# Improving the Learning Environment in Beginning Programming Classes: An Experiment in Gender Equity

**Thad Crews**
**Jeff Butterfield**
Computer Information Systems
Western Kentucky University
Bowling Green, KY 42101, USA
thad.crewsii@wku.edu tjbutterfield@yahoo.com

**ABSTRACT**

The under-representation of women in computing is well documented. This imbalance creates numerous problems including challenges of staffing and equity as well as more subtle problems such as a lack of balanced perspective on innovation and social implications. While there is universal agreement that females are equally capable of succeeding in a technical arena, there is a diversity of opinions as causes and solutions to this problem. One particularly interesting theory proposed by DePalma is based on trend differences between computing and other science disciplines. DePalma suggests that the positive trends in other science fields can likewise be achieved in computing if similar science pedagogies are implemented. This paper reports on an empirical study conducted to test some of DePalma's recommendations. While our investigation is preliminary, it does provide positive support for the theory that techniques that work in other science disciplines may also prove effective in computing. The results of our findings are presented along with a discussion and implications for future work.

**Keywords**: Gender equity, female retention, information technology, introductory computer programming, logic and design

## 1. INTRODUCTION

### 1.1 General Appearance

A report by the Presidential Information Technology Advisory Committee suggests that there is a critical need for a diverse IT workforce if the United States is to meet the challenges of the information age (PITAC, 1999). It is a well-documented phenomenon that the Information Technology (IT) industry suffers from a shortage of females entering the profession at both post-secondary institutions and in the workforce (Camp, 1990; Freeman, 1999; NSF, 2000).

Although the number of technology jobs in the United States shrank by an estimated 5 percent in 2001-2, falling from 10.4 million workers to 9.9 million as the economy contracted, hiring managers now say they may be unable to fill as many as 600,000 tech jobs in 2002-3, according to a survey by the Information Technology Association of America (ITAA, 2002). Survey respondents indicated that more than 1.1 million tech jobs will be available as the economy recovers, but they predict they will be unable to fill some 578,000 of those positions. Part of the discrepancy arises from a consistent "gap" between supply and demand of IT workers. The Commission for the Advancement of Women and Minorities in Science, Engineering and Technology Development argues that much of this shortfall could be satisfied through programs that increase the number of women that enter the field (Commission, 2000).

The problems associated with encouraging more women to enter the information-technology field are complex and multi-faceted. Research suggests the need for incremental advancements in many areas, including removing gender bias in computer software, increasing female access to, and experience with computers, increasing the number of female role models in the profession, and even developing more "girl-friendly" computer games (Davies 2000; Thom, 2001; Woodfield 2000).

One of the most direct recommendations for increasing the percentage of females in the discipline was presented by DePalma (2001). He argues that in spite of talk about 'math anxiety', women earn almost half of the undergraduate degrees in math. His hypothesis is

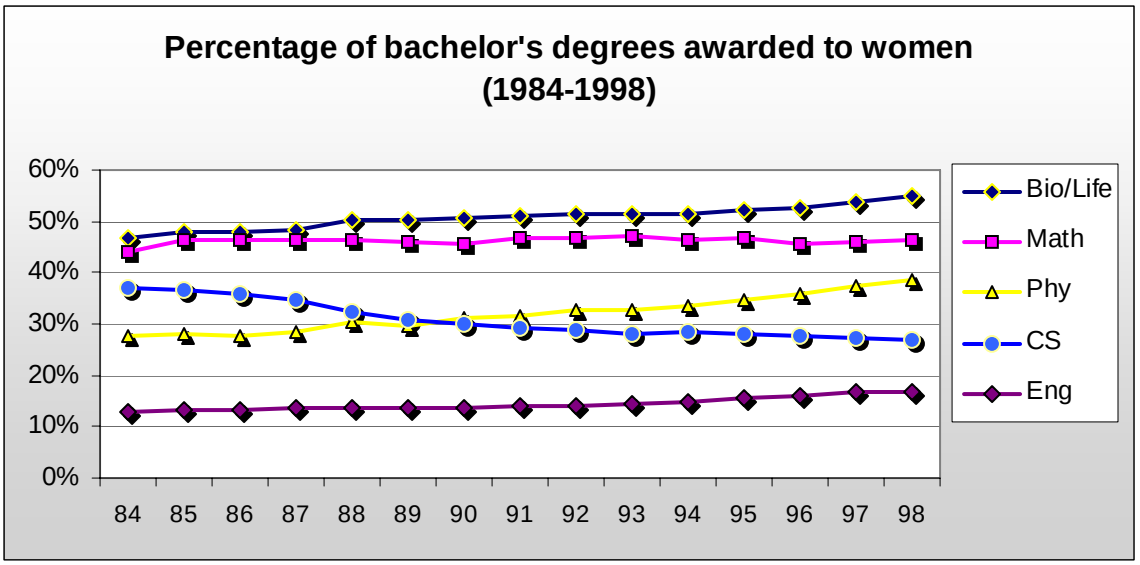


**Figure 1: Percentage of bachelor's degrees awarded to women**

that the precision of mathematics makes the field attractive to women, while the ill-defined nature of the computing field tends to drive them away. DePalma offers five suggestions for introducing female students to computers in general and programming specifically:

First, teach programming to any girl with an aptitude for symbolic manipulation. Second, when teaching programming, keep it as close to pure logic as possible (minimize reliance on software packages, user interfaces, text editors, etc.). Third, if at all possible, try to teach programming without microcomputers (try to decrease the distraction from the hardware). Fourth, treat programming languages as a notational system. Finally, keep programs short (following the Mathematical drill model that emphasizes many small incremental problems instead of one or two major projects).

This study seeks to empirically test some of DePalma's recommendations and try to determine what, if any, difference they might make on teaching entry-level computing students – both female and male.

## 2. LITERATURE REVIEW

The problem of gender equity in IT is decades old. The Department of Education in the United States shows that from 1983-84 through 1997-98 there was a 28% decrease in the percentage of women earning bachelor's degrees in computing (NCES, 2001). The Department of Education data is particularly revealing as it also shows a steady increase of women choosing to study in other science fields during that same time. From 1984 to 1998 there was an 18% increase in the percentage of women earning bachelor's degrees in Biology/Life Sciences, a 31% increase in women earning Engineering degrees, a 6% increase in women earning Math degrees, and an 11% increase in women earning Physics degrees. The contrast between the steady declines in the computing discipline compared to the steady growth in the other disciplines is shown in Figure 1.

The prognosis for women at the beginning of their college experience is no better. The Cooperative Institutional Research Program (CIRP) is the nation's largest and oldest empirical study of higher education, involving data on some 1,800 institutions and over 11 million students. The annual CIRP Freshman Survey provides normative data on each year's entering college students. Data from the 2000 survey suggests that women lag far behind their male counterparts when asked about their computing self-confidence (CIRP, 2001). Women are half as likely as men are to rate their computer skills as "above average" or "top 10%" relative to people their age (23.2 percent among women, versus 46.4 percent among men). This gap in self-confidence likely contributes to the fact that men are five times more likely than women are to pursue careers in computer programming (9.3 percent of men, versus 1.8 percent of women). The gap among the 2000 freshmen is the largest in the history of the survey (see Figure 2).

Just as there is no silver bullet for taming the complexity of large software projects (Brooks, 1986),

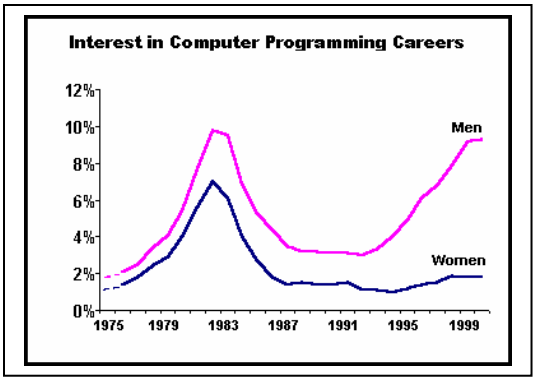


**Figure 2: CIRP freshman survey 2000 results**

Just as there is no silver bullet for taming the complexity of large software projects (Brooks, 1986), there is likewise no simple solution to this problem (Alper, 1993; Camp, 1990; Davies 2000; Freeman, 1999; Humphreys, 2002; NSF, 2000; Thom, 2001; Woodfield 2000). According to Camp (2000), differences in past experiences with computers are likely to cause female students in introductory computing courses to, "…be left behind on the very first day unless the course is designed for those less experienced students in the class."

This study specifically seeks to empirically examine some of DePalma's specific recommendations for improving the recruitment and retention of female students in the computing fields. It should be noted that this is an preliminary study that seeks only to validate DePalma's basic hypotheses. The next section introduces the methodology used in the study. This is followed with the results and analysis of the data. The report concludes with a discussion of implications and future directions of this research effort.

## 3. EXPERIMENTAL STUDY

This study specifically seeks to test three of DePalma's (2000) five recommendations for improving the success of female students in beginning programming classes: (1) teach programming by emphasizing the logic and design rather than focusing exclusively on the specific syntax of a language (e.g., Java, C++, etc.); (2) make efforts to decrease the distraction caused by hardware-specific demands (i.e., minimize the overhead associated with using the computer, its operating system and complex program GUIs); and (3) require subjects to complete numerous small programs – each progressively more difficult than the last. This allows each student to methodically work through a common body of problems that compensates for differences in individual experience with computers. Consistent with DePalma's final recommendation, the programming language is treated as a tool for problem solving rather than being the learning objective itself.

### 3.1 Experimental Test Bed

Several of DePalma's recommendations deal with simplifying the early experiences with computer programming and minimizing (or eliminating if possible) distractions from the hardware, operating system and programming environment. This idea could be taken to the extreme of teaching programming using only a pencil and paper. Many years ago when computers were scarce, this was the pedagogy of choice for many schools. However, flow-charts, pseudo-code or hand-drawn programs must be manually traced to identify errors – a slow and laborious process. Studies indicate an instructional limitation with manual approaches that do not provide immediate feedback to learners (Gould, 1975; Mayer, 1975; Wittrock, 1974).

As a compromise between simplicity and the need for feedback on one's solutions, the Visual-Logic (VL) programming tool was used in this study. Visual-Logic is an east-to-use development environment that was designed to be easy for novice programmers to learn and use. VL avoids the complexity of a high level programming language by utilizing flowcharts to represent logical solutions. These solutions are created by adding, deleting and moving flowchart elements using an intuitive point-and-click, drag-and-drop interface. VL emphasizes patterns of logic and design, abstracts out the details of different hardware, and supports a broad range of programming and problem-solving activities. Figure 3 shows the flowchart design window with a one possible solution to an overtime wage-calculation problem.

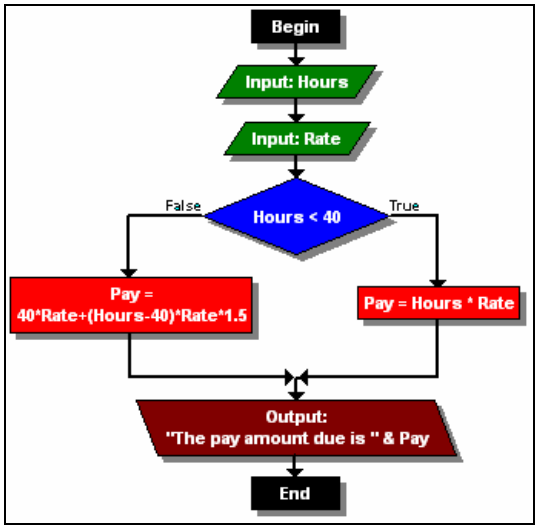


**Figure 3: Visual-Logic solution to overtime problem**

The most significant contribution of Visual-Logic, however, is its ability to execute the logical flowcharts. When as student wants to check the validity of their program design, VL is able to immediately interpret and execute it. This feature provides students both the

simplicity of a flow chart and the immediate feedback that previously was only available with a traditional high-level language. When using the Visual-Logic tool, students may design, develop, execute, test and evaluate computer programs without any prior exposure to a formal computer language.

Creating the traditional "Hello World" program is as easy as selecting an output flowchart element and entering the greeting. There is no need for the student to understand libraries, I/O streams, or anything beyond the logic of the solution. When the student selects "Run" the Visual-Logic System executes the flowchart, including input dialog prompts and the appropriate data in an output window. Visual-Logic supports a broad range of activities (e.g., input, assignment, output, conditions, loops, variables, procedures, etc.) while minimizing the specifics of hardware and language syntax.

### 3.2 Experimental Design

The objective of this study is to examine DePalma's recommendations for teaching introductory computer programming to see if they have sufficient validity to merit additional research.

**Procedure:** Students assigned to the control condition were trained in VB using what Brusilovsky (1997) defines as a 'traditional' approach. This pedagogy used a VB manual for a text, a lecture format in class, and several VB programming projects as homework. Students assigned to the treatment condition spent the first five weeks of the semester learning problem-solving approaches and programming logic. A series of short programming activities using Visual-Logic were assigned as homework. No Visual Basic instruction was provided to the treatment subjects during this time. After the first five weeks, the treatment group began using the same pedagogy and materials as the control group sections. This research design model, shown graphically in figure 4, was drawn from the literature (Schneider, 2001; Shackelford, 1999). The learning measurement occurred for both groups during the 16th week of the semester as part of a regular class meeting.

**Participants:** The subjects for this study were 73 undergraduate students (53% male, 47% female) at a comprehensive public university with a traditional 16-week semester. The courses were 200-level introductory Visual Basic (VB) programming. Two professors worked cooperatively to teach these classes as part of the project. Both instructors have much-higher-than-average teaching evaluations and employed common teaching methods for the experiment. Both instructors coordinated textbook and course content except for the treatment. Students in the courses came from different backgrounds and had different majors. Subject assignment to class sections was quasi-random as the students registered for their classes independent of the study.

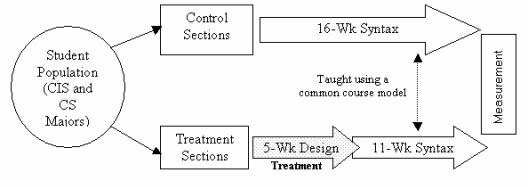


**Figure 4: Research Process**

To guide this work a set of three hypotheses were formulated from the recommendations:

***Hypothesis 1***: Female students in the treatment group will demonstrate a superior understanding of programming logic compared to female students in the control group.

***Hypothesis 2***: All students in the treatment group will demonstrate a superior understanding of programming logic compared to all students in the control group.

***Hypothesis 3***: Differences in learning outcomes between male and female students in the treatment group will be less than differences between male and females in the control group.

### 3.3 Methodology

**Apparatus:** To measure the subjects' understanding of program logic, flow and syntax, the experiment followed Scanlan's (1989) design. This methodology presents subjects with three computer programs of varying difficulty, classified as simple, medium, and complex, shown in Figure 5. Answer sheets were constructed containing three data sets of initializing values and space for participants to write down the calculated output for each data set. In addition to the solution, subjects also noted the time when they completed the three data sets, and self-assigned a confidence ranking on a 5-point scale (1 = Not Confident, 5 = Very Confident) for each of their solutions.

To assess learning outcomes, participants were provided the simple program with a randomly assigned answer sheet. The data set for each of the problems was nearly identical, but varied slightly to prevent contamination due to cheating. Participants determined the output for each of the three data inputs provided on the answer sheet. All participants were given sufficient time to complete each task before proceeding. This process was repeated for the medium and the complex programs.

**EASY PROBLEM**

```
If A < D Then
   Print D
Else
   Print A
   If B < C Then
      Print C
   Else
      Print B
   End If
End If
```

**MEDIUM PROBLEM**

```
Do While A < 4
   If A < D Then
      Print B
   Else
      Print A
      If C < B Then
         If A < B Then
            Print C
         Else
            Print D
         End If
      Else
         Print A
      End If
   End If
   A = A + 1
Loop
Print C
```

**COMPLEX PROBLEM**

```
Do While B <= 6
   If D > B Then
      Do While C >= 2
         Print C
         C = C - 1
      Loop
      If A < B Then
         Print D
      End If
   Else
      Print B
      If B < C Then
         If A < B Then
            Print A
         Else
            Print B
         End If
         Print C
      Else
         Print D
         If A < B Then
            Print B
         End If
      End If
   End If
   B = B + 1
Loop
Print C
```

**Figure 5: Simple, Medium and Complex Problems**

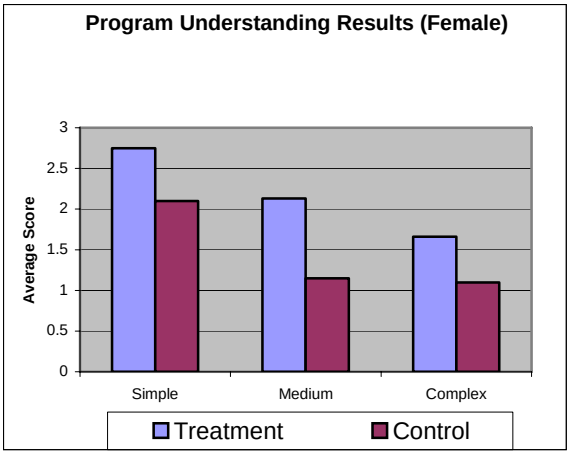


**Figure 6: Hypothesis 1 Results (Females)**

**Scoring:** Individual correctness scores were determined by awarding one correctness point for each of the three outputs that were correct per solution sheet. Half credit was given to outputs that were correct through the initial loop iteration, but incorrect at some later point. The time data was recorded as the number of seconds required for determining the outputs. An overall confidence score was calculated from the responses on the 5-point scale. This grading process was the same for each of the six answer sheets provided by each participant.

## 4. RESULTS

Hypothesis 1 suggests that female students in the treatment group will demonstrate a superior understanding of programming logic over female students in the control group. Support was found for Hypothesis 1 as females in the treatment group scored significantly better than females in the control group on all three programs.

The simple program contained a nested-if statement and no loops. For this program, the treatment female group average score was 2.75 and the control female group average score was 2.10, which was a significant difference when examined with an Analysis of Variance between groups test (ANOVA) ($p <= .01$). The medium program contained two nested-if statements inside a single loop. For this program, the treatment female group score was 2.13 and the control female group score was 1.15, a significant difference ($p <= .01$). The complex program contained two loops and four conditions. Once again, the treatment female group scored significantly better than the control female group (1.66 to 1.10, $p <= .01$). The results for all three programs are shown in Figure 6.

Hypothesis 2 suggests that all subjects in the treatment group (male and female) will demonstrate a superior understanding of programming logic over all students in the control group. Support was found for Hypothesis 2 as the treatment group scored significantly better than the control group on all three programs. The treatment group achieved scores that were significantly higher than those of the control group for all three programs, simple (2.68 to 2.18, $p <= 0.01$), medium (2.28 to 1.50, $p <= 0.01$) and complex (1.74 to 1.10, $p <= 0.01$). The results are displayed in Figure 7.

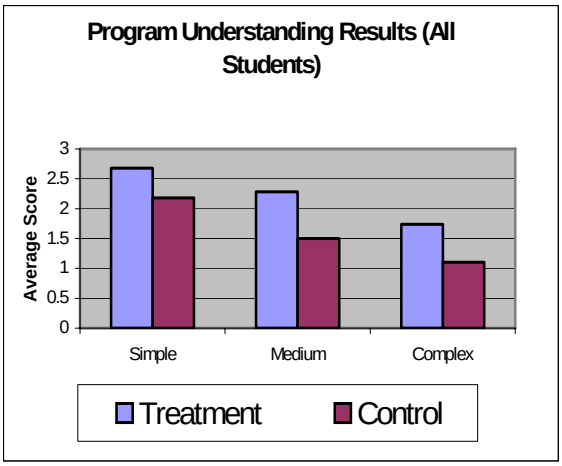


**Figure 7: Hypothesis 2 Results (All Students)**

Hypothesis 3 suggests that differences between male and female students in the treatment group will be less than differences observed between male and females in the control group. Again the hypothesis was supported by the data. An analysis of total scores for all three problems shows that both male and female students from the treatment group outperformed their counterparts in the control group. It is equally important to note that the data also suggests that the difference between sexes was quite small for the treatment group (6.53 to 6.77, 3.5%). The difference between sexes in the control group is much greater (4.35

to 5.07, 14.2%). Both the improvements resulting from treatment and the reduction in gender differences can be seen in Figure 8.

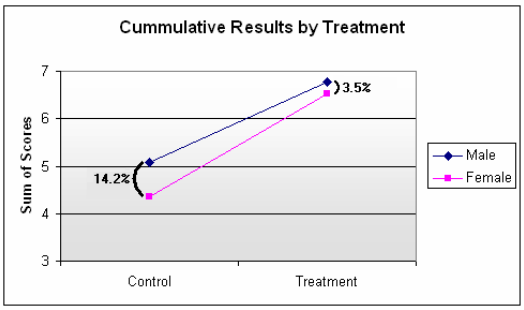


**Figure 8: Gender Differences Between Groups**

The findings of this preliminary study support DePalma's (2000) recommendations for introducing computer programming to female students. Specifically, female students' ability to work with program logic improved and the performance gap between the female and male students decreased. The pedagogical approach is also helpful for male students.

## 5. DISCUSSION AND IMPLICATIONS

The findings of the study are encouraging and suggest the need for additional research. The teaching approach that was used in this study (Visual-Logic) was successful at improving the learning outcomes for the students in the treatment condition. The approach was also well-received by the students. A de-briefing survey revealed that students in the treatment group found Visual-Logic easy to learn and use. By the end of the 5-week introduction, these students had already established their ability to be successful in the class.

A number of students in the control group indicated that the Visual Basic programming environment was intimidating, complex and difficult to learn. In contrast, none of the subjects in the treatment group made such a report – in spite of having less time to learn VB.

The problem of declining female participation in the computing disciplines is the subject of much concern and speculation. Many of the proposed solutions involve activities outside of the control of the classroom, such as presenting young girls with gender-friendly computer games and increasing the number of tech-savvy teachers (Cohoon, 2002; Wardle, 2002). These, and many other suggestions may prove beneficial over time. However, the data suggest that immediate improvements in overall student performance – and most notably, female performance – can be realized through basic changes to the teaching approach and content.

DePalma's suggestions are worthy of additional study and exploration. The choice of Visual-Logic as a means of emphasizing logic and de-emphasizing language syntax is but one way that these goals can be met. Other pedagogical approaches and educational technologies may also prove successful at improving learning outcomes for novice programmers.

By leveling the playing field in this way, instructors are able to create an environment where women succeed at a rate consistent with their male counterparts. It is hoped that this early success encourages other researchers, teachers and IT professionals to identify and develop methods for improving the learning environment for all individuals who may be interested in pursuing a career in Information Technology.

## 6. REFERENCES

Alper, J. (1993). "The Pipeline is Leaking Women All the Way Along," Science, 260, 16, pp. 409–411.

Brooks, F. (1986). "No silver bullet—essence and accidents of software engineering," Information Processing 86, H. J. Kugler, Ed. Amsterdam: Elsevier Science (North Holland), pp. 1069-1076.

Brusilovsky, P., Calabrese, E., Hvorecky, J., Kouchnirenko, A., and Miller, P. (1997). "Mini-languages: a way to learn programming principles," Education and Information Technologies, 2, pp. 65–83.

Camp, T. (1990). "The Incredible Shrinking Pipeline," Communications of the ACM, 33(11), pp. 47-57.

Camp, T. (2001). "Women in Computer Sciences: Reversing the Trend," Syllabus, vol. 15, no. 1.

Cohoon, J. (2002). "Recruiting and Retaining Women in Undergraduate Computing Majors," SIGCSE Bulletin, Vol. 34, No. 2, pp. 48-52.

Commission on the Advancement of Women and Minorities in Science, Engineering, and Technology Development, (2000), "Land of Plenty: Diversity as America's Competitive Edge in Science, Engineering and Technology," Report to Congress: September, 2000.

Cooperative Institutional Research Program (CIRP), (2001). "The American Freshman: National Norms for Fall 2000—Executive Summary" < http://www.gseis.ucla.edu/heri/heri.html > (accessed June 22, 2002).

Countryman, J., Feldman, A., Kekelis, L., and Spertus, E. (2002). "Developing a Hardware and Programming Curriculum for Middle School Girls," SIGCSE Bulletin, Vol. 34, No. 2, pp. 44-47.

Cukier, W., and Shortt, D., and Devine, I. (2001). "Gender and Information Technology: Implications of Definitions," Proc. Information Systems Education Conference, Cincinnati.

Davies, A., Klawe, M., Mg, M., Nyhus, C., and Sullivan, H. (2000). "Gender Issues in Computer Science Education," Proc. National Inst. Science Education Forum, Detroit.

De Palma, P. (2001). "Why Women Avoid Computer Science," Communications of the ACM 44, 6, pp. 27–29.

Freeman, P. and Aspray, W. (1999). "The Supply of Information Technology Workers in the United States," Computing Research Association Report, http://www.cra.org/reports/wits/cra.wits.html (accessed June 13, 2002).

Gorriz C. and Medina C. (2000). "Engaging girls with computers through software games," Communications of the ACM 43, 1, pp. 42–49.

Gould, J. (1975). "Some psychological evidence on how people debug computer programs," International Journal of Man Machine Studies, 7, pp. 151-182.

Humphreys, S. and Spertus, E. (2002). "Leveraging an Alternative Source of Computer Scientists: Reentry Programs," SIGCSE Bulletin, 34, 2, pp. 53-56.

Information Technology Association of American (ITAA), (2002) "New Survey Finds Slight Increase in Workforce Size, but Demand Forecast Softened for IT Workers," ITAA Report.

Mayer, R. (1975), "Different problem solving competencies established in learning computer programming with and without meaningful models," Journal of Educational Psychology, 67, pp. 725-734.

National Center for Educational Statistics (NCES), (2001). "Digest of Educational Statistics 2000." Washington, D.C., U.S. Department of Education, Office of Educational Research and Improvement, NCES 2001-034.

National Science Foundation (NSF), (2000). "Women, Minorities, and Persons with Disabilities in Science and Engineering," National Science Foundation Report, http://www.nsf.gov/sbe/srs/nsf00327/start.htm (accessed June 13, 2002).

Presidential Information Technology Advisory Committee, (1999). Information Technology Research: Investing in our Future. < http://www.itrd.gov/ac/report/ > (accessed June 13, 2002).

Scanlan, D. (1989). "Structured Flowcharts Outperform Pseudocode: An Experimental Comparison," IEEE Software, pp. 28-36.

Schneider, M. and Gersting, J. (2000). An Invitation to Computer Science, Java Version, Brooks/Cole, Pacific Grove, CA.

Shackelford, R. (1998). Introduction to Computing and Algorithms. Addison Wesley.

Thom, M. (2001). Balancing the Equation: Where Are Women and Girls in Science, Engineering, and Technology? The National Council for Research on Women: New York, NY, 2001.

Wardle, C., and Burton, L. (2002). "Programmatic Efforts Encouraging Women to Enter the Information Technology Workforce,"SIGCSE Bulletin, 34, 2, pp. 27-31

Wittrock, M. (1974). "Learning as a generative process," Educational Psychology, 11, pp. 87-95.

Woodfield, R. (2000). Women, work and computing. Cambridge University Press, UK.

Webb, N., and Shavelson, R., (1985). "Computers, education, and educational psychologists," Educational Psychology, 20, 4, pp. 163-165.

## AUTHOR BIOGRAPHIES

**Thad Crews** is an Associate Professor of Information Systems at Western Kentucky University. He teaches software development, systems analysis and design, and intelligent training systems. His research interests are in the areas of instructional technology and human-computer interaction. His publications include a variety of journal and professional conference papers including the *Journal of Artificial Intelligence in Education*, the *Journal of Computing in Higher Education*, and the *Journal of Informatics Education and Research*. Dr. Crews has worked professionally as a software engineer and is a regular consultant in the area of instructional technology. He currently serves as Chairman for National Association of Information Technology Professionals (AITP) Committee on Technology in Education. Dr. Crews first textbook, *Programming Right From the Start*, will be available from Prentice Hall in 2003.

**Jeff Butterfield** is Chair of the Information Systems Program at Western Kentucky University. He teaches courses in database management, systems design, project management, local-area networking and manage-ment information systems. His professional interests include computer-aided training/testing, computer history, the management of technical professionals, and information systems security. He has worked professionally as an electronic engineer, systems analyst, hardware manager, and trainer. He has also worked in Mexico with maquiladora manufacturing operations. Dr. Butterfield's work has been published in a variety of journals and professional conference proceedings including the Journal of Systems Management, Information Systems Management, Team-Performance Management, and the Journal of Industrial Technology. His research focuses on information system strategy, methods of improving the design process, group-process management and technology education. He has served on the editorial board for Journal of Systems Management, Journal of End-User Computing, and Informing Sciences and regularly

reviews for other journals and textbook publishers. Dr. Butterfield has consulted for a number of businesses in the areas of database design and system security.


Information Systems & Computing
Academic Professionals


## 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