Increasing Student Performance Through the Use of Web Services in Introductory Programming Classrooms: Results from a Series of Quasi-Experiments

Bryan Hosack
Billy Lim
W. Paul Vogt
Illinois State University
Campus Box 5150
Normal, IL 61790
bhosack@ilstu.edu, bllim@ilstu.edu, wpvogt@ilstu.edu

ABSTRACT

An introduction to programming course can be a challenge for both students and instructors. This paper describes a study that introduced Web services (WS) and Service-Oriented Architecture in Information Systems 1 (IS 1) and Computer Science 1 (CS 1) programming courses over a two-year period. WS were used as an instruction tool based on their increased use in industry as well as their ability to provide a real world feel to student programming activities. The paper includes an example WS teaching module and a proposed implementation model for future studies based on lessons learned from the current experiment. The study was successful in showing a significant increase in student test performance for WS-taught courses over standard-taught courses.

Keywords: Programming, Student performance, Web literacy

1. INTRODUCTION AND MOTIVATION

Historically, teaching introduction to programming can be challenging for both students and instructors for a variety of reasons from psychological to pedagogical (Sheil, 1981; Kolling et al., 1995; Huet et al., 2004; Pendergast, 2006; Avouris et al., 2010). This is also evident in the plethora of approaches from robots to games or different models of delivery (Lawhead et al., 2003; Rajaravivarma, 2005; Pedroni and Meyer, 2006). This paper describes a study that introduced Web services (WS) and Service-Oriented Architecture (SOA) in Information Systems 1 (IS 1) / Computer Science 1 (CS 1) programming courses over a two-year period. The benefit of using the WS over a standard, typical IS1/CS1 teaching approach was evident in this study as seen in the increase in a common (no WS content) final exam performance at a large Midwestern University. While there are many factors that affect student performance in any course, there was a positive gain in WS sections over a typical section offering. This coupled with exposing students with a burgeoning technology used in the IT industry and, by their admission, tools that were interesting, we feel the study was successful and merits further study in the area.

WS provide a standard means of interoperating between different software applications, running on a variety of platforms and/or frameworks (Booth et al. 2004).WS were chosen as a mechanism to increase student interest because of the ability to access familiar real world contents, such as a Google map or a Twitter feed, and bring this data into their assignments and projects. Furthermore, WS are becoming a strategic platform that supports how companies use IT to conduct business. Industry use of WS continues to grow as the benefit WS for standardizing the integration of applications to delivering more complex services (Gates, 2008; Laufmann, 2010; Phifer, 2012). According to a recent survey of industry professionals from both public and private sectors conducted in the fall of 2010 by the authors in support of this study, forty-one IT professionals from a broad range of organizations indicated that their companies were using WS (9 always use WS, 30 sometimes, 2 do not use WS) and that WS should be integrated into a university or college IT curriculum. As a follow-up to the survey, ten of the survey participants were interviewed during the spring of 2012. Over this two year period, the interviewees indicate that WS use in companies has 1) grown in the past two years, 2) become a strategic IT focus of the companies, and 3) shifted from not only working with custom WS but also using more third party and/or public facing WS across the entire organizations. To address the growing need for IT professionals to work in this space, the same interviewees supported 1) including WS as part of an IT curriculum, 2) having a class or classes focused on WS, and 3) ideally targeting upper level students with WS but possibly spreading throughout the curriculum when relevant. The findings lend support to the use of WS in the teaching approach outlined here.

The WS approach to teaching IS1/CS1 integrates the use of WS technology throughout the course assignments and lectures. Students in the WS sections were shown to perform better on a common final exam then students in standard sections. The common final exam tested programming concepts and was given to all introductory sections. WS were not covered as part of the final exam. The approach provides students with an interesting collection of services that allows for more sophisticated apps to be built. Students using WS early in the course can see the benefits of reuse and, by the end of the course, build mashups that involve Google Maps, YouTube, Twitter, etc. as opposed to producing programs that may teach the same concepts but fail to allow students to connect their work to the real world.

The results indicate that the approach presented here was successful and based on the outcome of this study, a framework that includes a comprehensive pre-test and post-test for students in the control and treatment sections to complete, a common content knowledge survey module for all students to take, and a faculty survey for the instructors to complete is proposed. This will enable educators to answer many questions regarding the effectiveness of the WS approach, including “Do students using the Web service approach perform better in the common assessment exam module?” and “Do students and faculty members find the Web service approach more engaging?”

The remainder of the paper is organized as follows. A review of the relevant literature is presented in the Related Work section below. This is followed by a sample module of the WS approach. The research design of the study is discussed next, followed by the results of the study. A discussion of the results is presented in the section that ensues. Finally, the conclusions, limitations, and future work of this study are given in the last section.

2. RELATED WORK

This section is divided into two subsections to highlight efforts to improve introductory programming or programming courses that have incorporated WS. But under our extensive literature review, we found no indication of a course that did both.

2.1 Introductory Programming Approaches

In terms of approaches used for teaching introductory computing, there have been numerous proposals reported in the literature over the years for new and appealing methodologies to attract and retain students. They include using personal robots in a CS1 course, through the Institute for Personal Robots in Education program (Markham and King, 2010) or LEGO Mindstorms (Lawhead et al., 2003; McWhorter and O’Connor, 2009). Another notable effort is one that uniquely makes use of the context of art and creative coding (Greenberg et al., 2012). Here, the students create a portfolio of aesthetic visual designs that employ basic computing structures. Other approaches have used a Web centric approach to teaching IS1/CS1, but do not incorporate WS (Stepp et al., 2009; Yue, 2010).

There is also an approach that leverages active learning techniques in the form of team-based learning (TBL). The effects of TBL have been shown to have major improvements both in terms of the drop rate and students’ success, as measured by final exam grades (Lasserre and Szostak, 2011). Yet another approach takes the gaming route and introduces games as a “flavor” of CS1 (Bayliss and Strout, 2006) and a simple framework for interactive games (Luxton-Reilly and Denny, 2009). Lastly, similar to the WS approach proposed in this research, where problem solving involving real world activities are emphasized, real world programming assignments such as spam evaluator and web crawling are integrated in a CS1 course (Stevenson and Wagner, 2006).

2.2 WS in IT Courses

In terms of the use of WS in IT curricula, there have been a number of efforts that involve the use of SOA/WS in education in some fashion. First, the work by the authors and colleagues on the initial concept (Lim et al. 2005) and later on the interim report (Hosack et al. 2011), are the only ones that aim at the introductory level. All the others are primarily in upper division, emerging technology, capstone, and/or graduate IT curricula. For example, Humphrey uses WS as the foundation for learning complex software system development in a first-year graduate course to allow for more concrete discussion of software design, implementation, and evaluation (Humphrey, 2004). A similar effort, but for an undergraduate project-based course and using open source software, is reported by Reed and colleagues (Reed et al., 2007).

Another work in the area involves a graduate-level XML programming course where a WS-based solution is used to address the problems of insufficient complexity in a typical course project and the need to prepare students to work on real-world project teams (Zilora, 2004). Yet another effort is given by Assunção and Osório (2006), where the teaching of WS concepts, standards and technologies using the .NET platform (Visual Studio .NET with Web Services Enhancements tools) is described.

More recently, Holliday et al. describe the historical development of network programming techniques (from low level sockets programming to Remote Method Invocation) and extends the techniques to WS (Holliday et al., 2008). Also, the general concern about how SOA can be used in a learning environment and how the environment must be articulated in the context of business needs and other software architecture methodologies are described by Lopez et al. (Lopez et al., 2007). Finally, Tsai (Tsai et al., 2008) introduces WS at the introductory level (high school grades 9 through 12 in this case).

In each of the above publications, the work described has been aiming at the non-introductory levels. The only exception is the work by Tsai. However, in that work, the use of WS occurs at a very specialized level—in robotics programming, unlike the generalized approach used in this research.

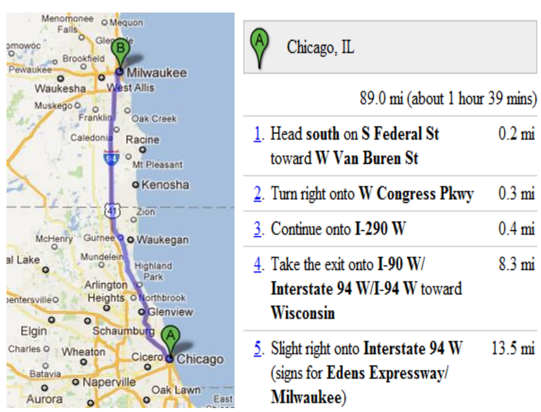
3. SAMPLE TEACHING MODULE OF THE WS APPROACH

This teaching study targets sections of Information Systems 1 (IS 1)/Computer Science 1 (CS 1), the early programming courses in the computer science and information systems programs in many curricula. Both courses are designed to introduce the basic problem solving and program design skills that are used to create computer programs. To give a sense of how the WS approach is used, a sample module comparing the WS and traditional approaches for a typical topic covered in IS1 or CS1 is presented. This topic, along with various other topics, can be easily enhanced so that students are exposed to the state-of-the-art technology. The topic is presented with a typical delivery mechanism using the traditional approach, then augmented with the WS approach, and finally followed by an example depicting the WS approach to the topic.

In the following selected module, the topic presented is "Sequence, Iterative, and Decision Structures." The learning objectives aim to reinforce the concepts behind the fundamental control structures of sequencing, looping, and decision making (via if/else and/or case statements). Upon completion of this module, students should be able to ascertain the order in which the various tasks need to be carried out, to apply the appropriate looping structure to iterate over a collection of data, and to impose the necessary conditions to filter the data for display purposes.

In the table below, three sections (Typical Delivery, WS Delivery, and Example) are presented.

Module Name:	**Sequence, Iterative, and Decision Structures**
Typical Delivery:	These topics are typically covered by traditional discussion of scenarios that (1) necessitate a certain ordering be imposed in order to solve a problem (e.g., read the input values before processing them), (2) require a loop be used (e.g., processing a collection of numbers to find the average), and (3) need an if-else structure be employed (e.g., find the largest and smallest numbers from a collection of numbers).
Web Service Delivery:	Instead of merely processing a collection of meaningless numbers or strings that may not resonate with students, one could present a scenario where the goal is to solve a problem by using the three fundamental structures and existing WS to form a solution.
Example:	A plausible scenario would be to solve the problem of finding the nearest city from, say, Chicago, given a collection of cities to process. Further, the nearest city needs to be plotted on a map. Lastly, get a route from Chicago to the nearest city. This scenario may seem intractable in the traditional introductory programming environment. But there exist various publicly available WS that can be composed together to form a *mashup* application that solves this problem rather effortlessly. For example, there exist WS that convert a given city into its latitude/longitude coordinate, find the distance between two coordinates, plot a particular coordinate on a map, and plot the route given two coordinate endpoints. Thus, one can cover the Sequence, Iterative, and Decision topics using a more interesting approach. There are a variety of web sites that offer freely available WS. Sites such as xmethods.net, webservicex.net, and wiki.cdyne.com would be three good places to start looking. Here, the students need to determine the sequence in which the tasks ought to be carried out. They also need to setup a loop that iterates over all the cities. Then, as each city is processed, its latitude/longitude coordinate needs to be determined and compared with the current nearest city (involving if/else statement). Finally, once the nearest city is ascertained, a map and a route can then be plotted, as given below in Figure 1.



[*directions truncated*]

Figure 1. A plot of directions to a location

In the “Typical Delivery” section, a typical approach used for discussing the topics of “Sequence, Iterative, and Decision Structures” is discussed. One example that encompasses all three aspects of the above structures is: “Process a collection of numbers (from the user), determine which one is the largest, and finally display it.” Clearly, the *sequential* aspect of this is that one needs to read the input first before one can decide and then display the largest. The *iterative* aspect is that one needs to establish a loop to go through the list. The *decision* aspect is that as each number is processed, an if/else statement is needed to keep track of the largest (so far).

In the “WS Delivery” section, a comparable scenario to the above is described. The idea here is to cover the same topics, but using WS as the delivery mechanism. With WS, the possibilities are endless and one can be creative in incorporating the topics at hand in a way that engages the students more. For example, instead of processing a random list of numbers, the students can be processing a set of cities and determining which one is closest to a given city. Further, an added task might be to plot on a map the route to go from the given city to the closest one found. Now, the input data have meanings and the processing seems more interesting as it ties in with their general knowledge about the US geography and their experience with mapping.

Finally, in the “Example” section, a specific scenario that details how the “WS Delivery” section can be implemented is given. In the table, the example is about finding closest city to the city of Chicago, plotting the cities on a map, getting a route to go from one location to another, and displaying turn-by-turn instructions for the route.

With many modern Integrated Development Environments (IDE) such as NetBeans 7.x (NetBeans, 2012) and Eclipse 3.7.x (Eclipse, 2012), one can easily plug in a given WSDL (Web Service Description Language) URL, which describes what the WS is and where and how it can be accessed, and get the relevant code auto-generated. For example, in NetBeans 7.0.1, the user interface for the process is given below in Figure 2. Here, the WSDL for a Microsoft’s WS called TerraService, which allows one to convert a city/state/country to its latitude/longitude coordinate, among other things, is used to generate the necessary code to communicate with the underlying WS.

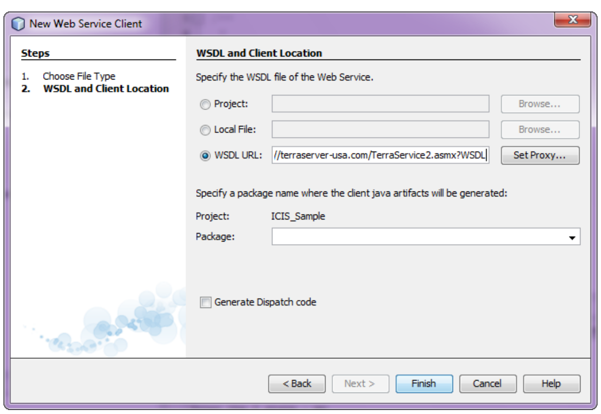


Figure 2: NetBeans WS Wizard

The generated code, which consists of a collection of Java classes that permit the client to communicate with the remote WS, is readily available for a client program to create a local object that communicates with its remote WS counterpart, see Figure 3. A similar approach can be used other programming interfaces such as Microsoft’s Visual Studio IDE, using Visual Basic and/or C#.

```
init:
wsimport-init:
Created dir: C:\ICIS_Sample\build\generated-sources\jax-ws
wsimport-client-TerraService2:
Created dir: C:\ICIS_Sample\build\generated\jax-wsCache\TerraService2
command line: wsimport -d C:\ICIS_Sample\build\generated\jax-wsCache\TerraService2
parsing WSDL...

generating code...

...
com\msrmaps\ConvertLonLatPtToNearestPlace.java
com\msrmaps\ConvertLonLatPtToNearestPlaceResponse.java
com\msrmaps\ConvertPlaceToLonLatPt.java
com\msrmaps\ConvertPlaceToLonLatPtResponse.java
...

compiling code...

javac -d C:\ICIS_Sample\build\generated\jax-wsCache\TerraService2 -classpath C:\Pr
Copying 54 files to C:\Billy\JavaTestingGround\NetBeans7.0\ICIS_Sample\build\gener
BUILD SUCCESSFUL (total time: 9 seconds)
```

Figure 3. Code generated when importing a WS

Figure 4 below shows how one can use the generated code to communicate with the WS to convert a city to its latitude/longitude coordinate. First, note that the IDE generates the convertPlaceToLonLatPt method where the WS and its port (endpoint) are created upon request (a simple drag and drop). With the generated method, one simply needs to call it with a Place object and expect to get a LonLatPt object back. In the main method, a place object is created and its city, state, and country set. The method is called and with the returned result, the latitude and longitude of the city are displayed.

Sample Code:

```
public static void main(String[] args) {
  // This code finds and displays the lat/long coordinate of
  // Chicago, IL
  com.msrmaps.Place place = new com.msrmaps.Place();
  place.setCity("Chicago");
  place.setState("IL");
  place.setCountry("United States");
  LonLatPt result = convertPlaceToLonLatPt(place);
  System.out.println("The lat/long coordinate is:
                       "+result.getLat()+"/"+result.getLon());
}

// This method is generated by the NetBeans IDE when the WS
// method is dragged and dropped private static LonLatPt

convertPlaceToLonLatPt(com.msrmaps.Place place) {
  com.msrmaps.TerraService service = new
                       com.msrmaps.TerraService();
  com.msrmaps.TerraServiceSoap port =
                       service.getTerraServiceSoap();
  return port.convertPlaceToLonLatPt(place);
}
```

Output:

The lat/long coordinate is: 41.900001/-87.629997

Figure 4. Sample code and the resulting output utilizing the city to latitude/longitude conversion WS

This section illustrates how the topic of simple control structures, typically covered using the traditional approach, may be covered using an approach that is enhanced with WS. It shows that students can begin to experiment with a state-of-the-art technology that permits them to explore various, more meaningful data when learning essential topics in an introductory course.

4. RESEARCH DESIGN

The course offerings at a large mid-western Midwestern state university located in the USA were used to conduct a series of quasi-experiments. Included were the introductory programming courses in: Information Systems 1 (IS 1) and Computer Science 1 (CS 1). Because the courses were offered in the same school in the same university, students were from a similar population. Sections of traditionally-taught introductory Java programming courses were compared with sections of the WS-based introductory Java programming. The IS1/CS1 courses were combined in an effort to ensure that a large enough sample of students was available for analysis and because the content covered in each course was similar. Each course is required for the respective majors in the two fields. It should be noted that the same topics and concepts were covered in all of the IS1/CS1 sections whether they were using WS or not. Table 1 provides the number of sections and students taught over four semesters.

	Computer Science 1 (CS1)	Information Systems 1 (IS1)
Traditional	4 (150)	5 (214)
Experimental (Web service)	4 (94)	4 (128)

Table 1. Number of Sections (Number of Students)

At least two sections of each course were offered in each semester in both IS1/CS1, one experimental section and one or more traditional sections. Each semester, two CS instructors and two IS instructors taught the courses; a total of six instructors taught over the four semesters.

Students registered for the classes in the typical way, which means they were not randomly assigned to the control and experimental groups. While random assignment would have enhanced the internal validity of the study, the fact that the experiment was conducted in actual programming courses offered in a university setting, with students who selected courses using their usual criteria (e.g., to fit their schedules), did much to insure the generalizability of the study's results to a real world context.

To check for possible selection bias that might have arisen from student choices of class sections, we compared students who enrolled in the WS sections with those in the traditional sections in terms of student gender, academic majors, and mean cumulative grade point averages (GPA). A total of 586 student participants were involved in the study over four semesters. As can be seen in Table 2, the students in the WS-taught classes were 38% of the sample, those in the traditionally-taught classes were 62%; 20% of the students were female, 80% male. After the University official withdrawal date a total of 514 students continued in the treatment and comparison group classes. Table 2 provides further descriptive statistics on the 586 students who enrolled in the four semesters from the Fall of 2009 through the Spring of 2011.

When gauging the effects of the experimental teaching method on the outcome measures (criteria), all of the variables described above (gender, major, class rank, cumulative GPA, etc.) were controlled in the analyses. The main analysis method was multiple regression, which enabled the researchers to measure the size of the effects of the independent variable (WS instruction) on the dependent variable (student learning as indicated by final exam scores) while controlling for covariates that have been shown in past studies to influence the outcome or dependent variable (academic major, class rank, GPA, and gender).

Variable	Number of Students	Percent
Semester		
Fall 2009	183	31.2
Spring 2010	119	20.3
Fall 2010	167	28.5
Spring 2011	117	20.0
Totals	586	100.0
Gender		
Female	117	20.0
Male	469	80.0
Totals	586	100.0
Class Rank		
Freshmen	133	22.7
Sophomores	161	27.5
Junior	176	30.0
Senior	90	15.4
Other	26	4.4
Totals	586	100.0
Major		
CS & IS	278	47.4
Others	308	52.6
Totals	586	100.0
Instructors		
Instructor 1	105	17.9
Instructor 2*	94	16.0
Instructor 3*	128	21.8
Instructor 4	202	34.5
Instructor 5	27	4.6
Instructor 6	30	5.1
Totals	586	100.0
Group		
Control	364	62.1
Experimental	222	37.9
Totals	586	100.0

*Note: Instructors 2 and 3 taught the experimental WS sections.

Table 2. Students by Semester, Gender, Class Rank, Major, Instructor, and Group

Group, WS v Tradl		N	Mean	Std. Deviation	Std. Error Mean
GPA, Cumulative	WS	222	2.71	.76606	.05141
	Traditional	364	2.58	.91999	.4822
	Total	586	2.62	.86641	.03579

Note: t = 1.716; df 584, [F = 2.946, df 1] p = .087.

Table 3. Cumulative GPAs of Students in WS and Traditional Classes

Cumulative GPA was of particular concern as it is frequently the most important predictor in studies such as this one. As can be seen in Table 3 above, there was little difference in the cumulative GPAs between students registering for the WS and the traditionally taught classes (2.71 and 2.58 respectively). Not surprisingly, a difference of 0.13 was not statistically significant at the .05 level (t = 1.716; df 584, [F = 2.946, df 1], p = .087).

Similarly, there were no significant differences in enrollment patterns (i.e., student self-selection into the control and experimental groups) between males and females (Table 4, Chi Squared = 0.325, p = .569) nor among academic majors (Table 5, Chi squared = 1.299, p = .254).

		Group, WS v Tradl		
		Traditional	WS	Total
Gender	Female	70	47	117
	Male	294	175	469
Total		364	222	586

Note: Chi Squared = .325; p = .569.

Table 4. Gender Distribution of Enrolled Students

		Group, WS v Tradl		
		Traditional	WS	Total
Major	Others	198	110	308
	IS & CS	166	112	278
Total		364	222	586

Note: Chi Squared = .1.299; p = .254.

Table 5. Majors of Enrolled Students

The main criterion (outcome) variable was: Final Exam Scores. The scores were obtained using two common cumulative final exams—one for IS sections and one for the CS sections—with each major exam (IS or CS) containing the same problems. WS were not included as part of the exam material since WS were used in the experimental sections as a tool to illustrate and to teach the concepts of programming, but not used in the traditional sections. Therefore, the cumulative final focused on the conceptual material covered by both the control and experimental sections.

5. RESULTS

The design was quasi-experimental. The 586 student participants were not randomly assigned to treatment and comparisons groups. Rather they enrolled in class sections using whatever criteria they happened to use. Treatments were assigned to some of those sections; the students in the other sections served as a comparison group. The disadvantage of a quasi-experimental design is that the researchers had no control over the *assignment* of students to experimental conditions. Because the learning experiment was not a laboratory simulation but occurred in ordinary classes, the investigators also had less control over the *delivery* of the instruction than they probably would have had under more controlled laboratory conditions. The advantage of the study design was ecological validity (the study closely approximated the situations to which it was intended to generalize). The participants were real students in real courses earning grades for credit. It could be argued that investigating the real world, rather than the laboratory world, makes it more likely that what was learned in the research could be generalized to other real students in real courses.

Even had the assignment to sections/groups been random, it would still have been important to check for the equal distribution of non-treatment variables that researchers knew from literature reviews could influence the outcome variable (final exam score). In a quasi-experiment without random assignment, this step is crucial. Three of such covariates examined in this study were students' cumulative GPAs, their genders, and their academic majors (see tables 3, 4, and 5 above). These three in addition to class rank were included in the regression models along with the independent variable: WS versus standard instruction. Regression analysis assumes the normal distribution of variables and the samples from which they were drawn. Both the independent variable (treatment versus control) and the dependent variable (exam scores) were left-skewed (more scores at the lower end of the range), but not seriously enough to merit transforming the data before analysis (details are available from the authors).

It makes most sense to initially evaluate the project as four separate quasi-experiments conducted over two years in four consecutive semesters. While the treatment was repeated each semester, semester-long teaching activities naturally varied from semester to semester and over the two years. The samples were distinct each semester. The comparison group instructors also differed from one semester to the next, and their teaching also undoubtedly varied.

Table 6 summarizes the results summary for each semester. The main dependent variable is score on the common final exam. This is measured two ways for each semester: with zeros included and excluded. In each semester some students simply did not show up for the final examination and were assigned a score of zero. One could make a case for including the scores of these students, who had, in essence, unofficially withdrawn from the course, or for excluding them as missing data. We think the argument for the latter is stronger, but in the name of completeness we present the results both ways.

Column 1 gives the number of students still enrolled in the course at the end of the semester who received a score on the final exam. Looking at the first two rows, for Fall of 2009, we can see the effects of the different ways of computing the outcome variable. For example, in row 1a, with zeros included, 164 scores were used to calculate the results; in row 1b, 145 were. The difference between the N values 164 and 145 indicates that 19 students unofficially dropped the course by not taking the final exam. Excluding those scores of zero raises the exams' means (Column 2) and reduces their standard deviations (Column 3). The overall effect of the missing data is to reduce the variance and thereby the percentage of the variance explained by the predictor variables (shown by the adjusted R^2 in Column 7).

The main findings are in Column 4, which presents the regression coefficients *b*. The figures in the column indicate the difference in the scores of students who were in the experimental groups. For example, the first number in Column 4—6.38—means that on average students in the experimental group classes scored 6.38 points higher on the final exam than students in the comparison group classes. This figure controls for other variables available to the researchers that could have explained the outcome: students' cumulative GPAs, genders, class ranks (freshman, sophomore, etc.) and their academic majors (Computer Science or Information Science, versus all others).

We also present the *p* values and the 95% confidence intervals for those regression coefficients (in Columns 5 & 6 respectively). We do this more out of tradition than from a belief that these statistics are appropriate for these data from quasi-experiments. The *p*-value indicates the probability of the outcomes in the population being as large, or larger than those in the *random* sample, if the null hypothesis were true. The null hypothesis here is: no difference between the scores of the experimental and comparison groups. However, it should be stressed that the calculation of *p*-values and confidence intervals is firmly based on an inferential statistical theory that *assumes* that the students were randomly sampled from a known population and/or that they were randomly assigned to experimental and comparison groups. Neither of these *necessary* assumptions is true in our research. When these statistics are provided in research and for samples such as this one, what they represent is what the statistics *would have been if* the students *had been* randomly sampled from a population and randomly assigned to control and experimental groups.

The results in Table 6 (apart from Columns 5 and 6) are descriptive, not inferential. They pertain to the samples actually studied in the quasi-experiments. We can make no claim that they are representative of the results that would be achieved at other universities with other professors. Such claims can only be based on broader studies, which is why we have expanded our study to include several other institutions and instructors (see discussion below).

In any case, the results from the individual semesters in Column 4 are mixed. For 3 of the semesters the increase associated with being in the experimental group classes ranged from about 2 to about 7 points on a 100-point test with mean scores ranging between about 60 and 70. However, in one semester (Fall '10, lines 3a & 3b), the results were strongly in the opposite direction: students in the experimental sections got markedly *lower* scores, between 9 and 16 points lower depending on the method of calculation (including or excluding zeros). When discovering such an anomalous result, you have to ask yourself, in the words of an old statistics professor: "Is this a clue to pursue or a case to erase?" Often, it is some of each, as it may be here. Examining the detailed data, it appears that students taking the programming from one of the experimental group's instructors were less inclined to drop out either officially or unofficially. Rather, this instructor's students tended to persist in the course and earn low grades on the final exam, an effect that seemed to have been especially marked in the 3rd iteration of the experiment. One could make a case that such student persistence is itself a positive outcome even if it tends to spoil the experimental results by altering the measurement criteria. This kind of complication is another consequence of studying the real world rather than the laboratory world.

With comparatively small numbers of cases—ranging from 82 to 166 in our four quasi-experiments—regression estimates can be quite unstable, especially as the number of predictors grows. We have 5 predictors in our study: membership in the experimental group, GPA, gender, class rank, and academic major. One solution to the problem of unstable estimates due to small sample sizes is to pool data into a sort of meta-analysis of the 4 sets of outcomes. When

	1	2	3	4	5	6	7
	N	Mean	SD	*b*	*p* val	95% CI	Adj R^2
SEMESTERS							
1a. Fall 09, zeros included	164	65.47	28.37	6.38	.084	-.857, 13.62	.441
1b. Fall 09, zeros excluded	145	74.05	16.48	7.02	.007	1.93, 12.10	.268
2a. Sprg 10, zeros included	92	63.28	27.04	4.77	.242	-3.29, 12.83	.499
2b. Sprg 10, zeros excluded	82	71.00	16.33	1.92	.519	-3.98, 7.82	.357
3a. Fall 10, zeros included	166	51.74	32.35	-9.16	.064	-18.85, 0.54	.233
3b. Fall 10, zeros excluded	129	66.57	18.81	-15.79	<.01	-22.01, -9.57	.226
4a. Sprg 11, zeros included	92	58.37	26.76	5.09	.275	-4.12, 14.30	.346
4b. Sprg 11, zeros excluded	89	60.33	24.91	5.34	.246	-3.75, 14.43	.289

Table 6. Scores on the common final exams, by semester

Model		Unstandardized Coefficients		Standardized Coefficients	t	Sig. (*p*)
		b	Std. Error	Beta		
1	(Constant)	17.176	5.682		3.023	.003
	Expm Group = 1	4.168	2.022	.102	2.061	.040
	Major, IS & CS = 1	2.751	2.113	.069	1.302	.194
	class rank (1 – 4)	.361	.981	.019	.368	.713
	Cum GPA (0 – 4)	15.089	1.495	.504	10.094	.000
	Gender, male = 1	6.794	2.442	.139	2.783	.006

Table 7. Regression results for Semesters 1, 2, & 4 pooled, N = 316

pooling data from all 4 semesters (full results not shown but available from authors), the modest positive results from semesters 1, 2, and 4 are canceled out by the negative results from semester 3. The *b* coefficients for all semesters pooled are, depending on the methods of calculation, + 0.878 and −1.34, which essentially indicate no effect one way or the other.

If the explanation for the anomalous 3rd semester is convincing, we can set it aside and pool the results from the other three semesters. The results are presented in Table 7.

Also presented are the full results of the regression analysis including the coefficients for all 5 predictors: the independent variable and the 4 control variables. This enables us to examine the comparative importance of these predictors. The unstandardized coefficients, *b*, present the results in the original metric: points on the final exam. In Table 7 we see that students in the experimental group scored 4.168 points higher on average (after controlling for other variables). This is a gain of about 6% (4.168 ÷ 69.39 = 6.0%) for students in the experimental group. Students who were Computer Science or Information Science majors got 2.75 points higher. Each increase in class rank, from freshman to senior (coded 1 – 4) was associated with about 1/3 of a point on the final. The effect of cumulative GPA (on a 4-point scale, 0 – 4) was associated with a whopping 15 points on the final exam. It is hardly surprising that students who generally do well in their courses tended to do better in this course too—and vice versa. Finally, gender also mattered; after controlling for the other variables, males scored an average of 6.8 points higher. These results are substantively interesting and also show why it was important to include the control variables in the model and why we have continued to do so in the expansion study described below. When controls are not included, it is difficult at best to estimate treatment effects accurately.

For comparing the relative sizes of the 5 predictor variables, one should use the standardized regression coefficient, beta. By expressing outcomes in standard deviation units, the beta adjusts for differences in measurement scales (e.g., Experimental Group, 1 – 0; Class Rank, 1 – 4; GPA, 0 – 4). The beta coefficients can be used as effect size measures in this mini-meta-analysis because the number and definition of the predictor variables is the same for each quasi-experiment. In Table 7, the pooled data for the three semesters are presented (using the somewhat more conservative estimate treating zeros as missing cases). The beta for experimental group membership is .102. This means that the gain attributable to participation in the experimental group is about one-tenth of a standard deviation, a number which by many rules of thumb is considered the meaningful minimum for policy and decision making. The effect size for gender is similar to that for the experimental group membership, while the influence of cumulative GPA is about 5 times as large (.504 or ½ a standard deviation). That indicates why it will be important to control for these variables, especially GPA, in subsequent studies. Without controlling these covariates, the effects of the treatment variable could easily be concealed.

In sum, we would characterize the overall results of the 4 quasi-experiments as encouraging, but by no means definitive. Even had the outcomes been stronger and clearer, no one university and no group of 6 instructors can be considered representative. That is why we have endeavored in our expansion study to include a much broader group of institutions and instructors and to improve the reliability and validity of our measures of predictors, mediators, and outcomes. These improvements are sketched in the following paragraphs.

6. LIMITATIONS, CONCLUSIONS, AND FUTURE RESEARCH

There are several limitations to the conclusions of this study. Most obviously, the data analyzed is from courses offered over four semesters by six instructors in one university. More data will need to be collected to determine whether the initial successes in the pilot years can be replicated and improved upon in the expansion study.

It should also be noted that there were a number of WS failures in this pilot study. Because this was a pilot and our experiences with the WS were limited in terms of their reliability, several publicly available WS were either not available when needed (perhaps due to heavy student usage) or simply disabled prior to student use. This complicates the assessment of the effectiveness of the WS approach. To overcome this shortcoming, homegrown WS have been developed to serve as backup services in case of failure.

Based on the pilot study, several improvements are being implemented in the expansion study. The assessment model from the pilot study was revised into a new framework that is more standardized and comprehensive. Four new variables have been added to the experimental model: two pertain to student characteristics — their self-efficacy and their engagement in instruction; also added are two new measures of students' learning.

In the pilot study, the instruments for assessing student learning (final exams) were developed in-house and not based on a standardized, widely used instrument that has been tested extensively. To address this shortcoming, a revised instrument that is based on SALG (Student Assessment of Learning Gains) (Seymour et al., 2000, www.salgsite.org; Douglas et al., 2012) has been developed. SALG is a nationally validated pre- and post-survey of students' self-assessment of their knowledge before and after a course. Because it has been used in numerous courses over many years, it can provide the basis for measured comparisons of student learning. In addition to the students' assessment of their learning, an objective test of knowledge of programming concepts has been developed in consultation with a team of faculty members teaching programming courses. This constitutes a researcher assessment of learning to supplement the student assessment; it will be tested for reliability and validity in the early iterations of the expansion study.

While measuring student learning was a key objective of the pilot project, measuring student engagement was not studied as systematically. Given the nature of WS, which allows for the wealth of information on the Web to be harvested easily through API (application programming interface) calls from one's computer program, it would be remiss for the new framework to not capture student engagement. Students are expected to be more engaged with the WS approach as they are interacting with activities that they often personalize to make them more interesting and relevant (e.g., find all 3D movies that are playing in *my* hometown (zipcode xxxxx), display all comments from *my* favorite YouTube video, etc.). To measure student engagement, we have also included in the SALG post-survey an instrument designed to capture student engagement. The instrument is modeled after several others in the field and will serve as a mediating variable in the analysis (Ahlfeldt et al., 2005; Carini et al., 2006).

There have been many different efforts in the literature on engaging student learning using a variety of approaches. They include the application of "gamification" to eLearning to engage learners where the theory behind gaming design is applied to build engagement interactive materials such as eLearning (Raymer, 2011), the study of how learning community participation affects student engagement (Pike et al., 2011), the research on curiosity, or interest and engagement in technology-pervasive learning environments (Arnone, 2011). The proposed framework in the expansion study allows the researchers to assess if the WS approach represents another means to actively engage students in learning the fundamentals of computer programming.

In addition to using the SALG assessments the researchers have designed an assessment test module of objective questions to be taken by students in both the control and experimental classes at the end of each semester. The questions measure student knowledge of programming concepts and skills. This common module of objective questions will allow comparisons across universities. The questions have been reviewed at a workshop with the first cohort of faculty participants; in the judgment of that group as well as of the principle investigators the questions have extensive face validity. Finally, the use of objective questions with a large N of student participants will enable the researchers to use more advanced analytic techniques to measure student outcomes in the study, specifically: (1) propensity score matching to simulate experimental attribution of cause and (2) item response theory (specifically differential item functioning or DIF) to conduct subgroup analyses of responses to particular questions in the module. The combination of these factors yields the causal model shown in Figure 5.

The model postulates that WS instruction will promote student engagement, which, in turn, will foster their learning of programming concepts. Students' background variables and their self-efficacy (Bandura, 2006; Zajacova et al., 2005) enter the model from the outside; they are determined before students begin the course. Still, because of their potential influence on the outcome, it is important to control for these external/exogenous variables. This more complete model enables the researchers to gauge the effects of WS instruction with greater accuracy.

In conclusion, a quasi-experimental study was presented that indicates using WS in an introductory programming course significantly improved test scores by almost a half a letter grade (4.2 on a 100 point grading scale) for sections taught with this approach as opposed to a standard offering. Suggestions for future research include using the SALG instrument for a pre- and post-test to test the proposed research model in Figure 5 at other colleges and universities, teaching introductory programming courses incorporating various programming languages.

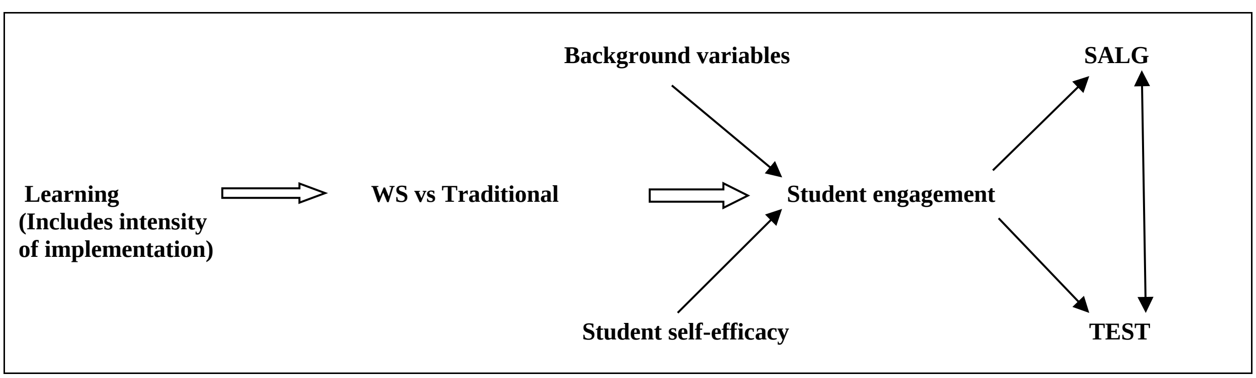


Figure 5. Causal Model

7. ACKNOWLEDGEMENTS

This material is based upon work supported by the National Science Foundation under Grant No. NSFDUE-CCLI-0837056.

8. REFERENCES

Ahlfeldt, S., Mehta, S., & Sellnow, T. (2005) “Measurement and analysis of student engagement in university classes where varying levels of PBL instruction are in use.” *Higher Education Research and Development, 24*, 5-20.

Assunção, L. & Osório, A. (2006) “Teaching Web Services Using .NET Platform,” *ACM ITiCSE'06*, Bologna, Italy.

Avouris, N., Kaxiras, S., Koufopavlou, O., Sgarbas, K., & Stathopoulou, P. (2010) “Teaching Introduction to Computing through a project-based collaborative learning approach,” *2010 14th Panhellenic Conference on Informatics*, pp. 237-241.

Bandura, A. (2006). Guide for constructing self-efficacy scales in *Self-Efficacy Beliefs of Adolescents.* New York: Information Age Publishing.

Bayliss, J. & Strout, S. (2006) “Games as a ‘Flavor’ of CS1,” SIGCSE'06, March 1–5, 2006, Houston, Texas, USA

Booth, D, Haas, H., McCabe, F., Newcomer, E., Champion, M., Ferris, C., & Orchard, D. (2004) “Web Services Architecture,” *W3C Working Group Note (http://www.w3.org/TR/ws-arch/)*, 1-98.

Carini, R.M., Kuh, G.D., & Klein, S.P. (2006) “Student engagement and student learning.” *Research in Higher Education, 47*, 1-25.

Douglas, J.A., Thomson, G., & Zhao, C. (2012) “The learning outcomes race: the value of self-reported gains in large research universities.” *Higher Education, 64,* 317-335.

Eclipse 2012, *Eclipse IDE*, Retrieved May 2, 2012 from http://www.eclipse.org.

Gates, W. (2008) “Internet Software Services,” Retrieved May 16, 2008 from http://www.scripting.com/disruption/mail.html.

Greenberg, I., Kumar, D., & Xu, D. (2011) “Creative Coding and Visual Portfolios for CS1,” *SIGCSE’12*, February 29–March 3, 2012, Raleigh, NC, USA.

Holliday, Mark A., Houston J. T., & Jones, E. M. (2008) “From Sockets and RMI to Web Services,” *ACM SIGCSE ’08*, Portland, OR, USA.

Hosack, B., Lim, B., & Vogt, P. (2011) “Incorporating The Use Of Web Services In Introductory Programming Classrooms,” Journal of Informatics Education Research.

Huet, I, Pacheco, O. R., Tavares, J., & Weir, G. (2004) “New Challenges in Teaching Introductory Programming Courses: a Case Study,” *34th ASEE/IEEE Frontiers in Education Conference*, pp. T2h5 –T2H9

Humphrey, M. (2004) "Web Services as the Foundation for Learning Complex Software System Development," *35th SIGCSE Technical Symposium on Computer Science Education*, Norfolk, Virginia USA March 3 - 7.

Kolling, M., Bett, K., & Rosenberg, J. (1995) “Requirements for a First Year Object-Oriented Teaching Language,” *26th SIGCSE Technical Symposium on Computer Science Education*, pp. 173-177.

Lim, B., C. Jong, & P. Mahatanankoon (2005) "On Integrating Web Services from the Ground Up into CS1/CS2," *ACM SIGCSE 2005*, St. Louis, MO, Feb., 2005.

Lasserre, P., Szostak, C. (2011) “Effects of Team-Based Learning on a CS1 Course,” *ITiCSE’11*, June 27–29, 2011, Darmstadt, Germany.

Laufmann, S. (2010) “When To Use Web Services: Don't deploy Web services without exploring these essentials.” *InformationWeek*, July 10, 2010 (accessed via the web, 5/2/2012).

Lawhead, P., Duncan, M., Bland, C., Goldweber, M., Schep, M., & Barnes, D. (2003) “Legos, Java and Programming Assignments for CS1,” *SIGCSE’03*, pp. 47-48.

Lopez, N., Casallas, R., & Villalobos, J. (2007) “Challenges in Creating Environments for SOA Learning,” IEEE Proceedings of the International Workshop on Systems Development in SOA Environments.

Luxton-Reilly, A. and Denny, P. (2009) “A simple framework for interactive games in CS1*,” 40th ACM Technical Symposium on Computer Science Education* (Chattanooga, TN, USA, March 04 - 07, 2009). *SIGCSE '09*.

Markham, S. & King, K. (2010) “Using Personal Robots in CS1: Experiences, Outcomes, and Attitudinal Influences,” *ITiCSE’10*, Bikent, Ankara, Turkey.

McWhorter, W. & O’Connor, B. (2009) “Do LEGO mindstorms motivate students in CS1?,” *40th ACM Technical Symposium on Computer Science Education (SIGCSE ’09)*, Chattanooga, TN, March 2009, 438-442.

NetBeans (2012) *NetBeans IDE 7.0*, Retrieved May 2, 2012 from http://netbeans.org.

Pedroni, M. & Meyer, B. (2006) “The Inverted Curriculum in Practice,” *SIGCSE’06.*

Pendegrast, M. (2006) “Teaching Introductory Programming to IS Students: Java Problems and Pitfalls,” *Journal of Information Technology Education* (5:X), pp. 491-515.

Phifer, G. (2012) “Agenda for Web Computing, 2012,” *Gartner White Paper, ID: G00230296*, (accessed via the web, 5/2/2012)

Rajaravivarma, R. (2005) “A Games-Based Approach for Teaching the Introductory Programming Course,” *inroads-The SIGCSE Bulletin* (37:4), pp. 98-102.

Reed, M.W., Balogh, B.A., Miller, D.C., & Chiang, C.C. (2007) “Developing and learning web services with open source software: an experience report,” *Journal of Computing Sciences in Colleges* (22:4).

Sheil, B.A. (1981) “The Psychological Study of Programming,” *Computing Surveys* (13:1), pp. 101-120.

Stepp, M, Miller, J., & Kirst, V. (2009) "A CS 1.5 Introduction to Web Programming," *ACM SIGCSE '09*.
Stevenson, D. Wagner, P. (2006) "Developing Real-World Programming Assignments for CS1," *ITiCSE'06*, June 26-28, 2006, Bologna, Italy.
Tsai, W., Chen, Y., Cheng, C., Sun, X., Bitter, G., & White, M. (2008) "An Introductory Course on Service-Oriented Computing for High Schools," *Journal of Information Technology Education* (7:X), pp. 315-338.
Yue, K.B. (2010) "Experience on Mashup Development with End User Programming Environment," *Journal of Information Systems Education*, Vol. 21(1).
WebServiceX (2010) "XML Web Service Solutions Provider: Weather Forecast Web Service," Retrieved June 15, 2010.
Zajacova, A., Lynch, S.M., & Espenshade, T.J. (2005) "Self-efficacy, stress, and academic success in college." *Research in Higher Education, 46*, 677-698.
Zilora, S. J. (2004) "Industry-Based Web Services Project as a Classroom Teaching Tool," *ACM SIGITE'04*, Salt Lake City, UT, USA.

AUTHOR BIOGRAPHIES

Bryan Hosack is an Associate Professor of Information Systems at Illinois State University. He received his doctoral degree at Florida State University in Management Information Systems. His Bachelor of Science degree came from Purdue University and his Masters of Science in Management Information Systems from Clarkson University. His research has been published in the *Journal of the Association for Information Systems*, *Journal of Informatics Education Research*, *Decision Support Systems*, and the *International Journal of Production Research*, as well as in the proceedings of national and international academic conferences. His research interests are in the areas of decision support systems, information assurance and security, information systems education, and IT decision making simulations. Prior to pursuing his doctoral degree, he worked in the manufacturing and information systems industries.

Billy B. L. Lim is a Professor in the School of IT of Illinois State University, Illinois, USA. His research/teaching interests are in the areas of Web development technologies, Object-Oriented systems, and modern DBMSs. He has published numerous journal articles in these areas and has also presented papers in many international and national conferences. Dr. Lim has also received numerous grants from agencies such as the National Science Foundation.

W. Paul Vogt is Emeritus Professor of Research Methods and Evaluation at Illinois State University where he won both teaching and research awards. He specializes in methodological choice and is particularly interested in ways to integrate multiple methods. His books include: *Dictionary of Statistics and Methodology* (4th edition, 2011); *Quantitative Research Methods for Professionals* (2007); and *When to Use What Research Design* (2012). He blogs about research methods at: http://vogtsresearchmethods.blogspot.com/

STATEMENT OF PEER REVIEW INTEGRITY

All papers published in the Journal of Information Systems Education have undergone rigorous peer review. This includes an initial editor screening and double-blind refereeing by three or more expert referees.

ISSN 1055-3096