MANAGING DIVERSITY: EXPERIENCES TEACHING PROGRAMMING PRINCIPLES

H.C. Davis
University of Southampton
Southampton
UK
hcd@ecs.soton.ac.uk
www.ecs.soton.ac.uk/~hcd

Les Carr
University of Southampton
Southampton
UK
lac@ecs.soton.ac.uk
www.ecs.soton.ac.uk/~lac

Eric Cooke
University of Southampton
Southampton
UK
ecc@ecs.soton.ac.uk
www.ecs.soton.ac.uk/~ecc

Su White
University of Southampton
Southampton
UK
saw@ecs.soton.ac.uk
www.ecs.soton.ac.uk/~saw

ABSTRACT

The majority of university level courses offer a similar experience to all students. However in the teaching of introductory computer programming this practice has become increasingly difficult to justify, due to the widely differing initial experience of students. This paper describes an attempt to provide, at low cost, some level of differing experience depending on the needs of the student. The students were allowed to select for themselves which group to join. The experiment was successful in improving student experience and in demonstrating that the students were capable of selecting appropriate groups.

Keywords
Differentiated Learning, Learning Programming.

1. INTRODUCTION

In the 1980's A level Computing courses became commonplace and at the same time desktop computers became affordable by many students. Ever since then, university teachers of first year introductory programming courses have needed to concern themselves with the diversity of initial skills and experiences of their students. The authors are not aware of any UK Computing degree course that requires an A level in Computing or equivalent as a prerequisite qualification. Given the well publicized career opportunities that are currently available in the IT field, and the fact that not all sixth formers have the opportunity to study computer science, it is probable that most universities experience a similar distribution of initial experience as we do at Southampton. We observe that a significant number of students (around 40%) have an A level or equivalent in Computer Science. Many have a little industrial experience, and a few have worked in IT for a year out. Yet at the other end we observe that a fair number of students have little experience of computers beyond game playing.

2nd Annual Conference on the Teaching of Computing, London

Producing an introductory course in programming principles to suit such a diverse range of students has always been problematic; we must find a way of enabling complete beginners to learn the basics, while providing enough interesting subject matter to keep the experienced programmers enthused. Our typical approach to this in the past has been to teach something that was significantly different to the experience of the majority of students. So, in the late 80's we taught structured programming, and this was a major change from the line numbered Basic's that most students knew. Some of us took a language independent approach as suggested by Bornat [1], and this certainly provided a new challenge for our experienced programmers. By the early 90's the schools and colleges were providing a good standard of structured programming so many universities, including Southampton, responded by starting to teach declarative or functional languages such as SML, Miranda or Scheme. The paradigm shift was a serious challenge for many experienced programmers and they often found the work as difficult as did the new programmers.

In the last few years the demand from students for relevant, directly applicable and up-to-date skills has lead many universities to move to teaching C++ or Java, using a true object oriented approach. The first three authors on this paper were charged with implementing a new Programming Principles course (CM143) in Java at Southampton, starting in October 1999. The module is the flagship course for the Computer Science programme, and is a double unit module, taking one third of the students' first semester hours. The module is taken by 180 students, of whom about 100 are CS majors, and the remainder are mainly computer engineers (electronics and computing) and Maths with

Computer Science students. The CS students follow this course in semester 2 with one unit in Advanced Programming (using both Java and C++) and a course in Data Structures (in Java).

The first year delivery of the course went perfectly smoothly but some problems came to light when the unit was reviewed at the end of the semester.

1. By 5 weeks through the course the experienced students were becoming noticeably frustrated by not being able to use the standard constructs (e.g. loops) that they had used in other languages. The reason for this was that the course had been designed in a very "objects first" approach, with the intention that this would provide some kind of paradigm shift to inspire the experienced programmers. This might have worked to some extent but, quite reasonably, they found the rate of progress through the other programming principles, for which they already had good analogies, tiresomely slow. This information was expressed to us unambiguously through the end of unit student questionnaires.
2. The atmosphere in the lectures had not been as good as one may have wished. Students who found the progress too slow were interrupting the lectures to ask questions that were at a level of detail quite beyond that expected on the course, and this was intimidating to the students who were new to programming, and also to the lecturer, who had to decide at what level of detail to answer the question, if at all.
3. Students at both extremes of the ability range demonstrated rather more dissatisfaction with the course than we were comfortable with.

At LTSN-ICS 2000, Tony Jenkins from Leeds University presented a paper [2] describing a differentiated approach to teaching introductory programming. At Leeds they used an aptitude test to categorize students as either *Rocket Scientists, Averages* or *Strugglers*. Rocket Scientists were already highly proficient programmers. Strugglers were those who would need support to help them achieve a fair standard. A fourth group, *Serious Strugglers*, emerges during the course, and these students are the ones who need the most support. The Averages are those who will pass the course well with only occasional assistance; The Rocket Scientists probably do not need to attend the course and were given alternative projects to keep them enthused. The Strugglers were given extra attention in the form of supervised hands-on lab sessions with high teacher to student ratios.

The Jenkins presentation inspired a considerable amount of conversation during which it became apparent that many universities were attempting to identify different groups of students and to channel appropriate support to these groups. The level of formality of the groupings tended to vary.

At Southampton we were convinced that we would be better able to concentrate on the students who needed support if we could remove the rocket scientists (actually we called them *space cadets*; we felt this best acknowledged their experience while reminding them they still had things to learn) from the mainstream classes. We decided to conduct an experiment in 2000. The aims of this experiment were

- to find out whether student satisfaction would be improved by providing differentiated experiences for the groups of students at either extreme of the initial experience continuum;
- to find out whether students were capable of correctly deciding for themselves which group they belonged in;

The remainder of this paper describes how we restructured the unit, the data we collected from the students, and a comparison of this data with the final results on the course. We conclude by suggesting further work and providing some results which we believe may be useful to the community as a whole.

2. CM143 RESTRUCTURED

Initially all students attended four lectures per week, attended a 2 hour supervised (and crudely marked) hands-on lab per week and completed a coursework every fortnight. Final marks were produced 50% by unseen examination, 40% from coursework results and 10% from lab marks,

In the new scheme, we decided to reduce the standard lecture load to three lectures per week. Space Cadets were not expected to attend these lectures. The fourth lecture was now split into two. The space cadets attended one session, and the strugglers attended another. Those who were in neither group were given independent study time. The space cadets were not required to attend the labs; instead they could carry out the required work independently and bring along their logbook as evidence. Assessment was not altered. Space cadets were required to do all the labs and courseworks, and do the examination.

The space cadets were expected to follow the lecture course by reading chapters from the book and by doing the same lab and coursework as the other students. The one lecture slot they attended was used to set and analyze weekly challenges. These challenges were set with a spirit of enthusiasm for discovery and informally addressed both the higher-level goals (step-wise design, object-oriented decomposition) and practical programming considerations (how to create graphical and interactive programs, how to use the Java API documentation, how to reuse code from the Web). Attendance for the sessions was good, usually reaching about 75%, however it was difficult to encourage students to submit their solutions to the challenges with only the most enthusiastic four or five (10%) regularly contributing. This proportion rose to 50% for a Spirograph drawing applet set as one week's challenge which became a joint assessment exercise in which each student assessed the code of three other students.

The extra Strugglers lectures were taken by the lecturer responsible for the lab course, with the intention that the strugglers would work in a small group in which they would feel confident to discuss their problems, and would be able to revise the work of the previous week with a different person.

3. GROUPING THE STUDENTS

We took the attitude that the students should be responsible for grouping themselves. No student was forced to join any particular group. Indeed we had no firm record of who was in which group as the groupings were fluid throughout the year. Since the main lecture course was at a different time from the space cadet and struggler lecturer, students were free to attend both space cadets sessions and the standard lecture course. A few did.

In order to assist students to select which group they might be in, at the beginning of the year we asked students to complete an Initial Skills Survey. This consisted of six simple multiple choice questions. The most important question was as follows

"Which **one** of the following best describes your attitude at the start of this course on Principles of Programming in Java?"

1	I am an experienced programmer, and I don't believe that I will learn much new from this course. I shall join the space cadet group.
2	I'm a bit of a hacker with a fair amount of previous programming experience. I should be able to manage the routine part of this course with ease. I am considering joining the space cadet group
3	Although I have a fair amount of previous experience with computing, I have little or no experience of the things that this course covers, so I don't expect the space cadets group is the place for me - but I'll keep an eye on it.
4	I have a bit of previous experience, and I expect that with careful attention I will get along fine on this course.
5	I have very little or no previous experience of this sort of computing, and I am expecting to have to work hard for success in this course.
6	I have little or no experience of computing, and I am worried by the challenge of this course - will I be able to do it?

Other questions identified what formal education they had had, what computing experience they had, and what programming languages they knew. Based on their answer to the above we suggested that those who gave answer 1 should join the space cadet group. To those who answered 2 we suggested they gave the space cadet sessions a try, but that they should keep attending the main lecture course until they were confident which way to go. Those who answered 5 or 6 were advised to attend the extra struggler lecture as well as the main sessions.

From the 161 replies to our initial skills survey we extracted the following interesting information.

Previous Qualification	**%**
A Level Computer Science	40%
Other FE/HE Qualification	2%
Professional Qualification (Microsoft/Java Cert. etc)	6%
None of the Above	60%

Table 1: Previous Qualifications in CS.

Initial Confidence	**CS Majors**	**CE Majors**	**Others**	**Total**
1 (Very Confident)	9 (8%)	2 (6%)	0 (0%)	11 (7%)
2	24 (22%)	12 (35%)	0 (0%)	36 (22%)
3	37 (33%)	7 (21%)	2 (13%)	46 (29%)
4	24 (22%)	5 (15%)	8 (50%)	37 (23%)
5	13 (12%)	7 (21%)	4 (25%)	24 (15%)
6 (Unconfident)	4 (4%)	1 (3%)	2 (13%)	7 (4%)
Total	**111**	**34**	**16**	**161**

Table 2: Confidence at Start of Course for Computer Scientists, Computer Engineers and Others (Mostly Maths with Computer Science).

From Table 1 we can see that we had around 65 students with A level Computer Science (and in some cases other qualifications as well), which was around 40% of the class. One would expect that these people, assuming they had been successful in their studies (which is likely given the 22 point minimum A level entry requirement) would rate themselves in the top 2 or 3 initial confidence levels.

From Table 2 we can see that around 45 students rated themselves in the top 2 confidence bands. Our best estimate of the number of people who were treating themselves as space cadets through the course was about 40, so this tallied. The interesting thing was to look at how many students who had no formal qualification still rated themselves as space cadets, and why? A total of 18 students who had no formal experience beyond GCSE had rated themselves in the top two levels of experience. We examined the other questions on their forms, and found the following.

Experience	**#1**	**#2**	**#3**	**#4**	**#5**	**#6**
OO Programming	Yes	No	Yes	Yes	Yes	No
Other Significant Programming	Yes	Yes	Yes	Yes	Yes	No
Web Site Maintenance	Yes	Yes	No	Yes	No	Yes
LAN Network Admin	Yes	Yes	Yes	No	No	No
Total Students	10	4	1	1	1	1

Table 3: Initial Experience of Students who had no A level or equivalent, but still rated themselves in the top two confidence bands.

From table 3 we can see that most of these students may have had good reason to rate themselves as confident. One of the students in column #2 gave inconsistent answers to questions (which made us suspicious of his understanding of the questions), and the student in column #6 may have been over confident. We will revisit the results for these students in the next section.

4. RESULTS

At the end of the module we were able to measure the success of our approach in three ways.

The first thing we did was to plot the final result on the module, for each student, against their initial estimate of confidence. The results are shown in figure 4. They are not entirely surprising. We see that on the whole all the first three confidence bands produced fairly similar marks - although there were a few people in band 2 (maybe the bottom 5 results) who appear to have been overconfident, and might have done better with more structure and help. We are able to identify that these students did not engage with the space cadet sessions, nor attend the main lectures. We also note that (nearly) all the failures (<40%) came from students in the bottom 3 confidence bands. But it is also rewarding to note that a good many students in the bands 4 and 5 still scored good marks on the module.

Of the 18 students identified in table 3 as rating themselves confident but having no formal qualification, all scored over 60%. Twelve of them scored over 70%, including the two students whose confidence estimates were possibly questionable.

The second thing we were able to do was to measure the improvement in student feedback at the end of the course. The student response to the first questionnaire was about 70% of the cohort, and to the second it was 83%. The results are shown in table 5, and demonstrate a significant improvement in student response. In particular the tail has reduced in size and the number of very satisfied students has increased. Since the questionnaires are anonymous it is not possible to relate the results back to the individual students; however from the response to other questions we are able to deduce that this shift is almost entirely due to making the space cadets happier. In 1999-2000 we were able to deduce that around half of those who were unhappy with the course (giving it 1, 2 or 3) were unhappy because the course was too easy. The other half were unhappy because the course was too hard. In 2000-2001 the number who rated the course too easy had was down to a few individuals. We believe that this was the major cause in the improvement in student response.

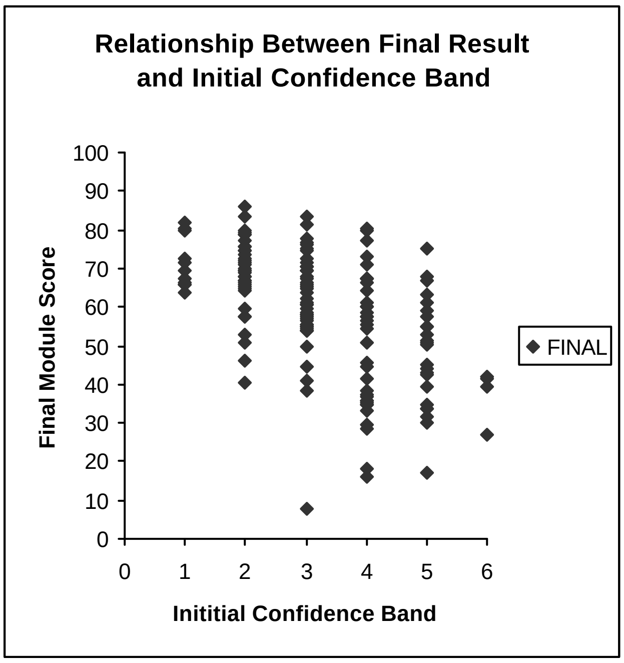

Table 4: Chart Showing Relationship between Final Result on Module and Initial Confidence.

	5 very good	4 good	3 fair	2 poor	1 very poor	Av
1999-2000	13%	61%	20%	4%	2%	3.78
2000-2001	22%	62%	13%	3%	0%	4.04

Table 5: Showing improvement in student answers to the question "Overall, how did you rate this course".

The final test we were able to apply was to ask the students at the beginning of the semester two course "Advanced Programming", how confident they felt about JAVA programming at this point. The results are shown in Table 6. The results show an small reduction in the number who feel unconfident to start this course.

	99/00	00/01
Brilliant. I'm after your job.	5%	8%
Competent. I might not have learnt everything yet, but I can do what I need to.	46%	52%
OK-ish. A bit shaky on some bits and pieces.	32%	26%
Timid. Can do simple programs but I can't see how to write new programs for new problems.	16%	14%
Completely lost! Couldn't write a "Hello World" program without help.	0%	0%

Table 6: Initial Confidence on Advanced Programming

5. RELATED WORK

The principle of enabling differentiated learning has been well established in school teaching [3]. Essentially learning and teaching methods are differentiated to match the needs of individuals within a group. Specific targets for differentiated learning have typically included gifted learners [4] and learners with special educational needs, although the approach is also appropriate to teaching a wide range of educational abilities within a learning group.

Because of the selective nature of Higher Education, there has not been any widespread perceived need for a differentiated approach, and where such approaches have been adopted, they have been set in the context of individual learning differences and accommodating the differentiated needs of different learning styles [5].

In the teaching of computer programming much of the debate focuses around the curriculum and appropriate languages and approaches to the conceptual content rather than on teaching methods which will support individual differences amongst the learners. However there have been some experiments with a differentiated approach [6], [1]. A more expensive solution to this problem is to write an Open University style self paced course, enabling all students to study in their preferred level of detail.

6. CONCLUSIONS

This paper has presented the results of an experiment to improve student perception of an introductory programming course by allowing students to select for themselves which of three courses of study they would follow. By comparison with previous years, the results certainly demonstrate that allowing students who already feel confident to study at their own pace, while providing them with a set of challenges was successful; this group of students were generally much more satisfied with their experience. Also we have shown that such students are capable of self selection; we see no evidence of the need to present students with aptitude tests.

The level of satisfaction with the course for the strugglers did not seem to improve; we had a similar percentage of students who found the course too hard, and who blamed their failure on the course administrators for providing insufficient support. However, the initial confidence on the successor course was encouraging and indicates that confidence has improved on previous years.

We believe that there is room for considerably more research into why a small group of students fail this course every year, although they appear to be very well qualified to study the subject; we believe that usually the problems are environmental rather than intellectual, but have only limited evidence to support this.

The only cost of the re-arrangement of this course has been the need for one extra weekly lecture (the struggler's weekly revision lecture), and believe that this investment has been worthwhile.

7. REFERENCES

[1] Bornat, Richard Programming from First Principles. Prentice Hall International,1987

[2] Jenkins T & Davy T, Dealing With Diversity in Introductory Programming paper. LTSN-ICS 2000, Edinburgh. http://www.ics.ltsn.ac.uk/pub conf2000/Papers/jenkins.htm

[3] Tomlinson C A, The Differentiated Classroom: Responding to the Needs of All Learners Association for Supervision and Curriculum Development, Alexandria, Virginia 1999

[4] Brown M E and Riley T L, Clever Kids and Computers: Catching the Wave. acec98 Australian Computers in Education Conference Adelaide Australia 1998 http://www.cegsa.sa. edu.au/acec98/papers/p_brown.

[5] Huber T & Pewewardy C "Maximising learning for all students: a review of literature on learning modalities, cognitive styles and approaches to meeting the needs of diverse learners." CORE. Vol. 14, no. 3: 90

[6] Jenkins T A participative approach to teaching programming; Proceedings of the 6th annual conference on the teaching of computing/3rd annual conference on integrating technology into computer science education on Changing the delivery of computer science education, 1998, Pages 125-129