ISSN: 1545-679X

Information Systems Education Journal

Volume 4, Number 77 http://isedj.org/4/77/ September 18, 2006

In this issue:

Is this Course Right for You? Using Self-Tests for Student Placement

Lucia Dettori
DePaul University
Chicago, IL 60604 USA

Theresa Steinbach
DePaul University
Chicago, IL 60604 USA

Martin Kalin
DePaul University
Chicago, IL 60604 USA

Abstract: Key to student success in introductory Information Technology and Computer Science courses is the adequate mastery of prerequisite concepts. One method utilized to ensure the proper placement of students in these first courses is through self-administered computerized assessment tests. These tests were introduced as a result of a major restructuring of the undergraduate curriculum at the authors' school, the establishment of a prerequisite-light introductory sequence, the wide range of skill sets of entering freshmen, the high number of transfer students from community colleges, and the use of these courses by several graduate degrees in their prerequisite phase. Six tests have been created to test fundamental programming concepts, database manipulation and design, data analysis and development methodologies. Advisors encourage the use of these tests for initial placement. Some students have long gaps between the first and second courses in a sequence and can assess their retention of material. Graduate students planning to test out of prerequisite requirements use them to assess their knowledge prior to taking a Graduate Assessment Exams. This paper describes the development, implementation, and effectiveness of these tests as advising and assessment tools.

Keywords: placement test, assessment, self-assessment, computer-based testing, advising

---

Recommended Citation: Dettori, Steinbach, and Kalin (2006). Is this Course Right for You? Using Self-Tests for Student Placement. *Information Systems Education Journal, 4* (77). http://isedj.org/4/77/. ISSN: 1545-679X. (Also appears in *The Proceedings of ISECON 2005:* §2312. ISSN: 1542-7382.)

This issue is on the Internet at **http://isedj.org/4/77/**

The **Information Systems Education Journal** (ISEDJ) is a peer-reviewed academic journal published by the Education Special Interest Group (EDSIG) of the Association of Information Technology Professionals (AITP, Chicago, Illinois). • ISSN: 1545-679X. • First issue: 8 Sep 2003. • Title: Information Systems Education Journal. Variants: IS Education Journal; ISEDJ. • Physical format: online. • Publishing frequency: irregular; as each article is approved, it is published immediately and constitutes a complete separate issue of the current volume. • Single issue price: free. • Subscription address: subscribe@isedj.org. • Subscription price: free. • Electronic access: http://isedj.org/ • Contact person: Don Colton (editor@isedj.org)

EDSIG activities include the publication of ISEDJ, the organization and execution of the annual ISECON conference held each fall, the publication of the Journal of Information Systems Education (JISE), and the designation and honoring of an IS Educator of the Year. • The Foundation for Information Technology Education has been the key sponsor of ISECON over the years. • The Association for Information Technology Professionals (AITP) provides the corporate umbrella under which EDSIG operates.

Is this Course Right for You? Using Self-Tests for Student Placement

Lucia Dettori
ldettori@cti.depaul.edu

Theresa Steinbach
tsteinbach@cti.depaul.edu

Martin Kalin
mkalin@cti.depaul.edu

School of Computer Science, Telecommunications, and Information Systems
DePaul University
243 South Wabash Ave.
Chicago, Illinois 60604, USA

Abstract

Key to student success in introductory Information Technology and Computer Science courses is the adequate mastery of prerequisite concepts. One method utilized to ensure the proper placement of students in these first courses is through self-administered computerized assessment tests. These tests were introduced as a result of a major restructuring of the undergraduate curriculum at the authors' school, the establishment of a prerequisite-light introductory sequence, the wide range of skill sets of entering freshmen, the high number of transfer students from community colleges, and the use of these courses by several graduate degrees in their prerequisite phase. Six tests have been created to test fundamental programming concepts, database manipulation and design, data analysis and development methodologies. Advisors encourage the use of these tests for initial placement. Some students have long gaps between the first and second courses in a sequence and can assess their retention of material. Graduate students planning to test out of prerequisite requirements use them to assess their knowledge prior to taking a Graduate Assessment Exams. This paper describes the development, implementation, and effectiveness of these tests as advising and assessment tools.

Keywords: placement test, assessment, self-assessment, computer-based testing, advising

1. INTRODUCTION

Key to student success in introductory Information Technology and Computer Science courses is the adequate mastery of prerequisite concepts. Incoming student preparation has always been quite varied and the problem is amplified by the wide range of students pursuing degrees at our institution. The undergraduate population at DePaul University's School of Computer Science, Telecommunications and Information Systems (CTI) consists of traditional freshmen continuing their education immediately after high school, a high number of transfer students from surrounding community colleges, and adult students who have been working in the IT field. The graduate population consists of traditional students who have recently completed a BS which may have been IT-related, working professionals retraining for a career shift, and a significant number

of foreign students from all over the world. Both populations also include a significant number of distance learning students. Given the wide range of backgrounds and the diverse student body, it is particularly challenging for advisors to assist students in identifying the introductory course that best fits his or her skill level. It should be noted that the placement problem is not limited to the first programming course, but extends to the entire suite of introductory IT courses, and the set of prerequisite courses for our graduate degrees. This added layer of complexity called for a solution that was highly scalable, easily distributed among as many faculty as possible, and available to the students at all times without much need for close supervision.

Our solution to this challenge is a suite of self-administered computerized self-assessment tests that students can take, and advisors and instructors can suggest at any time. Tests include immediate feedback on the student's performance and an indication of which course is appropriate. The creation, deployment, and administering of the tests is streamlined through the use of an internally developed Web application. This article describes the development of the self-test application and discusses the effectiveness of these tests as advising and assessment tools.

2. BACKGROUND AND RELATED WORK

In the early months of 2004 through the work of an ad-hoc curriculum task force, a significant restructuring of the undergraduate curriculum took place (Besana et al., 2005). Most degree programs were revamped and shifted from a traditional programming-first approach to a breadth-first funnel approach in line with the model later suggested in the Computing Curricula 2004 draft (Computing Curricula, 2004). A number of shared introductory courses were created, to be taken by most undergraduate students in CTI in their first two years, independent of the major they were pursuing. One of the recommendations of the task force was to create a suite of placement tests to assist advisors and students in choosing the appropriate introductory courses. The task force ruled out creating traditional placement tests to be taken along with the Mathematics and English tests administered by the University and opted for an internally-developed, non-credit-granting system for several reasons. Placement tests that give academic credit need to be proctored, frequently updated, and integrated in the existing system of the University. This would have significantly delayed the deployment of the tests, and would have made their monitoring unnecessarily complex.

As mentioned above, in addition to traditional undergraduates, CTI serves a large number of transfer students. These students begin taking courses at CTI at different times during a given academic year making it difficult to coordinate a common testing day. While DePaul University maintains several articulation lists with feeder schools in the area, it is not always possible to match individual courses, and, given the ever evolving nature of IT curricula, it is especially challenging to keep the lists fully up to date. In addition to course credit, students often have a working knowledge of the material making it difficult to advise them on which course is suitable for them. A similar situation is common at the graduate level. CTI allows rolling applications for the graduate program and accepts students without prior IT training as long as they master the content covered in the set of prerequisite courses. Students who have not completed similar course work may test out of prerequisite requirements by taking the corresponding Graduate Assessment Exam (GAE). This pass/fail, proctored exam is offered only two times during the first week of each month and is manually graded by a faculty member. In the absence of academic credit or successfully passing the GAE, advisors must rely on their evaluation of the student's work experience, or the student's claim of his or her knowledge. Giving students (and advisors) the option of self-evaluating their skill level at any time allows students and advisors to optimize their curriculum on a course by course basis.

Other institutions have created placement exams for their CS1 courses, e.g. Northwestern, Caltech, and University of Texas at Dallas. However, these exams are limited to placement in introductory programming courses, are geared primarily towards traditional incoming freshmen, and require significant time investment on the department side to develop and administer. The novelty of our approach lies in the comprehensive nature of the solution. This approach covers

a dozen courses, has flexibility of delivery, and a straightforward developing process which allows the deployment of tests in a matter of hours.

This article does not aim to analyze the specific content of each test and assess its effectiveness. Rather it emphasizes the portability of our development and deployment solution. For an analysis of the optimization of test questions for CS0/CS1 placement tests see SO et al. (2005) and Dierbach et al. (2005), and references therein.

3. DEVELOPMENT PROCESS

This section describes the process by which the self-test initiative was implemented at CTI. The ambitious goal of creating a dozen self-tests required a distributed, scalable, and user-friendly solution. In order to streamline the creation and maintenance of the self-tests as much as possible it was decided to create a simple Web application to administer the tests, and an intranet Web editor faculty could use to input test questions which would automatically generate the tests. Since the tests are mainly used for self-evaluation, it is important to provide students with immediate feedback. To use the tests as an advising tool during an advising session, it is key to keep the length of the test short. Students should be able to easily access the Intranet, take the test and immediately report the scores. No academic credit is earned by taking the test so it is not necessary to thoroughly test all possible concepts covered in a course. Testing the main topics is generally sufficient to judge if a student should take the course or a more advanced one. To fulfill these goals a uniform format for the tests was chosen: approximately 20 multiple choice questions in a simple, untimed Web application that generates an immediate report of the number of correct answers. An authentication step is built into the application but is currently not used since there is no immediate interest in tracking how individual students perform. We are mainly interested in aggregate data.

The introductory programming course was chosen to be the pilot course for the self-test initiative. A two-quarter Java sequence, CSC 211 and CSC 212, is required of several degrees. CSC 211 assumes familiarity with basic programming concepts such as using variables, branching and looping, and use of functions. These concepts are covered using JavaScript in the prerequisite course IT 130. It is often the case that students have some knowledge of these concepts without having taken the required prerequisite course and possibly using a different language. A basic programming concepts self-test allows students and advisors to evaluate if such knowledge is adequate for the student to enroll in CSC 211.

While the self-test application was being developed, one of the authors coordinated a small group of faculty who produced a set of multiple-choice questions on basic programming concepts. A test was generated by the application. The test was then sent to all faculty for comments and validation.

Within the first six months, six additional self-tests were generated. These tests help students decide if they need to take the following courses: IT 130 The Internet and the Web, IT 215 Analysis and Design Techniques, IT 223 Data Analysis, IT 240 Introduction to desktop databases, SE 325 Principles and Practices of Software Engineering, and CSC 211 Programming in Java I. An example of such test is provided in the Appendix. Students who pass this test are encouraged to talk to their advisors about substituting IT 130 with a more advanced course. For a complete list of self-tests, see https://securestore.cti.depaul.edu/tests/index.html. Five additional tests are being developed for CSC 212 Programming in Java II, CSC 261 Programming in C++ I, CSC 262 Programming in C++ II, CSC 373 Computer Systems I, and CSC 374 Computer Systems II. Students were made aware of the self-tests through email, the graduate and undergraduate newsletters, meetings with faculty advisors and other advising events. Links to the testing Website were added to the course description page and from the programs description page for degrees requiring any of these courses.

4. THE SELF-TEST APPLICATION

The software developed to support of the self-test system consists of three main parts:

1. A Web-based editor for writing tests.
2. A collection of utilities to generate a test and a corresponding answer sheet from

input documents of various formats such as plain text, HTML and Microsoft Word.

3. A Web-based, server-side script that grades a submitted test against an answer sheet and produces a test summary along with a recommendation for the test taker.

This section briefly describes each part in turn.

The Web-Based test editor

The editor is WYSIWYG in style and supports persistence so that a test author can save and later retrieve a partially written test. The editor presents the author with a standard XHTML form for entering questions, a list of candidate answers per question, and a grading scheme together with appropriate advisory messages for the test taker. A partially completed test may be submitted for saving and a completed test may be submitted for deployment. In either case, a C# script under ASP.Net handles the submitted document.

Software Utility for Test Generation and Deployment

There is a collection of small software utilities, written in Perl, that transform an author-submitted test into a standard XHTML document and an accompanying answer sheet. The utilities also can deploy the XHTML test document and the answer sheet on the server machine. A variety of utilities is needed because authors may submit a self-test in different formats and styles. For example, some authors submit a basic HTML document with the test answers embedded therein. A Perl utility extracts the questions and candidate answers from the HTML, generates a standard XHTML document with the answers removed, and generates a separate answer sheet, which in turn includes advisory messages for test takers. Similar Perl utilities handle the generation of the standard XHTML test document from plain text and Word document versions. The overall goal is to make the writing of self-tests as easy and straightforward as possible.

The Test-Grader Script

The self-test software system has the capability of user authentication and authorization but, at present, these features are turned off in order to encourage students to use the self-test system as often and as widely as they like. A test taker is presented with a list of available self-tests, each a standard XHTML document with questions and candidate answers. When the test taker submits a test, the server-side script *Test-Grader.ashx* does the following:

- Extracts the test taker's answers from the submitted test.
- Loads the corresponding answer sheet from a local file on the server.
- Grades the test against the answer sheet.
- Generates a test summary document in XHTML for the test taker.

The summary document lists the number of questions in all, the number attempted, the number correctly answered together with the correct answer for any missed question, a list of required but missed questions, a summary score (e.g., 16 out of 20), and an advisory message that, as noted, always includes a recommendation that the student meet with a faculty advisor to discuss the test results.

The script is written in C# under ASP.Net as a so-called *HTTP Handler.* The undocumented source is roughly a page in size. The script maintains a log of test results and can generate email summaries for faculty who are interested in particular test results.

5. CONCLUSIONS AND FUTURE WORK

The self-administered computerized assessment tests for introductory information technology and computer science courses were created in part as a response to a need for appropriate course placement of incoming students. It is critical that students be allowed to have a more challenging experience in an advanced course if warranted or this quantifiable measure can act as a "wake-up call" to the student to realistically plan their course selections. An analysis of the tests log file during the first few months of existence shows a significant numbers of students performing well in the test, as well as several students not performing at a passing level. This suggests that the test is being used by the two intended segments of the student population. This behavior is confirmed by the experience of one of the authors who has consistently used the self-

tests as an advising tool. It is particularly challenging for students to correctly judge a priori if their level of competency in a given subject is adequate preparation for an advanced course. Students tend to believe they are better prepared than they really are. Being able to point to and immediately discuss the result of the self-test proved a convenient and convincing advising tool. It is equally challenging for the advisor to be able to judge if a student should be encouraged to replace an introductory course with a more advanced one, particularly if the advisor is not completely familiar with the subject. The self-test results and recommendation have proven to be a valuable, at times critical, piece of information to advise the student. It is self evident that for this approach to be effective a traditional delivery of placement tests would be completely inadequate. It is critical that the self-tests be available at all times and provide an immediate feedback.

Another key use of self-test has been in multiple-quarters course sequences. It has been our experience that students who do not enroll in consecutive quarters for sequential courses use the self-test corresponding to the first quarter of the sequence to judge if their review of the material is sufficient to take the second quarter. One of the authors also has repeatedly used the CSC 211 (Programming in Java I) test for an initial assessment of class-wide knowledge in the follow up course CSC 212.

Several graduate students have reported using the computerized self-assessment tests as a study aid in preparation for a Graduate Assessment Exam. Prior to implementation the only tool available in preparation was a list of topics and reference textbooks.

The use of this self-administered computerized testing system is not limited to self-assessment for advising and placement purposes. For example, a slightly modified version of it has since been used in a pilot project to collect data for the purpose of assessing the learning goals for the Bachelor of Science in Computer Science (BS in CS). Assessment is carried out through a series of questions in the Data Structure course which all students in the BS in CS have to take. However, several graduate students also take the course to satisfy prerequisite requirements. In the past, a professor would have to manually eliminate the tests completed by graduate students. The problem is solved by an implementation of the assessment test system with the authentication module turned on. Only the tests from undergraduate students are kept for assessment purposes. Being able to triage the students and have the data and the grading immediately available in digital form proved critical to speed up the data evaluation procedure.

In the future we plan to add a timing module to the system to control how much time a student has to complete the test, and to allow delayed deployment of a test. The enhanced self-assessment application will then be integrated into the course management system used by our distance learning programs to support delivery of online quizzes and other evaluations. Once all needed tests are created, we will focus on studying the effectiveness of the individual test in assessing a student's familiarity with a particular subject.

6. ACKNOWLEDGEMENTS

The authors would like to thank the following colleagues for their contributions to the drafting of the current self-tests: Gian Mario Besana, Gerald Gordon, Jane Huang, Iyad Kanj, Steve Lytinen, Will Marrero, Craig Miller, Daniela Raicu, Marcus Schaefer, and Raffaella Settimi.

7. REFERENCES

Besana, G., Dettori L., Steinbach, T., 2005, An invitation to IT: a new perspective on the first two years, submitted *SIGCSE 2006.*

The Computing Curricula 2004. Retrieved August 19, 2005, from http://www.acm.org/education/Overview_Draft_11-22-04.pdf

Soh, L., Samal, A., Person, S., Nugent, G., and Lang, J. 2005. Designing, implementing, and analyzing a placement test for introductory CS courses. *SIGCSE Bull.* 37, 1 (Feb. 2005), 505-509.

Dierbach, C., Taylor, B., Zhou, H., and Zimand, I. 2005. Experiences with a CS0 course targeted for CS1 success. *SIGCSE Bull.* 37, 1 (Feb. 2005

Appendix

IT 130 Self-Test

The purpose of this CTI placement test is to help us place you in the right programming course. The current test is to help determine whether you should by-pass IT 130 (a prerequisite for the first programming course in Java) and take CSC 211 (the first programming course in Java) or some other programming course. Take this test by yourself and note the score. Working with your advisor, we can then choose the course that best works for you.

Top of Form
In the URL **ftp://download.depaul.edu/admissions/application.pdf** the protocol is:

- **ftp**
- **ftp://download.depaul.edu**
- **application.pdf**
- **download.depaul.edu/admissions/application.pdf**

Which of the following statements about the IP protocol is true?

- It routes a single packet to a destination address .
- It routes a sequence of packets to a destination address.
- It determines the IP address of a destination URL.
- It resends a packet if it is lost.

The **<b>** tag in the HTML code **<b><b>** will

- Make the picture appear in bold
- Make the picture larger
- Make the picture brighter
- Do nothing

Which tag allows you to specify a cell in a table?

- **<tc> </tc>**
- **<td> </td>**
- **<tr> </tr>**
- **<cell> </cell>**

Which of the following code samples turns the image **"cube.gif** (within the sub-folder **images**) into a clickable, hyperlinked picture?

- ****
- ****
- ****
- **<img src="images/cube.gif" border="2"></img>**

You have organized your files into two folders: **web_pages** and **web_images**. The folder **web_images** contains an image **"theseus.jpg"** which should appear on the page **main.html** in the folder **web_pages**. Which is the correct code:

- ****
- ****
- ****
- ****
- ****

Your page **myth.html** contains a section header **"King Arthur"**, which you named using the following code: **<h1>King Arthur</h1>**. How do you link to that section in the page **main.html** in the same folder?

- **King Arthur**
- **King Arthur**
- **King Arthur**
- **King Arthur**
- **King Arthur**

Which attribute of the **form** tag allows you to specify the program to be performed when submitting the form?

- **action**
- **program**
- **submit**
- **name**
- **asp**
- **method**

How can you implement multiple selection in a form?

- **<select>/<option>** and **<input type="checkbox">** both allow multiple selection
- **<select>/<option>**; **<input type="checkbox">** only allows checking a single box
- **<input type="checkbox">**; **<select>/<option>** allows only one option to be selected
- both **<select>/<option>** and **<input type="checkbox">** only allow single selection, you need the **<input type="radio">** tag for multiple selection

If you display the following web-page:

```
<html>
   <head>
    <style>
        body {color:blue;}
        h1, h2 {color:green}
    </style>
   </head>
 <body>
    <h1 style="color:red">The beginning </h1>
  </body>
</html>
```

The words "The beginning" will be displayed in:

- The default color (black for most browsers)
- blue
- green
- red
- turquoise

The user-centered design (UCD) process begins first by answering the question:

- What is the competition doing?
- Who will be using the product?
- What sample scenario should be created?
- What other means target users have for completing their tasks?

A low fidelity prototype is:

- Close enough to a final product to be able to examine usability questions in detail
- Used to make strong conclusions about how behavior will relate to use of the final product
- Used to quickly produce the prototype and test broad concepts
- Used to understanding relationships across a broad system and for showing the range of abilities of a system

Consider the following **JavaScript** code:

```
<script language="JavaScript">
        var x,y;
        x = window.prompt("First value");
        y = window.prompt("second value");
        window.alert("Output: "+x+y);
</script>
```

If we enter **1** for the first value, and **2** for the second value, then the program will:

- Bring up a pop-up box with the text **"Output: 3"**
- Write the text **"Output: 3"** on the web-page

- Bring up a new browser window with the text **"Output: 3"**
- Bring up a pop-up box with the text **"Output: 12"**

Consider the following piece of **JavaScript** code:

```
<script language="JavaScript">
        var x;
        function create_img(num){
                document.write('<img src="medea'+num+'.jpg"/>');
        }
        x= window.prompt("Enter a number between 1 and 5:");
        create_img(x);
</script>
```

If the user enters **3**, the following will be written to the web-page:

- ****
- ****
- ****
- **''**

Which of the following buttons will change the background color to red when clicked?

- **<input type="radio" onClick="document.write('bgColor="red";';"/>**
- **<input type="radio" onClick="backgroundColor='red';"/>**
- **<input type="radio" onClick="document.bgColor='red';"/>**
- **<input type="radio" onClick="<body background='red'></body>/>**

According to the following fragment of JavaScript code

```
if( ( ((year % 4) == 0 ) &&
    ( (year % 100) != 0)) || ((year % 400) == 0) )
{
   window.alert("Leap year");
}
else
{
  window.alert("Not a leap year");
}
```

(where **year%4** is the reminder of **year** divided by **4**)

- **2004** is a leap year, but **1900** is not
- Neither **1900** nor **2004** are leap years
- Both **1900** and **2004** are leap years
- All year divisible by **4** are leap years

The code
<input type="text" size="2" name="state"> IL </input>
appears in a form named **myForm**. What JavaScript code will change the state to **NY**?>

- **myForm.state = NY;**
- **myForm.state.value = "NY";**
- **state = "NY"**
- **document.state = "NY"**

Consider the following piece of JavaScript code.

```
<script language="JavaScript">
        var counter;
        for(counter = 1; counter <10; counter ++){
                width = 5+2*counter * counter;
                document.write('<hr width="'+width+'">');
        }
```

Which of the following pictures is the result of this code?

- 
-
- 
- 

Submit Test for Grading

Bottom of Form