Gender and Black Boxes in the Programming Curriculum

PETER McKENNA
Manchester Metropolitan University, Manchester, U.K.

This paper summarizes the results of an investigation into whether women and men have different (concrete and abstract) styles of programming, and whether the standard computing curriculum is therefore biased against women. The theory underpinning the hypothesis is critically reviewed in practical programming contexts. A concrete means of testing attitudinal gender differences to black-boxed programming elements is reported and the results described and analyzed. A survey of 50 students, designed to test the hypothesis that women are more likely to reject the techniques and "way of thinking" of abstraction in programming, casts doubt on the idea that there is any significant difference between female and male attitudes to prepackaged routines. This paper distinguishes between programming and ways of learning to program – between concrete learning strategies and the use of abstraction in programming - and discusses pedagogical practice as well as curriculum content.

Categories and Subject Descriptors: K.3.2 **[Computers in Education]:** Computer and Information Science Education; K.4.2 **[Computers and Society]:** Social Issues

General Terms: Human Factors

Additional Key Words and Phrases: Gender, programming styles, black boxes

1. INTRODUCTION

This paper reports on an investigation into abstraction and black box programming techniques and their possible role within the curriculum in putting women off computing. The wide-ranging theoretical issues underpinning the hypothesis are examined, and practical questions about simple programming situations devised in order to test the hypothesis that females and males have distinctive styles of programming computers, and have negative and positive attitudes, respectively, to the use of abstraction in programming.

The hypothesis of gendered programming styles originated with the MIT psychologist and sociologist Sherry Turkle [1984; 1988; 1996], and was developed in partnership with Seymour Papert [1990; 1992], a key pioneer in the area of learning technology. It is frequently cited uncritically in gender and computing literature [Sutherland and Hoyles 1988; Kvande and Rasmussen 1989; Frenkel 1990; Peltu 1993; Grundy 1996; Stepulevage and Plumeridge 1998]. Reflecting more general psychosocial characterizations, they identify the masculine programming style as "abstract" and distant, the feminine as "concrete" and engaged: men like to work with prepackaged routines ("black boxes"), but women prefer to see inside them ("glass boxes"), take them apart, or even make their own. Where men seek command from a distance, women seek

Author's address: Department of Computing and Mathematics, Manchester Metropolitan University, Manchester, U.K.

closer and deeper communication with their programs and program components. The masculine style is “abstract,” “hard mastery”; the female is “concrete,” “soft mastery”. The fundamental programming concept of “abstraction” (whereby detailed implementation specifics are concealed) is identified as being particularly inimical to women. Hence it is said that the role of abstraction in curricula constitutes “exclusion,” and proposed that abstraction be given a less privileged status as just one of several different approaches [Turkle and Papert 1992], or even eliminated altogether [Peltu 1993].

Turkle and Papert’s evidence was acquired primarily from interviews with novice programming learners, including children as well as undergraduates. Much of what was said by the undergraduates appears to reflect feminist perspectives on ethics, psychoanalysis, and epistemology [Gilligan 1982; Keller 1983]; but the comments and quotations seldom refer to specific or concrete programming scenarios. The intention of the present research is to examine the role of black boxes in specific programming situations in a programming module at a British university, in order to measure female and male attitudes to them.

The gender problem in computing in the UK is, if anything, more severe than in the US. In 2003, 14% of applicants for places for undergraduate computer science degrees in England were women, and 15% of those accepted were women [Universities and Colleges Admissions Service 2003]. For female applicants, these figures are typical of the last decade in the UK, with a very slight decline of 18%. from a 1999 figure. The situation in higher education is reflected in industry: between 1999 and 2003, the proportion of women in the UK IT workforce fell by almost 50%, according to one estimate [Godwin 2004], from 21% to 12.5%.

2. REASONABLE DOUBT

Some misunderstanding and confusion in the hypothesis have become apparent, suggesting that the theory of gendered differences in programming style may not be entirely reliable, including the apparent conflation of programming “abstraction” with the mainstream meaning of the word and the assumption that “communication” with the computer is analogous to human communication. Such issues have been deconstructed elsewhere [McKenna 2000], and will be raised and examined further. However, the theoretical potential for conclusions that may actually be antithetical to Turkle’s original proposition should be clear from the outset. First, the reliance on theories of gendered communication (where the male is closed and the female open) assumes – quite apart from any problems of stereotyping – that the duty of the feminine programmer as a good communicator is to communicate and “negotiate” individually with their *computer* (rather than with computer users or other members of a development team). Second, programming abstraction hides technical details, so that only details significant to humans are visible. Third, it may be argued that, to a beginner, the computer is one big black box, and the desire "to look inside" simply expresses a desire to understand. Identifying this as a fully formed epistemological style that is natural to women could therefore, far from addressing the extrinsic under-representation and material inequality of women in computing, actually compound it. An honest exploration of these issues may offer clues to possibilities for opening up the computing curriculum to a more representative constituency.

3. PROGRAMMING AND LEARNING

Programming represents the skills with which people make computers useful: the activity and body of knowledge that is most identifiable with power and empowerment in computing. Insofar as computers are important, programming is important. And yet, in the words of one university admissions tutor, "it is the programming that puts people off" [Krechowiecka 2002].

Programming is therefore an indicator of both the difficulty and power of the computer. The computer programmer's apparent obsessive attention to detail (and equally obsessive working hours) has led to the identification of the programmer, at least in popular culture, with the psychology and aptitudes of autistic-spectrum males. However, in an increasingly modular software world where the distinction between user and programmer has blurred, the ability to program assumes increasing importance.

Turkle and Papert explicitly assert that the concrete programming style is not a learning stage, nor even a learning style, but a fully-fledged way of knowing how to program, that is, an "intellectual style" [1992], which is at least as good as any epistemology that prioritizes abstraction. This claim clarifies the full significance of the hypothesis, and at the same time raises an important distinction between ends and means–between programming style and learning style. On the one hand we may learn a style of programming; and on the other we may have individual learning styles. Learning styles and programming styles are different, converging only insofar as the learning process involves programming. Indeed, it is possible to learn by using a programming style that is different from the one in which we will actually program. We could use concrete learning methods, a concrete learning style, and even concrete programming practice, in order to learn how to program in an ultimately "abstract" way.

4. DIFFERENCE AND ABSTRACTION

It is tempting to assume that the extrinsic differences in the representation of females and males in computing must have intrinsic correlatives to the way women and men understand and relate to the world in general and computers in particular. However, the notion that the inequality between women and men in computing can be explained in terms of significant natural differences may be only one remove from the idea that all social differences are natural.

The results of an initial study of first-year student programmers in an English liberal arts institution [McKenna 2001] where students were taught in a dedicated concrete learning environment found no significant gender difference in the students' experience of black boxing, which tends to support the suggestion that the gendering of programming styles may be premised on stereotypes and on a misinterpretation of abstraction. The present paper describes and analyzes the results of a second study, conducted at a traditional, predominantly male, computer science department in a British "new university," where students learn to program in a standard commercial programming language.

5. THE COURSE AND SAMPLE POPULATION

We chose a student population because Turkle and Papert used students and young learners (rather than expert practitioners). Additionally, any correlation or confusion between programming styles and styles of learning to program might not be as evident among practitioners; and attitudes at the higher education stage are perhaps less likely to be hidebound and less influenced by conformity to expectations.

The sample group consisted of first-year undergraduate students in an introductory Java module as part of the computing element of their BSc combined honours (CH) degree at Manchester Metropolitan University (MMU). While other computing cohorts (single honors in software engineering, computer science, and information systems) also studied in the same module, the CH group had a larger female minority, and had chosen to study computing along with another subject, rather than computing on its own. The group was chosen partly to reflect the interdisciplinary nature of Turkle's interviewees and partly because it was a coherent cohort with relatively non-trivial numbers of female students. CH students enrolled in one of two computing subjects: computing science, or information systems.

Of the 73 applicants originally offered places in the course, 16 (22%) were females and 57 were males. Eventually there were 41 students enrolled in computing science and 26 in information systems (with 33 and 20 students, respectively, proceeding to examination). The retention rate for males was 84%, as opposed to 94% for females. The average mark of all CH students in the first-year introductory programming unit was 40.7%; the average for females, 40.9%. More female students were enrolled in the computing science area than in the "softer" information systems stream. Consequently, two-thirds of female students completed the more technical computer architecture unit rather than systems analysis. And the average grade for female students in these units was higher – in defiance of conventional notions about which areas of computing are "female-friendly" (women achieved an average mark of 49.6% in computer architecture, and 42.5% in systems analysis).

For some time CH students had experienced programming as a difficult subject area. Up to the academic year in which survey for the present research was conducted, the language used to teach programming was Modula-2. Previous strategies included the extensive use of a computer-based learning package that adopted constructivist approaches to learning and facilitated relatively independent study. Results were disappointing, so to increase motivation it was decided to switch to a more high-profile and web-oriented language.

The tutors teaching the programming module were all male. The few female students in the cohort were deliberately grouped together, so it was not unusual for classes of 32 students to be exclusively male. The MMU course had a particularly young student profile, with only one student over the age of 25, and all the female students were between 18 and 20 years old. The eventual sample size was 50, of whom 39 were male and 11 female.

During the programming unit, students only implicitly encountered abstraction and black-boxing as concepts in testing programs and using stepwise refinement for design and the corresponding methods for implementation. Students had no awareness of gender as an issue in programming or even as a significant part of the research. The possibility that the choices made by the female participants in the questionnaire might be influenced by the overwhelmingly male environment was thereby minimized – although questions do remain concerning the general influence that such a masculine culture could have on all aspects of student response.

An immediate problem with the new programming language led to a "pre-packaged" solution. Implementing simple input – reading from a keyboard – is an inordinately complex task in Java. User input cannot be avoided even in writing one's earliest programs. A computer program is of little use if it cannot obtain data from outside itself; the level of technical detail needed to read an input stream is a serious distraction, an obstacle at the early stages of learning to program.

The book by Bailey and Bailey [2000] adopted for the programming unit simplified input and output by providing, in the words of Turkle and Papert, "prepackaged routines" that made it possible to write interactive programs from an early stage of the learning process. The book's approach, emphasizing the elements of programming rather than the intricacies of Java, was made possible by the use of Java's high-level abstraction feature, called a "package." The term *package* relates strongly to Turkle and Papert's concept of "prepackaged procedures" and "opaque containers." A Java package is simply a named library of routines used for a particular purpose; to quote the book, "a collection of related software components" [Bailey and Bailey 2000]. Lisa, one of Turkle and Papert's students [Turkle and Papert 1992], wrote her own procedures because she "resents" the "opacity" of "prepackaged ones from a program library." Java is fundamentally based on packages (every class (or "procedure") in Java is part of a package); for example the standard Java API (Applications Programming Interface) and the AWT (Abstract Windowing Toolkit), which provide the programmer with basic functionality and graphical user interface components respectively. The first lines of most Java programs consist of a statement or statements importing routines from one or more packages. A novice programmer, writing only small programs, will use only third-party packages; with more experience and time, a programmer could also write and then use his or her own packages to organize a large program, and to systematize and reuse building-block routines.

Abstracted routines are therefore hard to avoid. On the other hand, the book's early and extensive presentation of graphics routines allows for a "concrete" style of learning. In general, students enjoyed calling routines that immediately drew shapes and colors onto the screen. All of these routines were provided as part of Bailey and Bailey's *Elements* package – an example of a concrete learning environment made possible by high levels of abstraction.

Therefore, in the module, data hiding via opaque third-party routines was used in a practical way to simplify input and output, but not to actually teach data hiding. This process exemplifies the distinction between programming abstraction and conceptual "abstraction." Using prepackaged third-party routines represents a "concrete" application of abstraction, whereas programming in an explicitly object-oriented way requires a full conceptual understanding of abstraction. The perception of the teaching team was that students could not move to the level of full conceptual understanding without first experiencing and understanding the concrete representation of the routines. A substantial section of the students also had problems with stepwise refinement as a design technique. Stepwise refinement clearly represents the use of abstraction; hiding details is in many ways inherent to the design process itself. A tutor commented that to understand stepwise refinement sufficiently to make it a useful design technique, requires a level of reflection and conceptualization that beginners are unlikely to reach without first acquiring concrete experience of what it means in action. Such experience is the starting-point in a learning cycle that eventually leads to the conceptual understanding necessary to design solutions to problems rather than just implement possible solutions [Kolb 1984].

6. THE QUESTIONNAIRE

A questionnaire, with practical propositions concerning black boxes, was devised primarily to minimize any subjective influence on the part of the researcher. Interviews in particular can be readily compromised, especially where there is a teacher-student power relation between the researcher and the interviewee. The difficulty in setting particular tasks and observing students going about them was not matched by any obvious benefit; again, the perception and recording of attitudes was done by the researcher.

An attitudinal questionnaire with an ordinal scale could in theory provide the best measure of student attitudes. An odd-numbered Likert scale was chosen in order to avoid forcing respondents into making a decision. Since the intention of the study was to see whether there is in fact any difference in female and male attitudes to abstraction, a polarising scale could be prejudicial. So we adopted a simple three-point Likert scale that ranged from definite rejection (1), through a willingness to consider the notion (2), to definite confirmation (3). While the validity of the measure would in large part depend on the particular formulation of the questions, an even scale may have been interesting nonetheless, given the eventual strong distribution around the middle points in response to some questions.

As the unit made little or no explicit reference to the design concepts of abstraction or black boxes, it was neither possible nor desirable to ask questions that referred explicitly to these concepts. The guiding principle in formulating the questions was to exemplify, in a concrete way, what it means to use prepackaged routines. Preferences in relation to black boxes were elicited by asking implicit and strongly exemplified questions. The Bailey and Bailey book used on the programming unit made specific use of what Turkle and Papert call "prepackaged routines," so practical examples of these were presented to students and their reactions were elicited.

A key measure of the validity of the hypothesis is the extent to which students wanted to look inside a black box. The package provided by Bailey and Bailey is a clear example of a prepackaged routine, where the students do not have immediate access to internal source code. The question was formulated to reflect both Turkle and Papert's wording in reference to black boxes and the appropriate specifics of the students' programming experience (in Turkle and Papert's 1992 paper, an interviewee "frustrated with black-boxing" told her teaching fellow that she "wanted to take it all apart").

a. If time allowed, would you like to inspect and take apart the code that Bailey & Bailey wrote in implementing the *element* package, so that you could find the detail of how methods such as readLine were written?

   This question was designed to test the weaker end of Turkle and Papert's thesis; it entails a desire only to look within a prepackaged routine, an inclination to "bother with its detail". The proviso "if time allowed," was added to all questions because it was necessary to offset simplistic negative responses based on the extra effort implied in the question.

   Another of Turkle's students "prefers to write her own smaller 'building block' procedures even though she could use prepackaged ones from a program library; she resents the latter's opacity" [Turkle and Papert 1992].

   Hence the second question asks if students would prefer to actively write their own routines for reading input, instead of using the one supplied in the *Elements* package:

b. Would you prefer to write and use your own routines rather than using either readLine or another prepackaged method?

   The third question refers to the use of a somewhat lower level of prepackaging; the lower-level packages that the authors of the book themselves used in writing their packages:

c. Would you prefer to use the Java AWT and IO packages directly yourself?

   The packages in Bailey and Bailey are particular to that book, and are designed to ease the learning curve for students. The AWT and IO packages, on the other hand, are what programmers "in the real world" use. For this reason, although the AWT and IO packages represent a lower level and greater difficulty, students may want to use them directly.

The fourth question tests the stronger end of the hypothesis, by asking whether students would not only use the AWT and IO routines, but actually take them apart to see how they work:

d. Would you like to take apart the Java AWT and IO packages to find out how the underlying code has been implemented?

7. ADMINISTRATION OF THE QUESTIONNAIRE

The distribution of the questionnaire was discussed with the three tutors responsible for teaching the CH students and with the CH course leader. Because a small number of students might not understand what was meant by some of the questions, it was agreed that before they started to fill in the questionnaire it would be useful for the tutors to recap briefly on what packages were and how the students had encountered them.

The distribution of questionnaires took place at the beginning of the weekly laboratory session, approximately three-quarters of the way through the academic year. At this time there is no immediate pressure to complete assignments or revise for examinations, but there has been sufficient time for students to study and understand what was being asked in the questionnaire.

8. ANALYSIS OF DATA

The higher scale points indicate one of the following choices: (1) the student wants to see inside a prepackaged routine; (2) write his or her own routines instead of using those prepackaged by the authors of the coursebook; (3) to use routines from standard Java packages for themselves; and (4) to see inside the routines in the standard Java packages. If we accept Turkle's thesis concerning attitudes to black boxes, we would see a significant difference between the responses of female students and those of male students: females would register high scale points more frequently than males and males would register low scale points more frequently than females.

The key questions are whether or not the attitudinal data fit the pattern suggested by Turkle and Papert and whether or not any difference between female and male responses is significant. A X^2 test was therefore used to examine how much the observed response frequencies differed from those we would expect given the null hypothesis that there is no difference between women and men.

Question (a): Observed frequencies of response to the weak "glass box" question, i.e., whether students would like to inspect and take apart the package provided by the coursebook, appears in Table I.

Table I

		'See Elements Code'				
Observed Data	**O_{ij}**					
		scale point	***1***	***2***	***3***	***Total***
		Male	**3**	**27**	**9**	**39**
		Female	**4**	**7**	**0**	**11**
			7	***34***	***9***	**50**

The data shows a strong distribution around the middle scale point. The expected frequencies, under the hypothesis of no difference, are shown in Table II.

Table II

		'See Elements Code'				
Expected data	**E_{ij}**					
		scale point	***1***	***2***	***3***	
		Male	5.460	26.520	7.020	39
		Female	1.540	7.480	1.980	11
			7	34	9	50

It may be noted in Table II that the number of female responses at the highest scale point is zero, whereas the number of male responses is higher than expected at the top point (9 rather than 7) and at the middle point. The inverse is true at the lowest scale point: the number of male responses at this scale point is lower than expected (3 observed and 5.46 expected), and the number of female responses was higher (4 observed and 1.54 expected), indicating slightly less female dissatisfaction with black boxes.

The calculation of the value of χ^2 is shown in Table III.

Table III

		Chi-Squared				
		scale point	***1***	***2***	***3***	
		Male	1.108	0.009	0.558	1.676
		Female	3.930	0.031	1.980	5.940
			1.108	0.009	0.558	**1.676**

With two degrees of freedom from three scale-points and a 5% significance level, the critical value of χ^2 is 5.991. With Calc χ^2 1.676 < Tab χ^2 5.991, the null hypothesis was accepted for the *Elements* glass box question. Hence it was concluded that the difference between the responses of women and men is not statistically significant.

Question (b): The observed frequency of response to the question whether students prefer to write their own routines instead of using the prepackaged methods, is shown in Table IV.

Table IV

	2.	*'Own Routine'*				
Observed Data	**Oij**					
		scale point	***1***	***2***	***3***	***Total***
		Male	**19**	**18**	**2**	**39**
		Female	**4**	**5**	**2**	**11**
			23	***23***	***4***	**50**

The major difference between the responses in Table IV and those to question (a), lies in the much larger number of responses at the lowest (negative) scale point, which is as strong as that around the middle scale point. It is clear that most students do not prefer to write their own routines. The expected frequencies, under the hypothesis of no difference between female and male respondents, are shown in Table V:

Table V

	2.	'Own Routine'				
Expected data	E_{ij}					
		scale point	1	2	3	
		Male	17.940	17.940	3.120	39
		Female	5.060	5.060	0.880	11
			23	23	4	50

With this question the proportion of female responses among the few at the highest scale point is actually higher than expected. The number of male responses at the lowest point is higher than expected, the number of female responses lower. But the numbers (two students from each gender) are especially small.

The calculation of the value of χ^2 is shown in Table VI:

Table VI

		Chi-Squared				
		scale point	1	2	3	
		Male	0.063	0.000	0.402	0.46
		Female	0.222	0.001	1.425	1.65
			0.285	0.001	1.828	2.11

Again, Calc $\chi 2$ 2.113 < Tab $\chi 2$ 5.991, and the null hypothesis was accepted. There is therefore no significant difference between the responses of women and men to the proposition that, in this instance, they write their own routines instead of using the prepackaged routine.

Table VII

	3.	'Use AWT'				
Observed Data	Oij					
		scale point	1	2	3	Total
		Male	7	27	5	39
		Female	3	7	1	11
			10	34	6	50

Question (c), on the direct use of standard Java packages, yields results very similar to those obtained for question (a); that is, a fairly strong bunching around the center point and a larger number of males and a smaller number of females than expected at the top and middle points. The number of females at the bottom end is again higher than expected. But there is no significant difference between the responses of women and men to this question.

Table VIII

	3.	'Use AWT'				
Expected data	E_{ij}					
		scale point	1	2	3	
		Male	7.800	26.520	4.680	39
		Female	2.200	7.480	1.320	11
			10	34	6	50

Table IX

		Chi-Squared				
		scale point	1	2	3	
		Male	0.082	0.009	0.022	0.11
		Female	0.291	0.031	0.078	0.4
			0.373	0.039	0.099	0.51

Table X

	4.	'see AWT'				
Observed Data	Oij					
		scale point	1	2	3	Total
		Male	5	27	7	39
		Female	0	8	3	11
			5	35	10	50

Table XI

	4.	'see AWT'				
Expected data	E_{ij}					
		scale point	1	2	3	
		Male	3.900	27.300	7.800	39
		Female	1.100	7.700	2.200	11
			5	35	10	50

Table XII

		Chi-Squared				
		scale point	1	2	3	
		Male	0.310	0.003	0.082	0.39
		Female	1.100	0.012	0.291	1.4
			1.410	0.015	0.373	1.79

The data for question (d) is similar to that for question (b); that is, a very strong bunching around the center, but with the highest overall figure at the top point. This is puzzling, given that it is probably the most challenging proposition of the four. The number of females at the high point was higher than expected, and the number at the bottom point lower. However, again there is no significant difference between the responses of women and men to this question.

9. CONCLUSION

The survey results show no significant difference between female and male respondents in their attitudes to prepackaged routines. In general there was little desire to look inside or take the "prepackaged programs" apart, and, contrary to Turkle's hypothesis, no greater desire to do so on the part of the men than the women. Given the small number of females in the sample group (a consequence of the underlying problem of under-representation), these results should be viewed as speculative rather than conclusive in quantitative terms. Nonetheless, they contribute to a serious theoretical challenge to Turkle and Papert's influential hypothesis. Further research would benefit from a larger female sample, along with walk-through programming tasks and interviews.

The survey also illustrates the principle that hypotheses concerning supposedly gendered programming styles may and should be tested in a concrete way. While programming has changed in the 15 to 20 years since Turkle's interviews and observations, the principle of abstraction is essentially the same. The examples presented to students were reasonably representative, in terms of the granularity as well as the quantity and scale of the internal detail of practical black boxes.

The concrete learning style used in the unit served as a means rather than an end, and does not imply that a concrete approach should be taken to teach programming. The "concrete" learning approach to graphical activities is characterized by its ability to give the learner the means to produce code that gives immediate visual results. Such an approach, even when it encourages students to explore problems in a hands-on and sometimes experimental way, can exist without reference to any desire to get inside black boxes. For Turkle and Papert, however, the "concrete" programming style is not a learning style, nor a stage of development, but a complete way of knowing how to program. Yet the concrete activities represent the starting-point in a learning cycle, which move on to reflection and conceptualization, and then link back to the experimental [Kolb 1984]. Teaching methods can start from the concrete (concrete situations, concrete illustrations of "abstraction," and the concrete experiences of its effects) in order to develop sound conceptual understanding. Turkle and Papert's concrete/soft style may therefore have to be reconsidered as either a hacker style or a learning style from the early part of the learning cycle.

Industry – in any society or culture – will not shift to "softness" in Turkle's sense, simply because such a style is not capable of meeting the practical requirements of computerized systems. Change based on the premise of a feminine programming style opposed to the use of abstraction, planning, and structure is potentially damaging to the position of women in computing. If anything, any curriculum change should center positively on the very modular building-blocks that Turkle rejects as inimical to women.

As a concept and tool, abstraction has the potential to shift programming decisively from its identification with the male-obsessive spectrum and its intimacy with the machine and machine detail, and move it towards people and the real world. The black box helps to blur the distinction between software construction and use. From the more general learners' perspective, the distinction between learning style and product development is apposite: while different learning styles may or may not wish to explore from the bottom-up and the inside out, from a technological perspective the black box is intrinsic to usability and reusability and to the widening of participation in programming.

REFERENCES

BAILEY, D.A. AND BAILEY, D.W. 2000. *Java Elements: Principles of Programming in Java*. McGraw Hill, London.

FRENKEL, K. 1990. Women and computing. *Communications of the ACM 33,* 11 (Nov. 1990), 34-47.

GILLIGAN, C. 1982. Woman's place in man's life cycle. In *The Second Wave,* L. Nicholson, ed. Routledge, London, 1997.

GODWIN, B. 2004. Number of women in IT industry falls by almost half in four years. *ComputerWeekly.com*. (Oct. 5). http://www.computerweekly.com/Article133867.htm Accessed on 1 Nov. 2004.

GRUNDY, F. 1996. *Women and Computers*. Intellect Books, Exeter, UK.

KELLER, E. F. 1983. *A Feeling for the Organism: The Life and Work of Barbara McClintock*. W. H. Freeman, San Francisco.

KOLB, D. A. 1984. *Experiential Learning: Experience as the Source of Learning and Development.* Prentice Hall, Englewood Cliffs, NJ.

KVANDE, E. AND RASMUSSEN, B. 1989. Men, women and data systems. *European Journal of Engineering Education 14*, 1 (1989).

KRECHOWIECKA, I. 2002. Never mind about maths. *Guardian Education* (26 Feb. 2002).

MCKENNA, P. 2000. Transparent and opaque boxes: Do women and men have different computer programming psychologies and styles? *Computers & Education 35* (2000), 37-49.

MCKENNA, P. 2001. Programmers: Concrete women and abstract men? *Journal of Computer Assisted Learning 17,* 4 (2001).

PELTU, M. 1993. Females in tuition. *Computing 12,* 2 (Dec. 1993).

STEPULEVAGE, L. AND PLUMERIDGE, S. 1998. Women taking positions within computer science. *Gender and Education 10,* 3 (1998), 313-326.

SUTHERLAND, R. AND HOYLES, C. 1988. Gender perspectives on Logo programming in the mathematics curriculum. In *Girls and Computers.* C. Hoyles, ed. Institute of Education, London.

TURKLE, S. 1984. *The Second Self: Computers and the Human Spirit*. Granada, London.

TURKLE, S. 1988. Computational reticence: Why women fear the intimate machine. In *Technology and Women's Voices: Keeping in Touch.* C. Kramarae, ed. Pergamon, New York.

TURKLE, S. AND PAPERT, S. 1990. Epistemological pluralism: Styles and voices within the computer culture. *Signs: Journal of Women in Culture and Society 16*, 1 (1990), 128-157.

TURKLE, S. AND PAPERT, S. 1992. Epistemological pluralism and the revaluation of the concrete. *Journal of Mathematical Behavior 11* (1992), 3-33.

TURKLE, S. 1996. *Life on the Screen*. Phoenix, London.

UNIVERSITY AND COLLEGE ADMISSION SERVICE. 2003. *UCAS Annual Data Sets*. http://www.ucas.ac.uk/figures/ads.html. Accessed on 15 Dec. 2004.

Received November 2004; revised February 2005; accepted March 2005