of Required Course

Code	Knowledge Area	Frequency	Percentage
K4	Database Design	194	87.78%
K3	Systems Development Methodologies	190	85.97%
K2	Programming	167	75.57%
K11	Networking/ Telecommunications	137	61.99%
K13	Management	107	48.42%
K5	Project Management	101	45.70%
K6	Web Development	51	23.08%
K8	Enterprise Architecture	48	21.72%
K7	IS Trends	33	14.93%
K1	Security	31	14.03%
K12	E-Commerce	20	9.05%

Table 4: Frequency of Knowledge Areas Found in Programs

Table 5 provides a list of the technical skills emphasized for the required courses sorted by frequency found in the courses across the schools. This table also provides both the number of the courses and the associated percentage. Programming languages was the emphasis of 18% of the required courses reviewed. Database/Data Warehouse/SQL was the emphasis of 14% of the required courses. These mirror what was in the knowledge area. Many courses had both aspects of theory and hands on learning. Table 6 provides a list of the technical skills area and their frequencies of appearing as the emphasis of at least one course in a program at the university. At least one course in 87.78% of the programs emphasized Database/Data Warehouse/SQL. At least one course in 75.57% of the programs emphasized programming languages. Less than half of the programs, 47.51%, had at least one course emphasizing Project Management Software. Towards the bottom of the list was Microsoft Office (Word Excel, PowerPoint). Only 8.14% of the programs had this in at least one course.

Code	Technical Skills	Frequency	Percentage
T3	Programming Languages	289	18%
T2	Database/Data Warehouse/ SQL	227	14%
T6	Project Management Software	116	7%
T5	Web Development Software	59	4%
T4	Enterprise Systems Software	47	3%
T1	Microsoft Office (Word, Excel, PowerPoint)	19	1%
T7	Decision Support Systems	14	1%
T8	Statistical Packages	12	1%

Table 5: Frequency of Technical Area Emphasized in Required Course

Code	Skill	Frequency	Percentage
T2	Database/Data Warehouse/SQL	194	87.78%
T3	Programming Languages	167	75.57%
T6	Project Management Software	105	47.51%
T5	Web Development Software	51	23.08%
T4	Enterprise System Software	43	19.46%
T1	Microsoft Office (Word, Excel, PowerPoint)	18	8.14%
T7	Decision Support Systems	13	5.88%
T8	Statistical Packages	11	4.98%

Table 6: Frequency of Technical Skills Found in Programs

5. DISCUSSION

In reviewing the results of the study, it is clear that there are gaps between what industry professionals rank as important entry level skills (Jones et al., 2018) and where the current IS

curriculum stands. This discussion will begin by addressing areas where there seem to be gaps, and follow with areas that seem not to have gaps.

In the knowledge area, the industry professionals ranked Security as the top desired skill for entry-level employees in the information systems field (Jones et al., 2018). However, this study shows that only 14.03% of the programs have a required course in security. Many of the programs did have a course in security listed as an elective. However, this would not ensure that every student graduating from that program would acquire the skill seen as important by the industry professionals. Universities will want to add or alter their curriculum to incorporate security as a required course.

Project Management is another knowledge area that was in the top five desired skills of entry level IS employees. However, this study found that only 45.70% of the programs had at least one course with project management as the main topic. For the technical skills area of Project Management Software, only 47.51% of the programs had emphasized project management in at least one course. As the importance of project managers in IS projects continues to be shown (Venkatesh, Rai, & Maruping, 2017), project management knowledge will continue to be seen as an asset to graduates.

Microsoft Office (Word, Excel, and PowerPoint) was the number one ranked technical skill that the industry professionals desired in entry level IS employees. However, this study showed that only 8.14% of programs had a required course that emphasized the use of the programs. Most courses would utilize this software in some way throughout the course. However, most did not have set objectives to teach the software. Most likely, universities are making assumptions that students are coming in with certain skill sets and are no longer offering these basic computer literacy courses. However, perhaps this is not an assumption to make. Many high schools may not be using these specific software packages, even though they are utilizing something similar. Programs may find it worth at least some portion of a course dedicated to ensuring their students master these skills.

Networking/Telecommunications appears in 61.99% of the programs reviewed as a required course. Yet, it was listed as ranking 11th in the desired skills for entry level IS employees by industry professionals (Jones et al., 2018). While this is clearly on the radar of the industry professionals, it appears less important than some of the topics that are not appearing in many of the programs are. Perhaps if universities are looking for ways in which they can move courses around in order to accommodate courses such as Security and/or Project Management, moving the Networking/Telecommunications course to an elective and one of these to a required may be the answer.

Web Development and Enterprise Architecture are two other areas that receive little attention in the required curriculum but employers indicated a desire for these skills. While these areas are not in the top five for knowledge areas, they are in the top five for technical skills desired. However, the percentage of current IS curriculum that offers these as required courses is small (technical skills: 19.46% Enterprise System Software and 23.08% Web Development Software). Web development is a skill that is not going away and employers will continue to need it. Enterprise system software development can vary by industry but is definitely a desired technical skill as it ranked number four in the previous study.

Other desired knowledge and technical skills were heavily emphasized in the IS curriculum as expected such as programming, system development methodologies, and database design. These are standard and expected skill sets offered in the curriculum and desired for new hires in industry. It was a positive to see no gap in these areas between offerings and desires.

The intrapersonal/interpersonal skills, also indicated as important skills for entry level IS employees, still needs researching (Jones et al., 2018). These were difficult to determine if they were taught by reviewing the course descriptions, which were shown on the websites of the programs. Future researchers will want to survey programs to determine if these skills are in the programs. That will provide a guide for universities to know where their gaps are in regards to what the industry professionals rank as important.

Consider the limitations of this study when reviewing the recommendations. First, this study focused solely on AACSB-accredited schools. It is possible that IS curricula at non-AACSB accredited schools offer a different set of Knowledge Areas and Technical Skills. Second, the study focused solely on required courses. As a result, this analysis disregarded the content provided in elective courses. Given the strong emphasis on flexible curricula coupled with academic advising, it is likely that students at

many AACSB-accredited universities learn additional Knowledge Areas and Technical Skills not identified in this study of required courses. Third, this study relied solely on course descriptions when determining the coverage of Knowledge Areas and Technical Skills. While course descriptions should be a good indicator of course content, it is possible that certain minor content is not captured in the course descriptions and thus not in our study. Lastly, this study used a framework of Knowledge Areas and Technical Skills that were previously developed (Jones et al., 2018). As a result, this study did not address other potential content areas, including interpersonal skills.

6. CONCLUSION

IS curriculum and employer desires may not always meet eye-to-eye. This study has identified knowledge and technical areas where there are gaps in what universities are teaching, and what IS Professionals identified as skills that new hires need. More specifically, this study finds that universities offer programming, systems development methodologies, and database design in the curriculum and these are desired by employers. However, there are several areas that receive less attention in the required IS curriculum, such as security, project management, web development, and enterprise architecture, yet, employers desire them. IS curriculum must constantly evolve in order to produce students that meet the needs of industry. This study is part of a larger study which seeks to examine these gaps in the curriculum and offer recommendations to Universities seeking to improve their IS course offerings.

7. REFERENCES

Aasheim, C., Shropshire, J., Li, L., & Kadlec, C. (2012). Knowledge and Skill Requirements for Entry-Level IT Workers: A Longitudinal Study. *Journal of Information Systems Education, 23*(2), 193.

Jones, K., Leonard, L. N. K., & Lang, G. (2018). Desired Skills for Entry Level IS Positions: Identification and Assessment. *Journal of Computer Information Systems, 58*(3), 214-220.

Lang, G., Jones, K., & Leonard, L. N. K. (2015). In The Know: Desired Skills for Entry-level Systems Analyst Positions. *Issues in Information Systems, 16*(1), 142-148.

Longenecker, B., Babb, J. S., Waguespack, L., Janicki, T., & Feinstein, D. (2015). Establishing the Basis for a CIS (Computer Information Systems) Undergraduate Degree Program: On Seeking the Body of Knowledge. *Information Systems Education Journal, 13*(5), 37-61.

Mardis, M. A., Ma, J., Jones, F. R., Ambavarapu, C. R., Kelleher, H. M., Spears, L. I., & McClure, C. R. (2017). Assessing alignment between information technology educational opportunities, professional requirements, and industry demands. *Education and Information Technologies*, 1-38.

Mills, R. J., Velasquez, N. F., Fadel, K. J., & Bell, C. C. (2012). Examining IS curriculum profiles and the IS 2010 model curriculum guidelines in AACSB-accredited schools. *Journal of Information Systems Education, 23*(4), 417.

Plice, R. K., & Reinig, B. A. (2007). Aligning the information systems curriculum with the needs of industry and graduates. *Journal of Computer Information Systems, 48*(1), 22.

Veltri, N. F., Webb, H. W., Matveev, A. G., & Zapatero, E. G. (2011). Curriculum mapping as a tool for continuous improvement of IS curriculum. *Journal of Information Systems Education, 22*(1), 31.

Venkatesh, V., Rai, A., & Maruping, L. M. (2017). Information Systems Projects and Individual Developer Outcomes: Role of Project Managers and Process Control. *Information systems research*.

Yang, S. C., & Wen, B. (2017). Toward a cybersecurity curriculum model for undergraduate business schools: A survey of AACSB-accredited institutions in the United States. *Journal of Education for Business, 92*(1), 1-8.

Appendix A

Demographic Information of IS Professionals (Jones et al., 2018)

Demographic Variable		**% of Respondents**
Industry	IT	40%
	Healthcare	8%
	Financial Services	9%
	Consulting	10%
	Manufacturing	3%
	Chemical	1%
	Utilities	3%
	Education	3%
	Other	24%
Company Size (Number of Employees)	Less than 100	13%
	100 – 500	4%
	501 – 1000	1%
	1001 – 2500	4%
	2501 – 5000	13%
	5001 – 10,000	9%
	Over 10, 000	57%
Size of IT Staff	Less than 5	9%
	5 – 9	1%
	10 – 50	8%
	51 – 100	9%
	101 – 150	4%
	More than 150	70%
New IS Positions Per Year	Less than 5	26%
	5 – 9	9%
	10 – 19	8%
	20 – 29	13%
	30 – 49	3%
	50 or more	41%
Company Location	CA	1%
	CO	1%
	CT	37%
	IL	1%
	KS	1%
	MA	4%
	MN	1%
	NJ	5%
	NY	4%
	NC	3%
	OK	24%
	PA	4%
	SC	3%
	TX	11%
Job Title (Respondent)	CIO/VP, IS Director	10%
	IS Manager/Consulting Manager	11%
	Project Manager/Leader	10%
	Systems Analyst/Programmer, IS Consultant	29%
	Human Resources/Recruiter	3%
	Other	37%
Years of Experience (Respondent)	Less than 5	33%
	5 – 9	35%
	10 – 15	14%
	16 – 20	1%
	20 or More	17%

The Soul of the Introductory Information Systems Course

Minoo Modaresnezhad
modaresm@uncw.edu

George Schell
schellg@uncw.edu

Cameron School of Business
University of North Carolina Wilmington
Wilmington NC 28403

Abstract

The introductory course in information systems plays an important role in a business school's curriculum. The soul of that course should be information and systems. The introductory course is where all business students, information systems majors and non-majors alike, learn why information and the technologies associated with it are driving organizations' strategy, processes, and successes. The content in that course and how it is taught profoundly shape students' perceptions of our discipline. Over the years there has been a reduction of focus on information and systems and increased focus on the latest technological innovation. We must get back to teaching how information systems drive decision making.

Keywords: Information Systems, Curriculum, Decision Support, Introductory Course

1. INTRODUCTION

Information systems are crucial to organizations. Yet a recurring theme in our journals is that non-information systems colleagues, students, and even the business community do not understand nor appreciate the importance of information systems (IS) as a discipline in a business school. A solution might be to write more journal articles or to make more presentations at conferences but those outlets cater to the people already in our discipline. A better strategy would be to engage an audience that needs to be informed about the role of information systems in organizations who can become ambassadors or at least supporters of information systems in business schools and their organizations. That audience is the student body and by extension the organizations which hire the students.

This is a long-term strategy and it requires a willingness to accept that most of students in our introductory classes have no intention of becoming information system majors. It requires that the faculty who teach the introductory course realize that student understanding of information systems in organizations is important for the acceptance of our discipline. As these students enter and progress through their careers, they will be the ones looking to information and the supporting technology to help reach organizational goals. Managers and knowledge workers are decision makers and decisions require information processed by systems.

Recruiting information systems majors should not be the only reason or even the major reason for choosing the content within the introductory information systems course. We must focus on the information systems theory and practice that should be known by every business student. A student's interest in the materials in the course will lead him or her to become an information system major. Students of all majors need to be given the foundation knowledge and then carry the appreciation for information systems into

their careers. The number of information systems majors seems to increase and decrease with the economy and job market prospects but the role of information systems in organizations is a constant.

It is important to distinguish information systems from information technology in the introductory course. AACSB standards refer to information systems and information technology as separate subjects (AACSB p. 9). These two subjects can certainly be taught within the same course but it would be prudent to make information systems the dominate subject and information technology the subordinate subject. Too many students and non-information systems faculty have the false impression that the course is just about technology.

With information technology bleeding into so many business disciplines it is easy for a student to believe an information systems course is simply a way of being taught the technology that will be most useful to his/her major. A non-information systems faculty member may believe that he/she can teach information systems because he/she teaches an aspect of information technology in his/her discipline classes. The distinction between information systems and technology is fuzzy to too many students and faculty.

That is our fault as information systems faculty. The information systems class should clearly explain the three distinct disciplines of information systems, information technology, and computer science. Although a Venn diagram would show areas of overlap between two and even all three of these fields, there are still importantly distinct areas which do not overlap and this is essential knowledge to pass to the students. One of the important distinctions is that information systems majors must consider the economic value of an information system to decision making and information systems in general to the organization.

> "The IS discipline contributes significantly to several domains, including business and government. Information systems are complex systems requiring both technical and organizational expertise for design, development, and management. They affect not only operations but also the organization's strategy." (Topi et al. 2010, p. 1)

The purpose of this paper is to begin a discussion and/or rekindle old discussions to define the soul of the introductory information systems course. Exactly what are the most important concepts we wish every business school graduate to know about information systems? The IS 2010 curriculum guidelines correctly note that information systems are not solely within the domain of business schools (Topi et al. 2010, p. 9). And the removal of a course focusing on personal productivity tools from 2002 guidelines (Ives et al, 2002) was a welcome relief. But what are the guidelines for the introductory course that meet the needs of today's students?

The description in the 2010 guidelines for the focus of the course lists the key components of an information system (people, software, hardware, data, and communications technologies) and not information itself or how it is used. Neither a theory of information nor decision support is in the description (Topi et al. 2010, p. 36). This is at odds with the stated guiding assumption that "IS professionals must have strong analytical and critical thinking skills to thrive in a competitive global environment" (Topi et al. 2010, p. 8).

2. THE SOUL

What is meant by the "soul" of the introductory information systems course? We in the information systems discipline have too many diverse/conflicting opinions about how to answer that question. The last ACM/AIS curriculum guidelines were published in 2010 - eight years is a long time in a field that changes so rapidly. While the next decision on guidelines is being produced, we need a vigorous debate about the purpose of the introductory course. There will certainly be opposing opinions within the information systems community and those opinions should be expressed. The remainder of this section lays out the framework of how a fruitful discussion might play out. This is offered as a starting point for the discussion.

A Pareto-optimal solution to guidelines for the introductory course is not necessary. Possibly not even desirable. A consensus of the majority is a worthy goal. This statement may not be palatable to members of our discipline or to a committee that has to publish curriculum guidelines. But if the objective of a discussion about the content of the introductory course is to produce a course that will impress the importance of the information systems discipline upon every business school student then we must be willing to accept that at least some in our discipline will feel the changes make the course worse.

3. BEGIN AT THE BEGINNING – DECISION SUPPORT

It is the contention of this paper that decision support is the fundamental bedrock on which information systems rest in an organization. Three seminal works form the basis of this contention and all clearly point to decision making for the importance of information systems.

Herbert Simon passionately researched political science and psychology in addition to information sciences. In 1978 he won the Alfred Nobel Memorial Prize in Economic Sciences in large part to his research that shaped the theory of decision-making processes in organizations. He won the A. M. Turing Award in 1975 (along with Allen Newell) for his contributions to the field of artificial intelligence and other fields. Part of his brilliance was to understand not only how decisions were made but the role of information in decision making and the potential of computer-based systems to enhance the decision-making process.

His theory for decision making required three parts (Simon 1959). First, an entity is required that can make a decision. This may sound trivial but it has important implications for students in business schools who use information technology. A "decision" could be defined in a manner that would not require a human. A thermostat can "decide" to use heated or cooled air to adjust the temperature in the area controlled by the thermostat. With artificial intelligence we are seeing decision making capabilities, or at least a resemblance of that capability, being introduced into non-humans.

The second part was a set of possible choices. The choices had to be discrete so that choices could be compared and evaluated as separate objects. Before computer technology became available the number of choices in the set might be very limited simply because of the computational requirements to evaluate each choice.

Limiting the number of choices in a choice set is rational for a decision maker because of the associated information connected to each choice. Consider a situation where a decision maker makes a choice (i.e. a decision) about staffing levels which is then followed by an event occurring. For example, the manager may have the choices of (1) adding a second shift for production, (2) committing to overtime pay for current employees, (3) keeping staffing levels unchanged, or (4) reducing staff by 25%. Assume three market events may occur once the manager's decision is made; (1) the demand for the company's product rises by 20%, (2) stays unchanged, or (3) drops by 10%. For each possible choice added (such as adding a second shift) the manager must determine (1) the economic impact an event (e.g. demand dropping by 10%) has on profits and (2) the probability that that event will occur. In our scenario each added choice to the choice adds 3 more economic impacts to be determined as well as an additional probability associated with the new choice. Depending upon the number of possible choices and the number of events that can occur after a choice the additional information added to the decision process could become substantial.

In our paper we will limit the number of choices in a choice set to some number that a decision maker might reasonably consider in a short period of time – such as two weeks. For example, instead of have choices be $1M, $900,000, $800,000, $700,000, etc. for the amount of a marketing budget we may limit the choices to 3 by executive fiat as $1M, $500,000, and $0. Readers who wish to learn more about the theoretical underpinnings of choice sets can find additional information in (Reutskaja, et al, 2011), (Karni, 2013), and (Manski, 2017).

The third part of Simon's theory was a keen insight for his time – the selection of a choice could be made upon either optimizing or satisficing the value/utility gained from decision making. The ability to have a satisficing choice brings the ability to apply managerial judgement to the selection process. A simple computer program would lack the ability to make judgements about satisficing solutions.

Peter Keen and Michael Scott Morton extended Simon's concept when they began to explore effectiveness as opposed to efficiency for dealing with semi-structured decision making (Keen and Scott Morton 1978). A particularly useful line of research since many knowledge workers engage in decision making where mathematically optimal solutions may not exist or where they may be impractical. The title of their second chapter is particularly insightful, "Management, Information, Systems, and MIS." It quickly establishes that what we commonly call "MIS" is actually an amalgamation of ideas and concepts. They call the phrase "management information systems" an example of something that means different things to different people with no generally accepted definition recognized by those working in the field. They are as correct today as they were in 1978.

Academics in the information systems discipline might argue that there is most certainly a widely accepted definition of information systems. But we are looking inward, teaching our doctoral candidates, who are in turn going out to teach other doctoral students as well as undergraduates about information systems. Our observed and reported experiences with non-information systems faculty and people in industry document the lack of a generally accepted definition. The ramification of not having a generally accepted definition is that the information systems discipline is not highly valued.

The third seminal work is by Ralph Sprague. He correctly described decision support systems as those drawing from and interacting with other information systems in an organization to support the work of managers and knowledge workers in organizations (Sprague, 1980). He also observed that decision support systems were no longer the sole purview of information systems professionals. This acknowledges that non-information systems majors have a need to study information systems in their business school coursework.

Sprague's models often related to a triad of dialogue, data, and models for the systems he envisioned. This reinforces the theme that information systems are not only for IS professionals since they presumably would not need the dialogue feature. Berinato (Berinato, 2019) also writes about the need to effectively communicate between decision makers and analytics professionals in order for projects to succeed. The concept of an intuitive user interface is the natural progression of Sprague's model.

Hugh Watson (2018) revisited Sprague's framework for developing decision support systems. Watson suggested that the Sprague's model is still relevant today. He emphasized the idea that information and systems are about decision support and the practice of decision support continues to evolve considering the new technological development such as enterprise data warehousing, real-time data warehousing, big data analytics, etc. (Watson, 2018). Sprague's work is robust enough to embrace emerging technologies.

There is a common theme that connects these seminal works – decision making. All of the authors understood and wrote about the technology behind information systems but it was (a) information in a digital format that (b) could be processed, assimilated, and reported that led to (c) a choice among competing possible actions that could be taken by a decision maker. Satisficing criteria for the choice among competing decisions, the effectiveness of the selected decision, and technology all go into supporting the managerial judgement. The keystone of the decision-making process is managerial judgement.

Figure 1 represents how technology both supports digitized information while being the repository for digitized information. Flowing upward, the digitized information feeds the decision maker, a choice set, and processes/information systems (models) described by Simon. Managerial judgement sits atop this flow and is the final process guiding decision making. Advances in the technology, the emergence of (big) digitized data, and the wide range of data and knowledge sources have increased the importance of strategic information systems that support managerial judgment and decision-making processes.

The phrase "decision support systems" is a classical (possibly old fashioned) term that is well known and understood but it may not be alluring to students. The phrases "decision analytics" and "business analytics" are beginning to be used and they might be easier terms that more students in business schools can understand (Power 2013; Vob 2014; Rodammer, et al. 2015; Schiller, et al. 2015; Vob et al. 2017; Arunachalam, Kumar, & Kawalek, 2018). Research shows that businesses that use analytics strategically at the managerial level often outperform their competitors (Frederiksen, 2009; Harris, 2008).

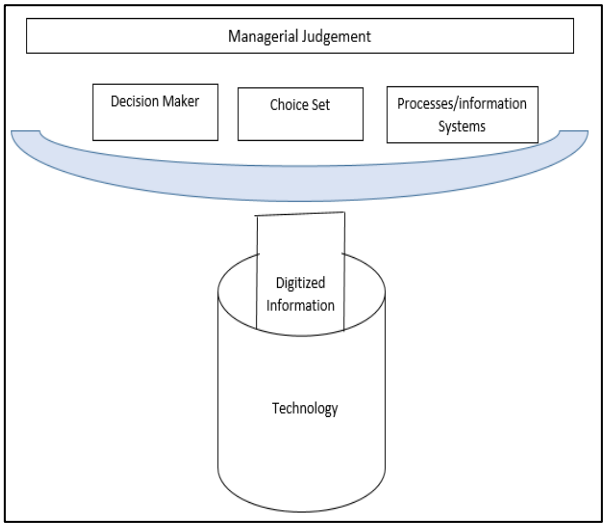

Figure 1 - Technology as the Support and Repository of Information

4. LISTENING TO CRITICS

It is fitting to look at criticism of the information systems discipline when we search for its soul. Some of that criticism is valid. Russell Ackoff was an early critic of those who viewed of the role of technology as supreme in a field where quantitative decisions for managerial issues seemed to rely too much on information technology and not enough on the decision to be made. He blamed the researchers of the era (Ackoff 1967, p. 147). His admonition is important since it forces academics in information systems to realize what is taught in our courses is more important than the technology used by the discipline.

> "Enthusiasm for such systems is understandable: it involves the researcher in a romantic relationship with the most glamorous instrument of our time, the computer."

Ackoff went on to foreshadow the era of big data. He argued that managers do not suffer from a lack of information but from too much information. Digitized information comes in from many sources and the people acting upon the information can become overwhelmed by the volume especially if they wish to consume detailed transactional data. He argues that decision makers have too much transactional data while actually lacking aggregated information that is relevant to the decision to be made.

We believe that one of Ackoff's criticisms - that decision makers lack relevant data - has been corrected in the years following Ackoff's paper. However, his criticism concerning our fascination with technology is still relevant to why the discipline of information systems does not get the respect it deserves. He feels our discipline creates information systems where we are too concerned with how easily the system can be used and that we are not concerned enough with the user's understanding of how information drives the system. If students leaving the introductory information systems course believe that it is a course about tools and technology then Ackoff's criticism still rings true 50 years after his paper was published.

Nicholas Carr is a more recent critic of our field. His premise is that information technology no longer matters to business organizations (Carr 2003). It is profoundly disheartening that an article would be published in the *Harvard Business Review* which perpetuated the misconception that our field is information technology.

There were many papers published in rebuttal to Carr that made reasoned arguments showing information systems are not equivalent to information technology and that while technology may be trending towards a commodity that information and the systems using information are still growing, evolving, and driving the success of many organizations. We may have convinced ourselves but too many students, colleagues, administrators, and business leaders still believe Carr was correct. It is through the introductory information systems course that we can change this perception in all business students.

5. LEARNING FROM PAST SOUL SEARCHING

There was a steep decline in the number of information systems majors between the 1990s and the early 2000s. This sparked some introspection among IS faculty (Granger et al 2007). There was even a fear that information systems might not stay as a component of the core body of knowledge in AACSB standards. In response to that perceived threat 40 leaders in the information systems academic community coauthored a paper on what every business student should know about information systems (Ives et al, 2002). Their reasoned argument stated that we are in the fifth wave of capitalism, the digital/knowledge economy, and the impact of information systems on industry makes it imperative that students understand the role of information systems.

Ten key concepts were proposed for students with learning objectives attached to each concept. The concepts focus on business not technology. Technology is the tool, the artifact used for information system development. Ives, et al. feared that a primary focus on the technology might lead to information technology expertise being a subject taught outside the business school. Such a situation would result in organizations unable to find sufficient numbers of information systems professionals with the business skills needed to be analysts, project managers, and senior information officers (Ives et al., 2002, p. 472).

Firth and his coauthors (Firth, Lawrence, & Looney, 2008) recognized this same enrollment decline. They even referenced that the president of AIS noted the crisis in his 2006 address at ICIS. The important contribution of the Firth paper was finding that the information systems class should

teach information systems – not information technology nor computer science. If we want to attract students to information systems and gain the non-information systems majors as information systems supporters then subject materials in the introductory class need to hold the interest of all students (Ferratt et al., 2010; Zhang 2007). Being the person making decisions in an organization will be interesting to all students.

Hershey (Hershey 2003) also takes issue with the 2010 guidelines for the introductory information systems course. Effective, impactful application of information systems in business processes and operations will stimulate IS and non-IS majors alike. But a fixation on technologies will reinforce the perception by non-IS majors that not much in the introductory course is important to the non-IS major.

Topic	Faculty	Students
System security	1	1
IS in organizations	2	5
Valuing IS	3	9
IS development	4	7
Globalization	5	4
IS infrastructure	6	6
IS ethics	7	2
The internet and WWW	8	3
IS components	9	10
Business intelligence	10	8

Table 1 - McCoy, Everad, and Jones IS Topic Importance in Rank Order

Faculty teaching the introductory course try to impress their opinions of the relative importance of the topics in the course. However, student perceptions do not agree with the IS faculty perceptions of the rank order importance for the topics in the 2010 guidelines for the introductory course (McCoy et al. 2015). The topics were condensed to the 10 by McCoy, Everad, and Jones as shown below in Table 1. Since the faculty control what is taught in the introductory course and have the ability to impress their views onto students in the course, wouldn't it seem intuitive that student and faculty rankings would be similar?

The disparity in rank order between faculty and students for "valuing IS" and "the internet and WWW" is very striking. If the introductory course is where IS faculty wish to impress all business majors with the importance of information systems for the businesses where they will work and for their personal careers then we are failing. Pay particular attention to "Valuing IS" and "The internet and WWW."

6. TECHNOLOGY STILL NEEDS TO BE TAUGHT

This paper does not mean to imply that information technology should not be taught in the introductory class. It has an important role to play but that role is subordinate to the focus on information systems and decision making. Like Ackoff noted, the computer is the most glamorous instrument of our time. There will always be some new technology that bursts upon the scene to capture our attention. If we constantly chase the newest technology it would easily overwhelm all of the time we have to teach our students.

The power of computing doubles every 18 months and although predictions of the death of Moore's Law seem to pop up occasionally it has held since the mid-1960s (Denning and Lewis 2017). Databases, spreadsheets, and other software can be easily used to support an understanding of decision making. For example, teaching students how to perform mathematical operations on cells in a spreadsheet adds little value to the introductory information systems class since those skills were probably taught to the students while they were in high school. However, using the SOLVER feature in Microsoft Excel to optimize a problem expressed in algebraic form or using the VLOOKUP feature to act as a primitive database query would further the concepts more appropriate for the introductory class.

Do you wish to ignite a discussion concerning personal information privacy? Then have your students go to MYACTIVITY.GOOGLE.COM and sign in with their GMAIL account. The discussion can easily spread to metadata and how metadata might be used to predict specific behaviors. How can privacy be retained and reduce the threat of private data be misused? Have students visit the TED Talk "A New Way to Stop Identity Theft" (Birch 2012). The resulting discussion brings you back to a solution to Ackoff's complaint that managers have too much information and not enough of the information they need. This use of technology to encourage discourse embraces the constructivist learning model (Leidner and Jarvenpaa 1995) but since the pedagogy for teaching the introductory course is beyond the scope of this paper further arguments will be left to later research.

7. CONCLUSION

A discussion is needed to decide the important content of the introductory information systems course in business schools. This paper argues that the two flaws of the current introductory course are (a) focusing too much on recruiting IS majors and not enough on educating non-IS majors and (b) too much emphasis on technology itself and not enough on how technology enhances decision making. IS faculty must accept the fact that most careers in business organizations are filled by non-IS majors and the number of IS majors must reflect that fact. Creating a commonly understood meaning for information systems in organizations and the value brought to organizations by those information systems will create the respect for the information systems discipline that seems to be lacking at this time.

This paper lays out an initial argument for the content in the introductory course based upon the early foundations for the information systems field. It looked at past criticisms of the discipline as well as some soul searching from within the IS discipline itself. It argues that information technology should be taught within the context of how it supports information systems and processes and not as an equal partner to information systems. These are debatable points. A robust discussion to the content of the introductory course will help shape a concept of the IS discipline which can be clearly understood by students, non-IS faculty, administrators, and the business community. That will be a major step towards the information systems discipline achieving its rightful acknowledgement of a driving, valuable discipline for all business majors to study. Please join the discussion.

8. REFERENCES

Ackoff, R. L. (1967). Management misinformation systems. *Management Science*, 14(4), 147-156.

Arunachalam, D., Kumar, N., & Kawalek, J. P. (2018). Understanding big data analytics capabilities in supply chain management: Unravelling the issues, challenges and implications for practice. *Transportation Research Part E: Logistics and Transportation Review*, *114*, 416–436. http://doi.org/10.1016/j.tre.2017.04.001

Berinato, Scott (2019). Data science & the art of persuasion. *Harvard Business Review*, 2019(January-February), 127-137.

Birch, D. (2012). A new way to stop identity theft. TED Talk, Retrieved from https://www.ted.com/talks/david_birch_identity_without_a_name.

Carr, N. G. (2003). IT does not matter. *Harvard Business Review*, 81(5), 41-45.

Denning, P. J. & Lewis, T. G. (2017). Exponential laws of computing growth. *Communications of the ACM,* 60(1), 54-65.

Ferratt, T. W., Hall, R. H., Prasad, J., & Wynn, Jr., D. (2010). Choosing management information systems as a major: Understanding the smiFactors for MIS. *Communications of the Association for Information Systems*, 27, 265-284.

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, 17-36.

Frederiksen, A. (2009). Competing on analytics: The new science of winning. *Total Quality Management & Business Excellence*, *20*(5), 583–583.

Granger, M. J., Dick, G., Luftman, J., Van Slyke, C., & Watson, R. T. (2007). Information systems enrollments: Can they be increased?. *Communications of the Association for Information Systems*, 20, 649-659.

Harris, J. G. (2008). How to fill the analytics talent gap? *Strategy & Leadership*, *36*(5), 17–18.

Hershey, G. L. (2003). A different focus and content for the core information systems course for business school majors. *Communications of the Association for Information Systems*, 12, 479-493.

Ives, B., Valacich, J. S., Watson, R. T., Zmud, R. W., Alavi, M., et al. (2002). What every business student needs to know about information systems. *Communications of the Association for Information Systems*, 9, 467-477.

Karni, Edi (2013). Bayesian decision theory with action-dependent probabilities and risk attitudes. *Economic Theory*, 53(2), 335-356.

Keen, P. G. W. & Scott Morton, M. S. (1978). Decision support systems: An organizational perspective. Reading, MA: Addison-Wesley.

Leidner, D. & Jarvenpaa, S. (1995). The use of information technology to enhance

management school education: A theoretical view. *MIS Quarterly*, 19(3), 265-291.

Manski, Charles (2017). Optimize, satisfice, or choose without deliberation? A simple minimax-regret assessment. *Theory and Decision*, 83(2), 155-173.

McCoy, S., Everad, A., & Jones, B. M. (2015). Foundations of information system course content: A comparison of assigned value by faculty, recruiters, and students. Communications of the Association for Information Systems, 36, 697-705.

Power, D. (2013). Using big data for analytics and decision support. MWAIS Proceedings.

Reutskaja, Elena, Nagel, Rosemarie, Cramerer, Colin & Rangel, Antonio (2011). Search dynamics in consumer choice under time pressure: An eye-tracking study. The American Economic Review, 101(2), 900-926.

Rodammer, F., Speir-Pero, C. & Haan, J. (2015). The integration of business analytics into a business college undergraduate curriculum. *In Proceedings of the Twenty First Americas Conference on Information Systems.*

Schiller, S., Goul, M., Iyer, L. S., Sharda, R. & Schrader, D. (2015). Build your dream (not just big) analytics program. *Communications of the Association for Information Systems*, 37(40), 811-826.

Simon, H. A. (1959), Theories of decision-making in economics and behavioral science. *The American Economic Review*, 49(3), 253-283.

Sprague, R. H. (1980). A framework for the development of decision support systems. *MIS Quarterly*, 4(4), 1-26.

Topi, H., Valacich, J. S., Wright, R. T., Kaiser, K. M., Nunamaker, Jr., J. F., Sipior, J. C., & de Vreede, G. J. (2010). IS 2010: Curriculum guidelines for undergraduate degree programs in information systems. *New York, NY: Association for Computing Machinery & Association for Information Systems.*

Vob, S. (2014). An interview with Daniel Dolk and Christer Carlsson on "Decision analytics". *Business & Information Systems Engineering*, 6(3), 184-184.

Vob, S., Sebastian, H. & Pahl, J. (2017). Intelligent decision support and big data for logistics and supply chain management - a biased view. *In Proceedings of the 50th Hawaii International Conference on System Sciences.*

Watson, H. (2018). Revisiting Ralph Sprague 's Framework for Developing Decision Support Systems. *Communications of the Association for Information Systems*, *42*(13), 363–385. http://doi.org/10.17705/1CAIS.04213

www.aacsb.edu. (April 8, 2013 and revised January 31, 2016). Eligibility procedures and accreditation standards for business accreditation.

Zhang, W. (2007). Why IS: Understanding undergraduate students' intention to choose an information system major. Journal of Information Systems Education, 18(4), 447-458