Teaching Introductory Programming to Information Systems and Computing Majors: Is There a Difference?

Christine Prasad and Xiaosong Li
School of Information Systems and Computing
UNITEC Institute of Technology
Auckland, New Zealand

cprasad@unitec.ac.nz and xli@unitec.ac.nz

Abstract

This paper reports on a pilot study that attempts to determine if there are differences between students majoring in Computing and those majoring in Information Systems (IS) enrolled in the same introductory computer programming course. Data is gathered on age, gender, opinion of computer programming, topics of difficulty, difficulty level in computer programming, and how students structure computer programming concepts. Given the small sample size in the pilot study, the results suggested only slight differences between the two groups of students. It also showed that there were some topics within the course that IS students faced difficulties in. Some suggestions for future research are provided.

Keywords: *Information Systems, Computing Systems, introductory programming.*

1 Introduction

"Information Systems (IS) is about the use of technology and ideas for tactical and strategic advantage in business. It is not about mathematics; most people in the discipline do not write computer programs. Instead they spend time creatively, identifying business opportunities and problems and devising approaches and solutions" (ECU, 2002). IS, as a field of academic study, has the nature of lesser theoretical, and more practical and applied emphasis (Gorgone et. al., 2002).

A quick 'scan' through the undergraduate course information web pages of ten popular tertiary education providers in New Zealand, showed that eight institutions offered IS as major or specialisation at degree level while two did not. A common factor across these institutions was that there was at least one compulsory Computer Programming course that formed part of the IS qualification. The other common factor was that none, except one, of these institutions had a custom designed Computer Programming course for IS students. In other words, IS students were expected to study the same programming course that their Computer Science (CS) or Information Technology (IT) counterparts were studying.

Copyright © 2003, Australian Computer Society, Inc. This paper appeared at the *Sixth Australasian Computing Education Conference (ACE2004)*, Dunedin. Conferences in Research and Practice in Information Technology, Vol.30. R. Lister and A. Young, Eds. Reproduction for academic, not-for profit purposes permitted provided this text is included.

Computer programming is of a mathematical and abstract nature. A course in computer programming often has a stereotyped image and is usually laden with "tedious" practical exercises (Prasad and Fielden, 2002, Lowe, 2002, Bucci, et al. 2001, Marks et al., 2001).

This raises two questions:

1. Do IS students need to adapt their learning strategies to suit the programming course they are doing?
2. Do computer programming instructors at introductory levels need to devise courses and delivery methods to suit a more general audience than the more technical and mathematically oriented audience?

Before embarking on a comprehensive study to investigate the answers to the above two questions, it was important to firstly investigate whether there was a difference between IS and mainstream Computing students studying in the current introductory computer programming course. This would determine if a move comprehensive study was required. The differences to be studied were age, gender, opinion, difficulty level, difficulty areas, and mental representation of concepts.

The main research question was to determine if there were any differences between the two groups in any of the factors above.

The implications of major differences would be that the course was not suitable for both groups, while minor differences would imply either that the course was suitable for both groups or that the IS students had adapted to the course. Also, if major differences were to be found, strategies would need to be devised to cope with them.

The factors chosen for this research are documented in previous research into teaching computer programming. Significant work is being done in this area and some of the factors investigated are previous programming experience, gender, mathematical background (Wilson & Shrock, 2001), background and preparedness (Morrison & Newman, 2001), motivation and interest (Jenkins, 2002), and conceptual structure (Adelson, 1981).

The organisation of conceptual structure has been studied in novice/expert literature related to computer programming, to provide insight into the meanings that people make out of concepts. There is evidence to suggest that the way in which subjects organise concepts reflect their mental representation of the way these concepts are related (Adelson, 1981; Allwood, 1986).

The study was carried out in an institute in New Zealand, where computer programming is a compulsory course for

both the Bachelor of Computing Systems (BCS) and Bachelor of Business (BBus) students doing a double major in IS. Only 18.6% of the total enrolment (approximately 172) for this course were BBus students in the semester that this study was carried out.

2 Method

A questionnaire survey was distributed to students in five streams of an Introductory Programming (IP) class, which is a first-year, undergraduate compulsory course taught using a subset of the C++ programming language. The term "questionnaire survey" is used loosely here, as it was not intended to provide quantitative data for the purpose of carrying out rigorous statistical analysis.

The questionnaire was closed-ended with three parts:

1. The first part sought demographic details from students like age and gender.
2. The second part sought their opinion of the course, the level of difficulty they felt the course to be at, and the topics they faced difficulties with in the course.
3. The third part was a modified cardsorting exercise to try and determine how the student categorised or structured programming concepts.

Card sorting is a sorting technique, described in Cooke (1994) and Rugg & McGeorge (1997). Sorting techniques are used extensively in knowledge acquisition and requirements analysis (Allwood, 1986).

The idea behind card sorting is to ask respondents to sort cards, with names of objects or situations in it, called stimuli, into groups according to a criterion. Card sort data is initially analysed qualitatively to find similarities and differences between responses then can be analysed via cluster analysis (Martin, 1999, Stockburger, 1998) leading to the production of tree diagrams or dendrograms that are very visual representations of the data.

For this study, subjects were asked to sort sixteen programming terms, representing the key concepts they had seen so far in the course based on a single criterion, which was "How I think of Introductory Programming Concepts". In order to enable the effective and timely administration of this exercise to a large number of students, the stimuli were written out on their questionnaire form instead of cards. It is acknowledged that this could have been a possible deterrent for those students who were visually oriented. The same cardsort exercise was also given to three instructors for the purpose of comparing responses against the student's responses.

The relationships exposed by categorisation tasks are taken to reflect relationships in the subjects' internal representations (Allwood, 1986).

The questionnaire was distributed three-quarter way through the semester since it is most likely that students have had sufficient exposure to the course material by this time. 45 out of 172 enrolled students, across five different streams, responded to the questionnaire that is a response rate of 26%. The responses from BCS students were compared against those from BBus students. For the cardsort exercise, instructor data was compared as well.

3 Results

3.1 Demographic Characteristics of the Sample

Figures 1 to 3 show some of the demographic distributions across the sample. Figure 1 shows the distribution of each student group. The "Other" group is made up of students from other programmes such as the Graduate Diploma for Computing and programmes from other disciplinary areas.

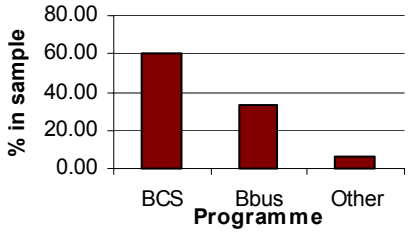

Figure 1: Distribution of students per programme in the sample

Figure 2 shows the age distribution across the sample within each group. The age group "ns" indicates that no age group was stated. No major differences are seen in age groups between the BBus and BCS groups.

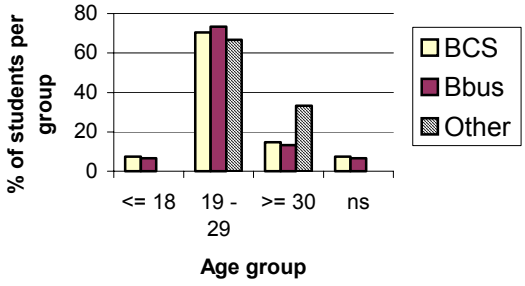

Figure 2: Distribution of age groups within BCS and BBus students.

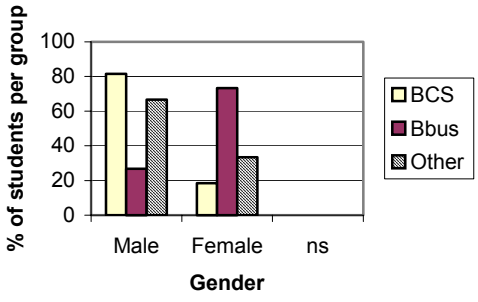

Figure 3: Distribution of gender within BCS and BBus students

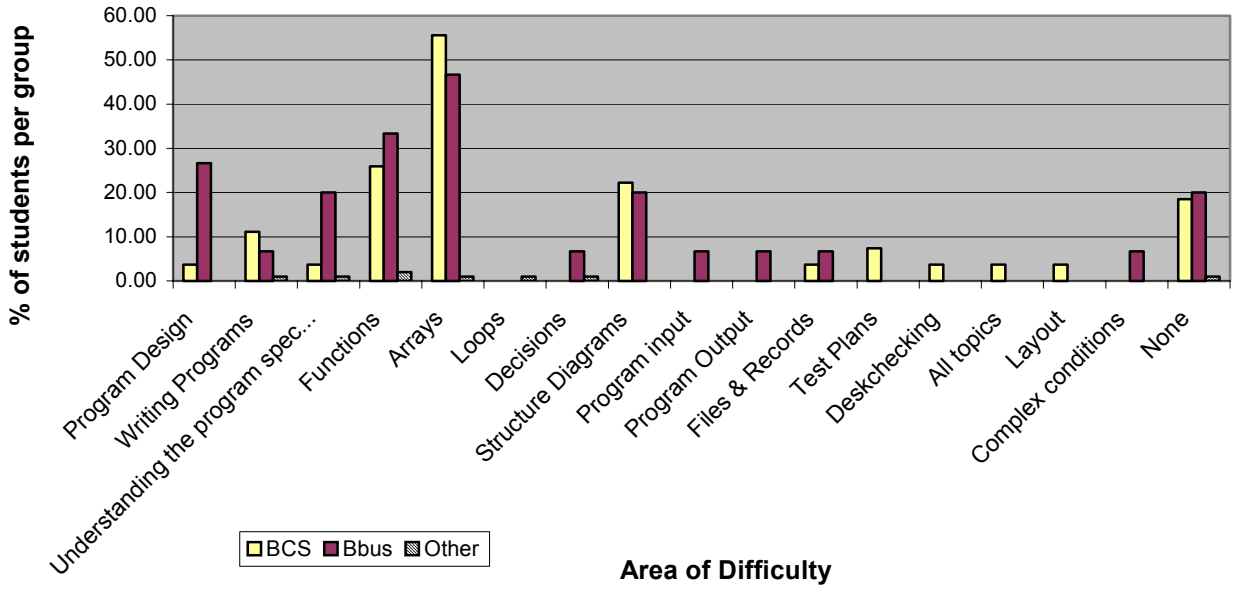

Figure 5: Distribution of areas of difficulty within BCS and BBus students

Figure 3 shows the gender distribution across the sample within each group. The most observable trend was that the number of females in the BBus group was significantly higher than in the BCS group, and the number of males was higher in the BCS group, a trend that has often been observed in Computer Science education (Beyer et. al, 2003; Rowell et. al., 2003)

3.2 Student Opinion

Students were asked to identify their opinion of the course from a selection of seven options. It was possible to select more than one option. The results were once again divided into the different student groups as shown in Figure 4.

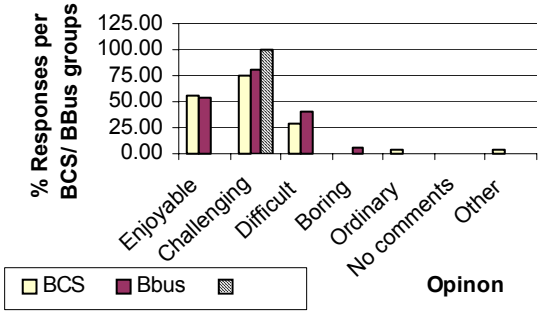

Figure 4: Distribution of opinion on the course with BCS and BBus students

The main observation from Figure 4 is that while most students found the IP course enjoyable and challenging, a slightly higher number of BBus students found it to be difficult.

3.3 Area of Difficulty

Figure 5 shows the responses from students when asked to indicate the topics they faced difficulties within the course. It must be acknowledged that these responses are subjective since they were self-rated. However, they could be tested under controlled circumstances to get a more objective result

Some interesting trends were seen from this graph:

- BBus students appeared to face more difficulty in the more technical areas that involve complicated logic, such as Program design, Decisions, Input, output, and complex conditions. An unexpected area of difficulty indicated by BBus students was "Understanding the Program Specification".
- BCS students indicated difficulty in the less technical areas like Test plans, desk checking and layout.

Common topics of difficulty amongst both the groups were arrays, functions and structure diagrams, which are topics that students usually face difficulties with.

When asked whether they were likely to continue with Programming, a surprising and encouraging result was seen as shown in Table 2.

Continue with Programming?	BCS	BBus
Yes	63%	60%
No	26%	40%
Undecided	11%	0

Table 1: Responses on whether students wish to continue with programming.

When asked why they will not continue with programming, two out of the six BBus students indicated that it was because the course was difficult while two indicated that they couldn't do so under their current elective allocations. From the BCS group, none of the students indicated that they did not wish to continue due to difficulty, but mostly due to other preferences.

3.4 Categorisation of Programming Concepts

Out of the 45 students, 37 carried out the cardsorting exercise. Within the BCS group, the response rate to this exercise was 85%, while within the BBus group, the response rate was 73%. The sorts were analysed by determining the commonality between categories and very simple cluster analysis.

Tables 2 show the average number of categories seen within each group. The fairly similar average number of sorts per student group indicates that, on average, most of the student felt that the stimuli could be categorised into about five groups.

	BCS	BBus	Other	Instructor
Average number of categories	5.13	5.45	5.67	6.33

Table 2: Average number of categories per group

3.4.1 Determining the Commonality Between Categories

The individual responses from each student per group were evaluated and 26 common categories were derived as tabled in Table 3. There were some category labels within the responses for which the authors' discretion was used when deciding which of the above 26 categories to place them in. A summary of the distribution of responses is shown in Figure 6.

From this data, it was easier to observe the common and different categories chosen by the students in each group. The categories to note were:

1. The number of "ragbag" categories, such as "don't know", "other", "the rest", i.e. categories C8, C14, and C16. This do not include the unnamed categories and ungrouped stimuli. According to Rugg & McGeorge (1997), "ragbag" categories indicate the level of uncertainty. In all of these three categories, BCS students appeared to have higher responses than BBus.
2. The "ungrouped" and "unnamed" categories, C22 and C23 respectively. These categories can indicate that students were either not sure of what group to put a term into, or they were not aware of the term. It could also indicate that students were not sure of the instructions given to them. Once again, a slightly higher response was seen within the BCS group as opposed to the BBus group. Also, the "Other" group of students had a markedly higher number of unnamed categories.
3. Subjective responses like "difficult", and "hard" (C6) had a higher response from BBus students while "easy" and "enjoyable" (C9) had a higher response from BCS students.

Category	Category Name
C1	C++ / Code/ Command/ Compile / Statement/ Computer Language
C2	Conditions
C3	Data/ Data type/ Data definition/ Variable type
C4	Decision
C5	Design
C6	Difficult / Hard/ Don't like
C7	Documentation
C8	Don't know
C9	Easy / Enjoyable
C10	Function
C11	Input / Output
C12	Logic
C13	Maintenance
C14	Not applicable
C15	Operators
C16	Other
C17	Program specification
C18	Program structure/ parts of a program/ parts of a function
C19	Programming Cycle / Program development/ Stages
C20	Terms used
C21	Testing/ Data verification/ Deskcheck/ Debugging
C22	Ungrouped
C23	Unnamed
C24	Variable/ Variable Declaration
C25	Process/ Processing
C26	Strategy

Table 3: Common categories from student responses

3.4.2 Cluster Analysis

A dendrogram or tree diagram was created for each group using the EZSort (2003) tool. Analysing the dendrograms included looking at similar and different categorisations. While there is no one correct categorisation, these provide an insight to the terms or concepts that student's cluster into a group.

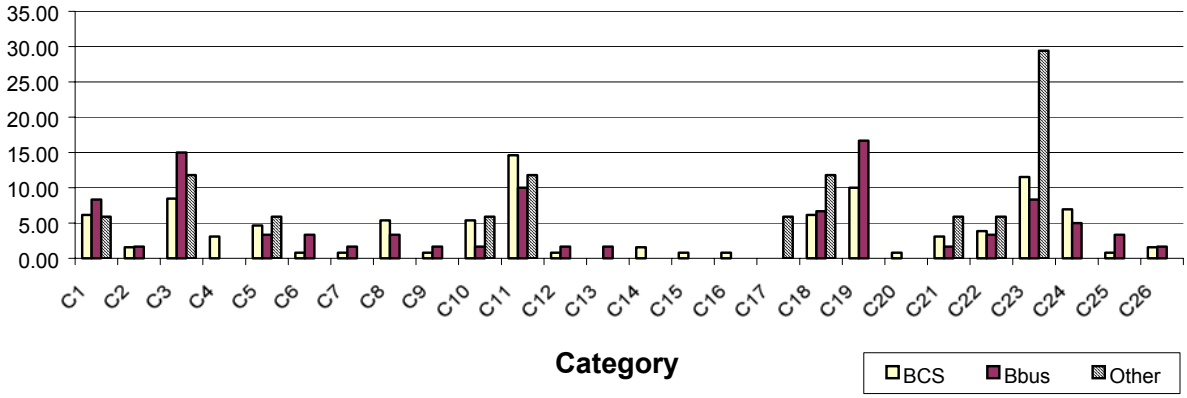

Figure 6: Summary of distribution of responses of BCS/ BBus and Other students

The BCS group had 7 categories while the BBus and Other group had 8 categories. A dendrogram was also created from the responses from the three instructors and seven categories were seen.

Collectively, the categorisations for the three groups are shown in table 4. It must be pointed out that these groupings represent an average of the data.

While the instructor's responses are not the only ones nor are they the correct ones, they do provide a benchmark to measure the student's responses against. The extra groups created by BBus and Other students have been shown in the table as split cells, which both appear to be logical categories. The only "misplaced" term in all three of the student groups appeared to be "array" which, for some reason, has been grouped with a "function". In actual fact, an "array" is a data storage mechanism while a "function" is a data processing mechanism. Apart from that, no major differences are seen from the dendrograms.

4 Discussion

Analysing the results from section 3 will provide directions for the future research. Some minor differences were observed between the student groups which we will try and discuss below.

An unexpected area of difficulty indicated by BBus students was "Understanding the Program Specification". About 20% of the BBus students found this a difficult topic as opposed to the less than 5% of BCS students in our sample. This is unexpected because that IS students are supposed to identify business opportunities and problems (ECS, 2002).

A program specification usually consists of business problem statements and programming technique requirements. We looked at some other BBus course assignments to identify why this may be the case and found that most assignments had similar problem statements. What was missing from their assignments was the programming techniques section. Therefore it appears that BBus students face difficulties understanding the programming (i.e. technical) section of the specification as opposed to the problem statements.

	BCS	**BBus**	**Other**	**Instructor**
1.	Test plan Structure diagram Design Debug	Test plan Structure diagram Design Debug	Test plan Debug	Test plan Structure Diagram
2.	Parameter	Parameter	Design Structure Diagram	Design Debug
3.	Local variable	Local Variable	Local variable	Function Parameter Local Variable
4.	Loop If statement	Loop If statement	Function Array Loop If statement	Loop If statement
5.	Boolean Char Int	Boolean Char Int	Parameter	Array Char Int Boolean
6.	Function Array	Function Array		Cout
7.	Cout Readchar Readint	Cout Readchar Readint	Cout Readchar Readint	Readchar * Readint

Table 4: Common categories amongst the four groups as observed from dendrograms.

* Readchar and Readint are two custom made function used for the input of character and integer respectively at this institution; these function do not form part of the standard C++ library.

One strategy to overcome this would be to communicate the specifications in a slightly different way to BBus students. According to Darling (2001), if you understand the patterns and practice being aware of the patterns in your student's language, you can adapt the way you communicate and make it easier for your students to learn from you.

The pattern we used for program specification in our assignments is basically a sequence, in which, the input in one case is specified first, and then the corresponding outputs are specified (some times supplied with a screen layout), and then the inputs for another case and so on. Obviously this is a pattern BCS students are used to.

The BBus assignments, on the other hand, consist of several components and sub-components, similar to a hierarchy. We also found that matrices were used extensively in many BBus courses, for example, Candidate Matrix, Entity Definition Matrix, Feasibility Matrix, and Problem Statement Matrix. However, a matrix is rarely appears as part of a program specification in our course. While it is not feasible to specify a program specification in terms of a matrix, it might be possible to further simplify the technical aspects of the specification somewhat for the purposes of the BBus students. Further research should be carried out to determine if the specification patterns could really make difference.

BBus students appeared to face more difficulty in the more technical areas that involve complicated logic, such as "Program Design", "Decisions", "Input", "Output", and "Complex conditions" than the BCS students. These are the main contents of a programming course and they reflect the nature of a programming course indicated in section 1.

This trend is possibly due to the higher technical inclination of the BCS students. It also implies that the course may be taught in a technical manner and is not suitable for a general audience. However, further research needs to be carried out in this area to really determine the reasons and specify strategies to overcome these.

Another interesting area of difficulty indicated by BBus students was "Program Design". More than 20% BBus students stated that this topic was difficult as opposed to less than 5% BCS students in our sample. This is perhaps a serious issue for BBus students, as many of them will be required to carry out a program design. This difficulty may be a result of them facing difficulties in understanding the Program specification. This might be due to that BBus students are lack of logic training. Once again, further research is needed to ascertain the extent of this difficulty.

As stated above, the card sort exercise was carried out to elicit the manner in which students internally structure their knowledge. While minor variations were found in the patterns exposed by the two groups, the results generally suggest that BBus students are reasonably comfortable with the programming course compared to BCS students, and that they do not reject programming concepts. The results also suggest that they can manage the concepts at a similar, if not higher level to BCS students. This, however, is not consistent with the areas of difficulty the BBus students face so further research will be required to investigate this.

5 Summary and Future Work

The main observations from the data gathered are summarised:

Age – No major differences were found between the BCS and BBus groups

Gender – A significantly higher number of females were discovered in the BBus group in comparison to the BCS group, and this could strongly affect the results of this study.

Opinion – Only a slightly higher number of BBus students found the course difficult than BCS students. In all the other opinions, the responses were fairly similar.

Areas of difficulty – It appears that BBus students were facing more difficulty in the more technical topics within programming. They also were facing unexpected difficulties in the areas of understanding program specification and program design. This, however, needs further research.

Categorisation of Programming concepts – Categorisation from BCS students matched the instructor's one more closely than the categorisation from BBus students. However, there were a couple of concepts in which both groups of students had a drastically different categorisation from the instructors. A qualitative investigation of the categorisations also showed that BCS students faced more uncertainty with programming concepts than BBus students.

Given that this was a pilot study, it must be acknowledged that the sample size was relatively small and most of the results are based on very small numbers, therefore the results of the study are far from conclusive. However, a number of areas of further research have been identified:

- The same exercise should be carried out with a larger sample size.
- Some of the data collected was highly subjective. This could be rectified by modifying the current study to include alternative or supplementary data collection methods. For example, a test could be used to determine if the student is really facing difficulties in an area. Also, an interview could be used to "drill down" into the areas in which students were facing difficulties to identify reasons for it.
- A visual card sorting exercise could be used instead of a "paper" one. This might enable students to visualise the concepts and form different categorisations.
- An interview could be used to "drill down" on the card sort data to provide more qualitative feedback from this exercise.
- Students could be asked to specify their own criteria for sorting instead of using the provided criterion. This might make data analysis difficult but would provide a wider insight into how students structure their knowledge.
- Attempting to use different teaching methods for the different groups of students and then evaluating their respective effectiveness.
- Investigating the final assessment results of BCS vs. BBus students

- Studying the persistency and retention levels in the Computer Programming course for BCS vs. BBus students
- Attempting to use different specification patterns for the different groups of students and then evaluating their respective effectiveness.

While this study suggests that BBus students are coping quite well with the programming course, the fairly low sample size makes it premature to draw any conclusions. The main course of action would be to carry out the same study with a larger population.

6 References

Adelson, B. (1981) Problem solving and the development of abstract categories in programming languages. Mem. Cognition, 9, 422-433.

Allwood, C. M. (1986) Novices on the computer: a review of the literature. *International Journal of Man-Machine Studies*, 1986, 25, 633 - 658.

Beyer, S., Rynes, K., Perrault, J., Hay, K., & Haller, S. (2003) Gender differences in Computer Science Students. *The Proceedings of the Thirty-fourth SIGCSE Technical Symposium on Computer Science Education*, Reno, Nevada, US, 49 - 53.

Bucci, P., Long, T. J., & Weide, B. W. (2001) Do we really teach Abstraction? *The Proceedings of the Thirty-second SIGCSE Technical Symposium on Computer Science Education*, Charlotte, North Caroline, US, 26 – 30.

Cooke, Nancy J. (1994) Varieties of knowledge elicitation techniques. *International Journal of Human-Computer Studies*, 1994, 41 (6), 801 – 848.

Darling, L. (2001) NLP … Not Another Computer Acronym! http://elementk.com/downloads/nlp.PDF, Accessed 30 August 2003.

ECU (2002) Bachelor of Business: Information Systems. http://www-business.ecu.edu.au/courses/undergrad/Bus/information_systems.htm, Accessed 12 November 2003

EZSort (2003) http://www-3.ibm.com/ibm/easy/eou_ext.nsf/Publish/410, Accessed 7 November 2003.

Gorgone, J. T., Davis, G. B., Valacich, J. S., Topi, H., Feinstein, D. L., & Longenecker, Jr, H. E. (2002) IS 2002: Model Curriculum and Guidelines for Undergraduate Degree Programs in Information Systems. Association for Information Systems. http://www.is2000.org/ , Accessed 21 May 2003.

Jenkins, T. (2001) Teaching Programming - A Journey from Teacher to Motivator. http://www.ics.ltsn.ac.uk/pub/conf2001/papers/jenkins.htm, Accessed 3 May 2003.

Jenkins, T. (2002) On the Difficulty of Learning to Program. http://www.ics.ltsn.ac.uk/pub/conf2002/jenkins.html. Accessed 3 September 2003.

Lowe, S. (2002) Bells & Whistles: Learning Programming Principles Through Multimedia Authoring. *Proceedings of the 15th Annual Conference of the National Advisory Committee on Computing Qualifications*, Hamilton, New Zealand, 271 – 275.

Marks, J., Freeman, W., & Leitner, H (2001) Teaching Applied Computing without Programming: A Case-Based Introductory Course for General Education. *The Proceedings of the Thirty-second SIGCSE Technical Symposium on Computer Science Education*, Charlotte, North Caroline, US, 80 – 84.

Martin, S. (1999) Cluster Analysis for Web Site Organisation . Internetworking, ITG Publication, Dec 1999: 2.3. http://www.internettg.org/newsletter/dec99/cluster_analysis.html, Accessed 20 May 2003.

McCracken, M., Almstrum, V., Diaz, D., Guzdial, M., Hagan, D., Kolikant, Y. B., Laxer, C., Thomas, L., Utting, I. & Wilusz, T. (2001) A multi-national, multi-institutional study of assessment of programming skills of first-year CS students. *SIGCSE Bulletin – Inroads*, 33, no 4, December, 1 - 16.

Morrison, M. & Newman, T. S. (2001) A Study of the Impact of Student Background and Preparedness on Outcomes in CS1. *The Proceedings of the Thirty-second SIGCSE Technical Symposium on Computer Science Education*, Charlotte, North Caroline, US, 179 - 183.

Prasad, C. & Fielden, K.(2002) Introducing Programming: A Balanced Approach. *Proceedings of the 15th Annual Conference of the National Advisory Committee on Computing Qualifications*, Hamilton, New Zealand, 101-107.

Rowell, G. H., Perhac, D. G., Hankins, J. A., Parker, B. C., Pettey, C. C, & Iriarte-Gross, J. M. (2003) Computer-Related Gender Differences. *The Proceedings of the Thirty-fourth SIGCSE Technical Symposium on Computer Science Education*, Reno, Nevada, US, 54 - 58.

Rugg, G. & McGeorge, P. The sorting techniques: a tutorial paper on card sort, picture sorts and item sorts. *Expert Systems*, May 1997, Vol 14, No. 2, 80 – 92.

Stockburger, D. W. (1998). Multivariate statistics: concepts, models, and applications. http://www.psychstat.smsu.edu/multibook/mlt04.htm, Accessed May 20 2003.

Wilson, B. C. & Shrock, S. (2001) Contributing to Success in an Introductory Computer Science Course: A Study of Twelve Factors. *The Proceedings of the Thirty-second SIGCSE Technical Symposium on Computer Science Education*, Charlotte, North Carolina, US, 183 - 188.