Computer Science

100 & 200 Level Curriculum Evaluation

One Year On

i.e. "Is the curriculum working?"

Stephen J. Marshall
Centre for the Study Of Higher Education

Liz Sonenberg
Department of Computer Science

The University of Melbourne
Parkville, VIC, 3052

November 1996

Computer Science
100 & 200 Level Curriculum Evaluation
One Year On

Preamble

As part of the process of evaluating changes to the structure and content of the 100 and 200 level program in the Department of Computer Science at the University of Melbourne, data were collected from students who had completed their first and second year Computer Science studies via these new arrangements over a two year period. The purpose of collecting these data was to determine ***students'*** *level of satisfaction with, and perceptions of, the appropriateness of the new program.*

Method

Data Collection. Data were collected via a written questionnaire (See Appendix A) sent to all students enrolled in at least one third year Computer Science subject, and who had completed the first and second year of their Computer Science course via the new 100 and 200 level program. Specifically, all students in the target group had completed the following Computer Science subjects: 433242-Algorithms and Problem Solving; 433243-Programming Languages Principles A; and 433244-Software Development. A response rate of 50% was achieved in year 1 and 36% in year 2. In the first year of data collection, following a preliminary analysis of the findings of this survey, a small focus-group discussion was conducted involving six volunteers from among the targeted, third-year, student cohort. The discussion was led by a member of the Centre for the Study of Higher Education (CSHE) and an Administrative Officer of the Department of Computer Science (the convenors). Its purpose was to explore, in depth, some of the issues that had arisen from the questionnaire data. Four broad questions guided this phase of data collection:

* In what ways has the Computer Science department's first and second year teaching program prepared you to become a good programmer?
* In Computer Science education, what do you see as the advantages/disadvantages of introducing one "industrial strength" language and studying and practising it in depth over several Computer Science subjects, as

compared with providing a less detailed but broader exposure to a variety of languages?

* In what ways do/should the teaching modes of lectures, tutorials, laboratories, assignments etc., contribute to your overall learning of computer science?
* In what ways could/should the first and second year Computer Science program be changed to improve its quality, appropriateness, and effectiveness?

Summaries of the participants' responses were made by each of the convenors, and for the first two questions, where a technical knowledge of the content of Computer Science was required, by a member of the lecturing staff of the department.

<u>Data Analysis</u>. Three different types of data analysis were used:

1. student responses to questions requiring answers on Likert-type scales were recorded on an Excel spreadsheet and graphs showing the distribution of students' answers across the range of possible responses were produced for each question.

2. content-analysis, of the type described by Glaser and Strauss (1967) was used to analyse data collected via short answer questions in the questionnaire (Effectively, this involved reading students' comments a number of times to identify the full range of different responses provided, and then counting the number of students who made each response).

3. content-analysis of the type described by Lincoln and Guba (1995), Merriam (1988), and Delamont (1992) was used to analyse the data collected during the focus-group discussion. The summaries of student comments, developed by the convenors of the focus group discussion, were read and re-read to identify any obvious themes in relation to each of the principal questions that guided the research. The data were then unitised and categorised to determine if there were any other themes contained within the data that were not apparent in these first readings. These themes were collated under headings associated with each of the principal research questions.

Necessarily, each of these last two processes of data analysis are influenced by the analysts' own understandings, beliefs, and interests. Consequently, in an effort to ensure that the findings of these stages of the research truly represented the views of the participants (i.e., that the convenors' paraphrasing of the participants' responses was indeed accurate), those students involved in the focus-group discussion were given the opportunity to reflect upon the interpretations made, and where necessary, to amend these accounts prior to the preparation of this report.

Findings

To reflect the three different phases of data analysis described above, the findings of this research are presented here in three parts. Part A summarises students' responses to those questions which required them to *assess the level of their current knowledge and skill* in a number of different areas covered in the first and second year Computer Science course. Part B summarises students' perceptions of the overall *quality and effectiveness* of the first and second year Computer Science programs. In presenting these findings, an effort has been made to separate and make clear the responses of students in each of the two different rounds of data collection. Part C provides a *summary of the issues* raised by students during the focus-group discussion held at the end of the first round of data collection.

Part A

It is evident from Table 1 that respondents to both rounds of the CS questionnaire generally believe that their problem analysis, program design, program creation, and program testing knowledge and skills are either adequate or excellent for their current stage of development.

Table 1

Students' Assessment of their knowledge of, and skills in:					
	Excellent	Adequate	Poor	Non-existent	No Response
Problem Analysis	38 *(40)**	59 *(50)*	3 *(8)*	0 *(0)*	0 *(2)*
Program Design	14 *(16)*	76 *(70)*	10 *(14)*	0 *(0)*	0 *(0)*
Creation of Efficient Programs	31 *(18)*	66 *(72)*	3 *(10)*	0 *(0)*	0 *(0)*
Program Testing	21 *(22)*	72 *(62)*	7 *(16)*	0 *(0)*	0 *(0)*

Note: All entries are percentages based on the ratio of the number of students who chose a particular descriptor to the total number of respondents.
* All entries in italics are based on data collected during the second round of data collection.

However, it is interesting to note that respondents' assessments of their level of knowledge and skills in these areas have declined between the first and second rounds of the survey. In all categories except "program design" more than twice as many respondents rated their level of knowledge and skill as "Poor" in the second round of the survey than did in the first round.

Table 2 indicates that survey respondents in both rounds of data collection believe that they have very different levels of understanding in relation to the four different programming paradigms presented during their 100 and 200 level Computer Science program. Clearly, nearly all respondents believe that their knowledge of imperative programming is adequate for their current needs. However, a large percentage of respondents in each round of data collection (between 35 and 55% in the first round and between 32 and 58% in the second round) believe that their knowledge of functional programming, object-oriented, and logic programming is inadequate. As in the results summarised in Table 1, the data included in this table indicated that there has been a general decline in respondents' overall assessment of their knowledge of these various programming paradigms between the first and second rounds of data collection.

Table 2

Students' Assessment of their Knowledge of Various Programming Paradigms:					
	Excellent	Adequate	Poor	Non-existent	No Response
Imperative Programming	69 *(48)**	31 *(48)*	0 *(2)*	0 *(2)*	0 *(0)*
Functional Programming	17 *(14)*	45 *(54)*	28 *(30)*	7 *(2)*	3 *(0)*
Object-oriented Programming	17 *(6)*	41 *(46)*	41 *(46)*	0 *(2)*	0 *(0)*
Logic Programming	3 *(8)*	41 *(34)*	48 *(46)*	7 *(12)*	0 *(0)*

Note: All entries are percentages based on the ratio of the number of students who chose a particular descriptor to the total number of respondents.
* All entries in italics are based on data collected during the second round of data collection.

The data summarised in Table 3 is clear in relation to respondents' assessments of their skills in the use of various programming languages. In both rounds of data collection nearly one hundred percent of respondents rated their current level of skill in C to be either adequate or excellent. However, in relation to Pascal, C++, Miranda, 68000 Assembler, and Prolog more than 30% of respondents in the first round and more than 46% in the second round rated their current level of skill as either poor or non-existent. Of most concern is the significant increase in the numbers of respondents who assessed their skills in Miranda, 68000 assembler, and Prolog as non-existent in the second round (around twice as many as did in the first round).

Table 3

Students' Assessment of their Skills in:					
	Excellent	Adequate	Poor	Non-existent	No Response
Pascal	14 *(22)*	48 *(30)*	10 *(24)*	21 *(22)*	7 *(2)*
C	72 *(64)*	28 *(34)*	0 *(2)*	0 *(0)*	0 *(0)*
C++	21 *(6)*	48 *(44)*	31 *(44)*	0 *(6)*	0 *(0)*
Miranda	21 *(8)*	38 *(36)*	28 *(34)*	10 *(20)*	3 *(2)*
68000 Assembler	10 *(0)*	17 *(24)*	52 *(32)*	17 *(40)*	3 *(4)*
Prolog	7 *(6)*	38 *(22)*	34 *(44)*	14 *(26)*	7 *(2)*

Note: All entries are percentages based on the ratio of the number of students who chose a particular descriptor to the total number of respondents.
* All entries in italics are based on data collected during the second round of data collection.

From the point of view of Unix Tools, respondents to both surveys generally believed that their current level of knowledge in relation to program development and configuration management is adequate for their current work (See Table 4). However, it is interesting to note that while a slightly higher percentage of respondents assessed their knowledge of UNIX tools for configuration management as "excellent", exactly twice as many respondents in the second survey assessed their knowledge in this area as "poor". Little doubt seems to exist in the minds of both cohorts of respondents in relation to the adequacy of their knowledge of UNIX debuggers with 42% of respondents in the first cohort and 64% in the second believing that on entering their 300 level studies, their knowledge of this area is either poor or non-existent.

Table 4

Students' Assessment of their Knowledge of UNIX tools for:					
	Excellent	Adequate	Poor	Non-existent	No Response
Program Development	10 *(26)**	86 *(70)*	3 *(4)*	0 *(0)*	0 *(0)*
Configuration Management	28 *(30)*	62 *(48)*	10 *(20)*	0 *(2)*	0 *(0)*
Debuggers	10 *(6)*	48 *(30)*	38 *(48)*	4 *(16)*	0 *(0)*

Note: All entries are percentages based on the ratio of the number of students who chose a particular descriptor to the total number of respondents.
* All entries in italics are based on data collected during the second round of data collection.

Not, surprisingly, the results summarised in Table 5 indicate that respondents to both surveys generally expect their level of knowledge and understanding of each of the key

Table 5

Students' Expectations in Relation to the Level of their Knowledge and Skills at the end of the Third Year of their Studies in:					
	Excellent	Adequate	Poor	Non-existent	No Response
Creating Efficient Programs	59 *(46)**	41 *(50)*	0 *(4)*	0 *(0)*	0 *(0)*
Program Testing	62 *(54)*	38 *(42)*	0 *(4)*	0 *(0)*	0 *(0)*
Program Design	69 *(54)*	28 *(42)*	3 *(4)*	0 *(0)*	0 *(0)*
Writing User Manuals	38 *(26)*	38 *(64)*	14 *(6)*	10 *(4)*	0 *(0)*
Working as a Team Member	48 *(44)*	41 *(50)*	3 *(6)*	7 *(0)*	0 *(0)*

Note: All entries are percentages based on the ratio of the number of students who chose a particular descriptor to the total number of respondents.
* All entries in italics are based on data collected during the second round of data collection.

aspects of software design and development (program creation, program testing, program design, writing user manuals, and working in teams) to be either "adequate" or "excellent" at the end of their third year of Computer Science studies. Such optimism is either a strong reflection of respondents belief in the quality of their experiences of third year Computer Science subjects to date, or simply an indication of their general level of optimism. Probably the most interesting finding in relation to the data summarised in Table 5 is that concerning "writing manuals". While respondents to both surveys are clearly not as confident that their level of knowledge of, and skill in, writing software manuals will be as good as their knowledge and skills in other areas, respondents to the second survey were generally more optimistic with less than half the percentage of students indicating that their expected level of knowledge and skill in this area to be either "poor" or "non-existent" at the end of their third year of studies.

Table 6

Students' Responses to the question: ***For your current needs, do you know enough about***			
	Yes	No	No Response
Using the UNIX Operating System?	79 *(82)*	14 *(16)*	7 *(2)*
Electronic Mail?	93 *(96)*	7 *(2)*	0 *(2)*
Electronic News Groups?	83 *(86)*	14 *(12)*	3 *(2)*
ftp?	59 *(88)*	34 *(10)*	7 *(2)*
X Windows?	69 *(70)*	28 *(28)*	3 *(2)*
Web Browsing	55 *(92)*	38 *(6)*	7 *(2)*

Note: All entries are percentages based on the ratio of the number of students who chose a particular descriptor to the total number of respondents.
* All entries in italics are based on data collected during the second round of data collection.

Table 6 would appear to suggest that respondents to both the CS questionnaires are generally happy with the level of their current knowledge of the UNIX operating system, electronic mail, and electronic news groups, ftp, X windows, and web-browsing. However, it is interesting to note that in both surveys over a quarter of respondents expressed the belief that they did not know enough about X windows to meet their current needs. These data suggest that careful consideration needs to be given to ways in which students might be provided with greater opportunities to develop these skills during their 100 and 200 level programs.

In considering this issue, however, due acknowledgment needs to be given to the findings of Table 7 which indicates overwhelmingly that respondents to both surveys believe that their knowledge and skills in each of these areas was primarily developed through interaction with friends, in laboratories, and via a range of activities *other than* those formal activities organised by the department (e.g., lectures and tutorials).

Table 7

Students' Responses to the question:
Where did you learn about

	Lectures	Tutorials	Labs	Friends	Other
Using the UNIX Operating System?	9 *(14)**	7 *(12)*	25 *(54)*	39 *(58)*	21 *(32)*
Electronic Mail?	0 *(2)*	3 *(2)*	14 *(26)*	66 *(78)*	17 *(24)*
Electronic News Groups?	3 *(2)*	0 *(0)*	0 *(8)*	73 *(70)*	23 *(20)*
ftp?	0 *(0)*	0 *(0)*	4 *(6)*	75 *(68)*	21 *(32)*
X Windows?	3 *(12)*	3 *(4)*	24 *(24)*	48 *(40)*	21 *(32)*
Web Browsing	0 *(4)*	0 *(0)*	0 *(8)*	70 *(68)*	30 *(36)*

Note: All entries are percentages based on the ratio of the number of students who chose a particular descriptor to the total number of respondents. Percentages can exceed 100% as students could chose more than one response.
* All entries in italics are based on data collected during the second round of data collection.

The extent to which respondents believed that departmental activities played little role in helping them to develop their knowledge and skills in these areas prompted one respondent to the first survey to write in answer to the question "Where did you learn [these skills]?"--"*definitely not from anything the department has taught*." Generally, respondents to both surveys believed that their knowledge and skills in these areas were developed by reading manuals and textbooks, experimenting with the software/tools etcetera, and discussing their understanding and experiences with their fellow students.

Part B

Before proceeding to outline the findings embedded within the data collected via open-ended questions and in the comments of participants in the focus-group discussion, the following points needs to be made. Analysis of these data requires interpretation. Before the data can be categorised, analysts must ascribe meaning to the data based upon their understanding of the contexts within which the data are presented. Thus, because of the different backgrounds, experience, and beliefs associated with different analysts, it is inevitable that these interpretations will vary from analyst to analyst.

Efforts must, therefore, be made to establish the validity of these interpretations. In the case of the data collected via open ended responses in the first CS questionnaire, the validity of the analyst's interpretations was checked by sharing these interpretations with members of the focus-group discussion and modifying these interpretations on the basis of the comments made. In the case of the data collected via open ended responses in the second questionnaire, the validity of the analyst's interpretations was checked by comparing interpretations made of the data collected by the first questionnaire. In the case of the analyst's interpretations of comments made by participants in the focus-group discussion, draft copies of the analyst's report of the discussion were sent to participants (both students and staff members) inviting them to comment on its accuracy and to make any alterations necessary to make it a better representation of the views expressed during the discussion.

It is inevitable, therefore, that the numbers/percentages quoted in the following discussion would vary if the analysis were undertaken by different individuals. However, they can be taken to be true representations of respondents/participants beliefs, to the extent that they have been calculated from data, validated by the respondents/participants themselves.

Eighty-six percent of respondents (84% on the second survey) believe that their 100 and 200 level subjects have adequately prepared them for their 300 level studies in Computer Science. For the remainder of respondents who believed that this was not the case, three principal reasons seemed to be given. First, it was suggested that there needed to be more emphasis in the 100 and 200 level program on "designing good programming algorithms before coding." Second, some students suggested that some lecturers "assumed too much." The implication appeared to be, that these student's preparation for third year could have been better had greater attention been given to the fact that they did not have as strong a background in some areas prior to their 100 level studies as their lecturers had assumed. Their preparation could have been improved had these deficiencies been recognised and catered for. Third, there was a belief among some students that their preparation for third year could have been improved had their 100 and 200 level subjects alerted them to a greater extent to the "tools" that the department had available.

In regard to the extent to which Computer Science subjects contribute to the development of students' oral and written communication skills, approximately 60% of respondents (50% in the second survey) suggested that "oral communication skills are "hardly used at all in most Computer Science subjects". Of the 34% of respondents who believed otherwise (46% in the second survey), more that 90% indicated that these skills were "really only used in third year subjects". Coupled with comments like "no oral presentations were required at the 100/200 level" and "It would have been nice to have been required to do some [presentations] prior to third year", the data seem to suggest that it is the respondents' belief that existing program arrangements do not place sufficient emphasis on the development of these skills at the 100 and 200 level. This interpretation is made cautiously, however, in the light of one student's comment from the first survey, that: "[Computer Science subjects] *probably don't* [place sufficient emphasis on the development of oral and written communication skills] *but WHO CARES? I would have done ARTS if I wanted to learn how to write essays. Instead I chose CS to learn about computers!*" (emphasis in the original).

Amongst those topics which respondents to both surveys considered to be "critically important and particularly useful" in their first two years of Computer Science, *programming in C* , stood out by far as the most important (See Table 8). This, combined with the fact that the majority of respondents on both surveys indicated that C++ was not presented in sufficient detail, seems to indicate that students value the

Table 8

Respondents Comments Regarding Topics from 100/200 Level Subjects

(C=Critically Important and particularly useful; P=Presented but not in enough detail or depth; O=Overemphasised; R=Repeated)

Topic	C	P	O	R
Language Specific Items.........................	**25 *(35)***	**12 *(25)***	**10 *(13)***	**7 *(4)***
* Logic programming / Prolog..........	2 *(3)*	0 *(1)*	-	-
* Functional programming / Miranda	3 *(2)*	-	7 *(9)*	4 *(0)*
* OOP / C++....................................	4 *(7)*	11 *(20)*	0 *(1)*	-
* C...	16 *(17)*	1 *(0)*	0 *(2)*	3 *(3)*
* Fortran..	-	-	1 *(0)*	-
* Pascal...	-	-	0 *(1)*	-
* Languages not used in industry.......	-	-	3 *(0)*	-
Algorithmic Concepts..............................	**8 *(23)***	**3 (3)**	**2 *(3)***	**1 *(6)***
* sorting and searching......................	1 *(2)*	-	2 *(3)*	-
* hashing...	1 *(0)*	-	-	-
* B-trees...	1 *(0)*	-	-	0 *(1)*
* algorithms......................................	0 *(10)*	-	-	-
* problem solving.............................	2 *(4)*	-	-	0 *(1)*
* linked lists....................................	2 *(0)*	-	-	-
* variables..	0 *(1)*	-	-	-
* data structures..............................	0 *(2)*	1 *(3)*	-	-
* complexity analysis........................	-	1 *(3)*	-	-

* efficiency.......................................	-	1 *(0)*	-	-
Syntax and grammars...............................	**2 *(2)***	**1 *(1)***	**-**	**-**
* grammars......................................	1 *(0)*	-	-	-
* programming language syntax........	1 *(2)*	1 *(1)*	-	-
Hardware...	**0 *(1)***	**1 *(0)***	-	-
* circuit assembly.............................	0 *(1)*	1 *(0)*	-	-
Software development concepts................	**5 *(31)***	**13 *(12)***	-	-
* software testing..............................	0 *(3)*	0 *(2)*	-	-
* version control................................	2 *(4)*	-	-	-
* configuration management.............	0 *(1)*	-	-	-
* revision control...............................	1 *(0)*	-	-	-
* UNIX tools....................................	1 *(8)*	3 *(6)*	-	-
* shell scripts.....................................	0 *(3)*	9 *(2)*	-	-
* makefiles......................................	0 *(3)*	-	-	-
Miscellaneous...	**1 *(0)***	**3 *(3)***	**0 *(2)***	**0 *(1)***
* fuzzy logic......................................	1 *(0)*	-	-	-
* operating systems...........................	-	2 *(2)*	-	-
* binary numbers...............................	-	-	0 *(2)*	0 *(1)*
* commercial use of software tools....	-	1 *(1)*	-	-
* not "real world" programs..............	-	-	1 *(0)*	-

Note: All entries represent the number of students who included a particular topic in their answer.
All entries in italics are based on data collected during the second round of data collection.
All entries in bold represent the total number of comments in each category

opportunity to learn, in-depth, an "industrial-strength language" (an issue which will be discussed in greater depth later in this report). While three respondents on the first survey believed that Miranda was a critical and particularly useful part of their first and second year Computer Science studies, only two thought so in the second survey.

Seven respondents on the first survey believed that Miranda was over-emphasised, while nine felt that it was overemphasised on the second. Generally the reason respondents to both surveys gave for believing that this topic was over-emphasised was that Miranda was not a language used in industry. One respondent suggested that the inclusion of Miranda in first and second year made it difficult for them to learn C. In their words: "it confused me when I got to learn C".

Few respondents (11 altogether on both surveys) identified topics that were repeated in the first and second year Computer Science program. Of those that did, Miranda and basic C programming were most frequently mentioned in each case.

Two particular topics stand out amongst those that respondents suggested were "not presented but should have been included". These were: (1) an introduction to the net and (2) an introduction to "tools". Other topics suggested by respondents were: (a) an introduction to the university's computer system, (b) systems used in industry, (c) email, (d) user interfaces, (e) JAVA, (f) HTML, (g) multimedia, (h) graphics programming, (i) news groups, and (j) x mosaic.

Respondents were generally not impressed with the Departmental Students Manuals. Of the 24 students who responded to these questions in the first survey (38 responded in the second) approximately 20% of students indicated that they didn't use Volume A (13% in the second survey); 30% indicated that they didn't use Volume B (21% in the second survey); 60% of students hardly used, or found volume A "not very useful" (32% in the second survey); while 38% made similar suggestions in relation to Volume B (21% in the second survey). One of the most worrying features of these data is that roughly 30% of respondents to both surveys indicated that they *did not have* a copy of Volume B.

Respondents criticisms with Departmental publications were restricted, however, to these student manuals. Individual subject handouts and the like were generally considered to be "good" by both groups of respondents. Twelve out of the 25 students who responded to this question in the first survey made this assessment (19 did so in the second). Three respondents (4 in the second survey) suggested that they were "excellent" with six respondents (10 in the second survey) suggesting that they were "very handy", particularly "around exam time". Suggestions about how these publications might be improved included: "using diagrams to illustrate points"; "making sure that they are available at the beginning of the semester"; and "making them more specific".

Not surprisingly, given the results discussed above, respondents to these surveys suggested that the best aspect of their 100 and 200 level studies was "learning C or a "real language" in first year" and "working on projects". Among the other aspects of the program which students rated as "best" were: learning C++; having internet access; the wide range of languages presented; and the subjects 433244 Software Development, 433247-Frontiers of Computer Science, and 433245- Programming Languages Principles B. Two students indicated that for them "everything was enjoyable", while one individual commented that the subject "seemed to flow nicely together".(See Appendix B for a detailed summary of these comments).

Three themes seem to run through respondents comments in answer to the question "What was the worst thing about the first and second year Computer Science subjects that you studied?" The first of these appeared to relate to the balance between "theory" and "practice." Six of the 27 students who chose to comment on this question made remarks which seemed to suggest that there was "too much theory" and not enough "practice" or "labs". Three of these students specifically identified "lectures" as the worst feature of their first and second year program. The second of these themes appears to relate to "projects". Four students in total made reference to this aspect of their first and second year studies. In two of these cases the students were concerned about the timing of the projects--particularly the dates of submission for the projects. They believed that "most of the projects were due in at the same time" and that most "were due in at the end of the semester". One was concerned with the fact that "projects were marked late", while the remaining students was concerned with the amount of time which some projects took to complete. The remaining theme concerns the lecturing staff. Comments related to lecturers'/tutors' attitudes towards their classes and students, and the assumption that some lecturers make that students know more than they actually do. As one student put it "When I had to use the TTY terminals you just didn't care!"

In addition to these three themes, students made two other comments regarding the content of the 100 and 200 level program. Both of these comments speak to the issue of connectedness between various aspects of the program. The first, raises the issue of connectedness in some subjects between the content of lectures and the content of the final examination. The second, raises the issue of connectedness between subjects. As one respondent suggested (and I paraphrase) "Some subjects are not well connected. Things not done before are sketched over. Things already done are gone over again in tedious detail."

Students suggestions as to what changes could be made to the teaching of 100 and/or 200 level Computer Science subjects to improve them are summarised in Table 9.

Part C

Students' comments in relation to their experiences and perceptions of 100 and 200 level Computer Science subjects can be categorised under three broad headings: Overall Impressions of the First and Second Year Program; The Content and Structure of the First and Second Year Program; and The Delivery of the First and Second Year Program.

Overall Impressions of the First and Second Year Program

Students who participated in the focus-group discussion were generally pleased with their experience of the first and second year Computer Science program. A number of students expressed the opinion that the program was made up of a good balance between material that was designed to provide an overview of the broad field of Computer Science and material that was aimed at developing specific knowledge and skills in relation to computer program design and development. For students who had little knowledge of, or experience in, the field of computing prior to entering the program, first and second year courses were understood to have provided them with a good overview of the field.

The Content and Structure of the First and Second Year Program

Given that the Computer Science course is designed "to do more than produce highly skilled computer programmers" (comment made by a number of the participants in the focus-group discussion), participants in the focus-group, unlike some of the respondents to the survey, generally believed that together, 100 and 200 level subjects strike an appropriate balance between theory and practice. The majority of the students who participated in the focus-group discussion agreed that the first and second year program not only provided them with the opportunity to develop an understanding of, and skills in, a number of different programming languages and paradigms, but also helped them to develop an appreciation for the research and conceptual bases upon which the study of Computer Science had been built.

Table 9

Students' suggestion for improvement of the 100 and 200 level program in Computer Science.
* Lecturers need to make the subject interesting; *(1)*
* Prolog/Pascal should be taught instead of Miranda; *(11)*
* Need (more?) disk drives; (1)
* Need more PC's; *(1)*
* PC's need to be installed with freeware UNIX such as LINUX (1)
* Need more laser printers;
* Need higher print quotas;
* Need more X terminals; *(1)*
* Need more industry standard software; *(1)*
* Need more lectures in each 200 level subject (3);
* Make material on FTP available to those who need/want it;
* Make material on the WWW available to those who need/want it;
* Don't make projects due during the final weeks of semester or swot vac;
* Address the differences in emphases between lectures and exams in some subjects; *(1)*
* Need a series of seminars going through the packages that the department has to offer;
* 433243 needs to be made more challenging;
* Need more lab time - a better balance between theory and practice; *(4)*
* Need more feedback on projects--more analysis of students' and experts' solutions;
* Lecturers should demonstrate their programming achievements;
* Needs to be a deeper investigation of programming languages in second year;
* Needs to be a deeper investigation of data compression in second year;
* Needs to be a tighter relationship between specified reading and lecture/tutorial content;
* Needs to be less lectures (2);
* Needs to be greater use of applications/audio-visuals in lectures;
* Electronic presentations need to be standardised
* Lecturers need to interact more with first and second year Computer Science students;
* Better coordination and integration between lectures;
* More teaching about C and C++ programming; *(6)*

Note: Numbers and items in italics refer to responses from the second survey

Table 9 (Cont)

Students' suggestion for improvement of the 100 and 200 level program in Computer Science.
* Should avoid assigning the same person as tutor and lecturer;
* Need to attend to the thermostat on Lecture room 1--its always too hot;
* *Provide tutorial solutions;*
* *More needs to be taught about DOS; (2)*
* *A more extensive introduction to programming environments in first year;*
* *More Object Oriented design;*
* *More MS Windows; (2)*
* *244 needs tutorials as well as labs; (4)*
* *Need to assume less knowledge in some subjects;*
* *Need more on UNIX shell scripts; (4)*
* *244 hurdle requirement needs urgent attention;*
* *244 too hard/too much work for the 9.5 points*
* *Need more detail on languages and concepts in 243*
* *Course seems tailored to CS or Eng students only*
* *Need supplemental Assignments for those who fail assignments narrowly*
* *Need to give more weight to projects in assessment*
* *Need to teach programming for graphics*

Note: Numbers and items in italics refer to responses from the second survey

However, despite the fact that there was general support for this position among focus-group members, considerable disagreement existed in relation to the amount of time spent on particular topics throughout the first two years of the program.

Three students (or approximately 40% of those who participated in this discussion) questioned the emphasis that was placed on the study of "Miranda" in the first year of their program. For some of these students this emphasis was believed to be inappropriate because "Miranda was not a language that was commonly used for the development of software in industry"-- it was not "immediately useful." For others, Miranda was believed to be a difficult language to learn. One student remarked "I know that a lot of students chose not to go on with Computer Science in second semester in first year because they had difficulty with, or could not see the point of, studying Miranda in the first semester." However, despite these misgivings, a number of students (four from the focus-group) suggested that the inclusion of Miranda was important because "it showed a very different way to do programming--a way that you would not see if you did, for example, Pascal and then C." "The abstractness (of Miranda) provides a challenge not involved in the learning of C or other languages."

Similar disagreement existed among participants in relation to the issue of "breadth versus depth of treatment". Some participants argued persuasively that the inclusion of multiple programming languages in first and second year Computer Science studies was of vital importance. As one student suggested: "knowing more than one language is helpful in revealing the strengths and weaknesses of each in relation to their suitability to solve particular types of problems." Another put it this way: "Often, learning a contrasting second language can highlight certain features in the first which may not have been picked up when studying that language from the original perspective." One of the students who participated in the focus-group discussion suggested: "Perhaps what is needed is an exposure to a variety of languages as well as an in-depth look at one industrial strength language. That way, students will get the skills they need for the workplace without becoming reliant on one language and afraid to use another."

Despite these differences of opinion regarding the number of languages needing to be taught and the appropriateness of some of these languages, participants in the focus-group discussion were generally supportive of both the content and structure of the first and second year Computer Science course. Comments like: "The subjects seemed to flow nicely together" and "the program gives a good overview of subject" were common among this group of students, as they were among respondents to the survey.

Some concern regarding the timing of particular topics was expressed by members of the focus-group however. In these students' opinions, more opportunity needed to be provided in first year for students to become familiar with the processes of problem analysis and program design. Learning the skills of problem analysis was considered to be something that took time and required considerable guidance and practice. Classes in 433244 Software Development in second year were considered to be useful in this regard, as were the "staged" projects in first year. However, it was believed by most of the participants in the focus-group meeting, that these skills would have been better developed had more time and emphasis been given to them in first year. As one student put it: "it is useful to have worked through a problem for understanding before you start working on a solution."

A critical concern of focus-group participants was the issue of "testing and debugging". In accordance with the respondents to the CS questionnaire, students identified "testing and debugging" as an important and essential skill for Computer Science students to develop in their first and second year studies. As one student put it: "you are expected to test your programs; create your own test data and see if your programs run--marks are deducted if you don't." However, as a number of students indicated, despite the apparent importance placed on debugging by Computer Science staff, insufficient emphasis is given to the topic within the first and second year program. One student in the focus-group meeting put the point this way: "The amount of lecture coverage that it [testing and debugging] gets, suggests that it's a footnote. But you soon learn otherwise." The "problem," according to the majority of students who participated in the focus-group discussion, is that staff assume that students know how to go about doing it. Students are "expected to create their own test data" and solve their own problems with relatively "little guidance." A number of students suggested that what was necessary was a greater emphasis on "testing and debugging" in both first and second year. The consensus of opinion seemed to be that students should be given opportunities in first year to discuss the types of common bugs found in software programs, the ways in which they are identified, and the ways in which they are overcome. Further, it was agreed that first year students needed to be given opportunities to develop the skills of debugging manually before being introduced to the tools that are available. In this way, it was argued, students would have a better understanding of what these tools actually do.

A lack of awareness of the types of tools and resources available and the ways in which these tools and resources work was considered to be a fundamental problem for many students. Focus-group participants suggested that more attention needed to be given in first year to ensuring that students are familiar with the availability and operation of the many tools which the Computer Science department has at its disposal. A number of students who identified themselves as having little or no experience of computing prior to entering the Computer Science course, suggested that the majority of staff assumed far too much in relation to the level of students' knowledge of and skills in the use of these tools. For example, three students, including one who believed that he had quite a bit of computing experience prior to entering the program, indicated that they felt very disadvantaged throughout most of the first year of their program because they were unaware of many of the tools available to them. At an even more fundamental level, some participants suggested that they "knew of students" who had difficulty using such resources as the "world wide web." Regardless of the examples cited, the message from these students was clear: greater attention needed to be paid in first and second year to enhancing students' knowledge of and skills in, the use of the variety of different tools and resources available to them. Students need/want the chance to develop, in closely supervised and supportive environments, their understanding of what tools are available, what they do, how they work, the types of problems that they can help students to solve, and so on.

The Delivery of the First and Second Year Program

Students' comments concerning the delivery of the first and second year computer science course tend to centre around two themes: (1) the integration of lectures, tutorials, and laboratories; and (2) teaching styles.

According to the participants in the focus-group discussion, in comparison with third year, the lectures, tutorials and laboratories in first and second year, at times lack apparent cohesion. One student put it this way: "While tutes and lectures are usually linked, some tutors are not up to date with what's happening in lectures." The problem, from this student's point of view, is that this lack of familiarity with what is happening in lectures, coupled with the difficulties which often arise due to the way public holidays fall, means that tutorials are often "irrelevant". The issue does not seem to be that the material being covered is not important, but rather that it is being presented at a time which makes it inappropriate. While it is important here to remember that these are student perceptions or impressions, and may not in fact be a true indication of what actually happens, these impressions are important because they help to define students' attitudes towards these activities.

To a lesser extent, students expressed a similar concern in regards to the relationship between laboratories, lectures and tutorials. However, the difficulties associated with this perceived lack of integration appear to be exacerbated by the conditions under which students engage in their practical work. Survey respondents and focus-group participants each expressed concerns "regarding the intimidating environment" of some of the computing laboratories. Students expressed feelings of isolation. Lack of immediate recourse to staff, coupled with the fact that they are often "the only first year student working in the lab" amongst a group of more "confident" and "capable" "engineering" or "third year" students were some of the reasons given to explain these feelings. Students' limited experiences of third year, however, have suggested that this feeling of isolation is not as great a problem at this level. The increased emphasis on group work, coupled with the different physical arrangements under which laboratories are conducted appear to account for these differences.

For a number of students who participated in the focus-group discussion, the use of group work in tutorials and laboratories was of particular importance in determining the quality of their learning experiences. As one student suggested: "interacting with others, or teamwork, facilitates the learning process. Discussion is important. Bouncing ideas off of each other as part of solving a problem helps. You get to find out what you really know!" Opportunities to work together; to interact with the Computer Science staff in the solution of appropriately structured problems, appear to lie at the heart of respondents/participants suggestions as to how their experience of first and second year Computer Science can be improved. As a participant in the focus-group discussion put it: "If group work is going to be used in 3rd year then group work should be gradually increased over the first two years." Suggestions for group projects for students in first and second year coupled with information about how to work effectively in teams were put forward as a way of dealing with this issue.

Other issues which appeared to have a significant impact on students' experiences of first and second year Computer Science have to do with the attitudes and teaching styles of the department's staff. Participants in the focus-group discussion suggested that there was a wide variation in the quality of teaching and support offered by the department's staff. Those staff who were considered to be most helpful and supportive were described in the following terms: "they are interesting and interested;" "they invite participation;" "they answer questions cordially;" "they answer the questions asked;" "they are accessible and approachable;" "they offer quality materials--legible and so on;" and "they don't assume you know everything and get impatient when you don't." Those who were found to be least helpful were often described by respondents/participants as simply "not interested." They were difficult to arrange appointments with and often appeared pre-occupied when approached.

Overview

The overall impressions gained by an outsider to the department from the data collected as part of this evaluation exercise are:

* That students are generally happy with the quality of their first and second year Computer Science experiences;
* That students believe the first and second year Computer Science program is generally appropriately structured;
* That students have some concerns over the emphases given to some issues (the study of Miranda in first year) in comparison with others (the insufficient emphasis in lectures/tutes on approaches to testing/, the absence of any formal instruction in the use of debuggers and other programming tools);
* That students perceive that there is a wide variation in the quality of teaching and support offered by the department's staff.

Student suggestions for improving the 100 and 200 level program generally focus on the following issues:

* Improving the degree of cohesion/integration between lectures, tutorials, and laboratory classes (making sure that the content covered in these activities in each week of the program is related);
* Increasing opportunities for students to work together on a variety of different activities throughout their program (to help overcome an existing sense of isolation by providing students with a built in support mechanism);
* Increasing the level of support (by improving accessibility and availability of teaching staff; improving the quality of the Departmental Manuals--providing sufficient background information on new topics to cater for the needs of the novice CS student, etc.);
* Increasing the level of resourcing in computer laboratories (more hardware, software, etc.); and
* Improving the balance in load amongst 100 and 200 level subjects;.

While comments made by respondents in the second questionnaire indicate that students are aware, and view favourably, the alterations to program, resources, etc that the CS department has made in response to the first survey, it is clear that respondents still hold many of their original concerns. This would suggest that more needs to be done, perhaps at a more fundamental level, than was attempted in responding to the first survey, before all students will be fully satisfied with the quality of their 100 and 200 level Computer Science experiences.

References

Delamont, S. (1992). ***Fieldwork in educational settings: Methods, pitfalls, and perspectives***. London, UK: The Falmer Press.

Glaser, B. & Strauss, A. (1967). ***Discovery of grounded theory: Strategies for qualitative research***. Chicago, IL: Aldine.

Lincoln, Y., & Guba, E. (1985). ***Naturalistic inquiry***. Beverly Hills, CA: Sage.

Merriam, S. (1988). ***Case study research in education: A qualitative approach***. San Francisco, CA: Jossey-Bass.

Appendix A

The University of Melbourne

Computer Science Curriculum Survey

This questionnaire seeks your opinions of first and second year Computer Science subjects and your view of how this teaching has contributed to the development of your knowledge and skills in Computer Science.

SECTION A: Your Background

1. The course I am studying is

BSc	[ ]
BE	[ ]
BSc/BE (or BE/BSc)	[ ]
Other (please specify)___________________	[ ]

2. Prior to 1995 I completed the following Computer Science subjects:
(Please [x] relevant box/es ***and*** indicate the year you completed the subject)

			Year
433-101	Computer Science 1A	[ ]	[ ]
433-102	Computer Science 1B	[ ]	[ ]
433-121	Engineering Computing 1	[ ]	[ ]
433-141	Computing Fundamentals A	[ ]	[ ]
433-142	Computing Fundamentals B	[ ]	[ ]
433-151	Introduction to Computing	[ ]	[ ]
433-241	Computer Organisation	[ ]	[ ]
433-242	Algorithms & Problem Solving	[ ]	[ ]
433-243	Programming Languages Principles A	[ ]	[ ]
433-244	Software Development	[ ]	[ ]
433-245	Programming Languages Principles B	[ ]	[ ]
433-246	Mathematical Software A	[ ]	[ ]
433-247	Frontiers of Computer Science	[ ]	[ ]

3. I have also completed the following subjects from other programs which are relevant to my interest in Computer Science: (Please list....)

4. In 1995 I expect to complete the following Computer Science subjects:
(Please [x] relevant boxes)

433-303	Artificial Intelligence	[ ]
433-313	Computer Design	[ ]
433-325	Mathematical Software B	[ ]
433-330	Theory of Computation	[ ]
433-332	Operating Systems	[ ]
433-340	Software Engineering Project	[ ]
433-341	Software Engineering 3A	[ ]
433-342	Software Engineering 3B	[ ]
433-343	Professional Issues In Computing	[ ]
433-351	Database Systems	[ ]
433-353	Networks & Communications	[ ]
433-361	Programming Languages	[ ]
433-380	Graphics	[ ]

SECTION B: Your Skills and Knowledge

(For each question in this section, please [x] the box which best reflects your view.)

E=Excellent A=Adequate P=Poor NE=Non Existent

5. My problem analysis skills are:	E [ ]	A [ ]	P [ ]	NE [ ]
6. My current skills in working as a member of a team are:	E [ ]	A [ ]	P [ ]	NE [ ]
7. My knowledge of imperative (procedural) programming is:	E [ ]	A [ ]	P [ ]	NE [ ]
8. My knowledge of functional programming is:	E [ ]	A [ ]	P [ ]	NE [ ]
9. My knowledge of object-oriented programming is:	E [ ]	A [ ]	P [ ]	NE [ ]
10. My knowledge of logic programming is:	E [ ]	A [ ]	P [ ]	NE [ ]
11. My programming skills in				
a) Pascal are:	E [ ]	A [ ]	P [ ]	NE [ ]
b) C are:	E [ ]	A [ ]	P [ ]	NE [ ]
c) C++ are:	E [ ]	A [ ]	P [ ]	NE [ ]
d) Miranda are:	E [ ]	A [ ]	P [ ]	NE [ ]
e) 68000 assembler are:	E [ ]	A [ ]	P [ ]	NE [ ]
f) Prolog are:	E [ ]	A [ ]	P [ ]	NE [ ]
Other languages (please specify)				
g)________________________are:	E [ ]	A [ ]	P [ ]	NE [ ]
h)________________________are:	E [ ]	A [ ]	P [ ]	NE [ ]
12. My current knowledge of UNIX tools for software development is				
a) for program development (eg, make)	E [ ]	A [ ]	P [ ]	NE [ ]
b) for configuration management (eg, RCS) is	E [ ]	A [ ]	P [ ]	NE [ ]
c) debuggers (eg, gdb, dbx) is	E [ ]	A [ ]	P [ ]	NE [ ]
Other (please specify)				
d)____________________________	E [ ]	A [ ]	P [ ]	NE [ ]
e)____________________________	E [ ]	A [ ]	P [ ]	NE [ ]
13. My current knowledge and skills in				
a) creating efficient programs are:	E [ ]	A [ ]	P [ ]	NE [ ]
b) program testing are:	E [ ]	A [ ]	P [ ]	NE [ ]
c) program design are:	E [ ]	A [ ]	P [ ]	NE [ ]
14. At the conclusion of my Computer Science 300-level studies I expect that my knowledge and skills in				
a) creating efficient programs will be:	E [ ]	A [ ]	P [ ]	NE [ ]
b) program testing will be:	E [ ]	A [ ]	P [ ]	NE [ ]
c) program design will be:	E [ ]	A [ ]	P [ ]	NE [ ]
d) writing user manuals will be:	E [ ]	A [ ]	P [ ]	NE [ ]
e) working as a team member will be:	E [ ]	A [ ]	P [ ]	NE [ ]

SECTION C: Other Issues

(Please [x] the relevant box or boxes.)

15. For your current needs, do you think you know enough about:

If **yes**, where did you learn it?

LE=lectures
TU=tutorials
LA=labs
FR=from friends
O = other

a) using the UNIX operating system?

Yes [ ] No [ ]

LE [ ] TU [ ] LA [ ] FR [ ] O [ ] (specify)

____

b) electronic mail?

Yes [ ] No [ ]

LE [ ] TU [ ] LA [ ] FR [ ] O [ ] (specify)

____

c) reading electronic news groups?

Yes [ ] No [ ]

LE [ ] TU [ ] LA [ ] FR [ ] O [ ] (specify)

____

d) ftp?

Yes [ ] No [ ]

LE [ ] TU [ ] LA [ ] FR [ ] O [ ] (specify)

____

e) X windows?

Yes [ ] No [ ]

LE [ ] TU [ ] LA [ ] FR [ ] O [ ] (specify)

____

f) Web-browsing (eg. xmosaic)?

Yes [ ] No [ ]

LE [ ] TU [ ] LA [ ] FR [ ] O [ ] (specify)

____

16. Have your 100-level and 200-level Computer Science studies adequately prepared you for 300-level study in Computer Science?

Yes [ ] No [ ]

If not, please explain, briefly.

17. Do you think there has been sufficient emphasis in Computer Science subjects on the development of your oral and written communication skills?

Yes [ ]   No [ ]

Please explain, briefly.

18. Reflecting on the first two years of your Computer Science studies, can you identify topics that were:

a) critically important and particularly useful?

Topics:

____

Comment:

b) presented, but not in enough detail or depth?

Topics:

____

Comment:

c) over-emphasised?

Topics:

____

Comment:

d) repeated?

Topics:

____

Comment:

e) not presented but should have been included?

Topics:

____

Comment:

19. How useful has *Departmental Student Manual ,**Volume A*** been for you?

Comment:

20. How useful has *Departmental Student Manual* ,***Volume B*** been for you?

Comment:

21. How useful were other Departmental publications (e.g. individual subject handouts)?

Comment:

22. What was the ***best*** thing about the first and second year Computer Science subjects that you studied?

23. What was the ***worst*** thing about the first and second year Computer Science subjects that you studied?

24. What ***changes*** do you recommend in the teaching of 100-level or 200-level Computer Science subjects? (You may wish to comment on teaching methods; assessment methods; programming platforms; or any other aspect that you feel needs attention).

(Continue over the page if you wish)

Thank you

Appendix B

Respondents' Views as to the Best Aspects of their 100 and 200 Level Computer Science Course

- Brief lecture notes were always provided, working in teams on projects, some lecturers have a sense of humour, Miranda was a fun introduction, having more than one lecturer in 141.
- Most of them are not terribly hard. The lecturers of 243 and 242 (first semester) are really good lecturers.
- Being able to program whole tasks by myself.
- Writing code.
- Efficient programs, 242 or 244 - very useful.
- Seeing a wide variety of languages.
- Learning about pointers.
- Access to a network on three levels i.e. University, national, and international. Certain lecturers showed enthusiasm for what they were lecturing, this enthusiasm rubbed off on me.
- Being taught a <u>real</u> language (C) instead of Miranda! :-)
- Having good lecturers is one. Study what is useful and practical is second. Out of all the subjects, I like 243 best.
- Anecdotes given in Computer Design and Professional Issues in Computing.
- Projects, 244 labs were tough but useful.
- C++ programming language, software development skills.
- 242 - developed algorithm and coding skills, 243 - showed that there is more to life than C, 244 - developed more advanced UNIX skills.
- Tutorials for those subjects that had them.
- Using UNIX with X-Windows.
- Broad range of topics discussed.
- It was interesting, often challenging. Also there are some really good lecturers and tutors.
- I felt that I was learning something useful, and drew great satisfaction from the assignments.
- Achieving producing a program that works and can be used outside teaching.
- I have learnt ANSI C.
- Learning UNIX tools in software development and comparative languages.
- Learning to program.
- 433-202 was an amalgam of stuff, unlike today - hard to say. Didn't need other (newer) 2nd yr. subjects.
- Never found them hard.
- C++.
- Learning programming.
- Very well presented material, very well organised presentation and the material has <u>NEVER</u> been boring.
- Lots of variation and a good background for further study. 245 and 247 GREAT SUBJECTS!
- Lecture notes.

Appendix B
(Continued)

Respondents' Views as to the Best Aspects of their 100 and 200 Level Computer Science Course
• Frontiers of Computer Science because it broadened my knowledge of CS so much and gave insight into research directions of CS. • Flexibility of study - you can go to lectures and study or just as easily study the subject by yourself. It was not crucial to attend lectures but it definitely helped (comments apply to most subjects). • Very interesting, especially for people who haven't had much experience with computers. • Learning how to program in C. Tutorials are generally very good. Projects have been consistently good. • Sense of achievement gained when programs were completed.