Issues in teaching object technology to joint honours undergraduates
P. Mclaughlin, G. Lovegrove

School of Computing, Staffordshire University, Leek Road, Stoke-on-Trent, Staffordshire, ST4 2DF, UK

Abstract
The problems of teaching object technology in mainstream computing undergraduate courses are currently being debated. Object technology is also featuring beneficially in joint and hybrid courses but this brings its own problems. These are especially relevant to increasingly popular, modular courses where there is an even greater diversity of previous and complementary study than at present. This paper discusses issues raised by the particular problems encountered when teaching object technology to joint honours undergraduate students for the first time. In 1994/95 some thirty students enrolled for a second level object oriented programming option course as part of the computing half degree in the scheme of applied sciences offered at the Stoke site of Staffordshire University. It is the experience gained from teaching this cohort that forms the basis of this paper.

1. Introduction

This paper discusses issues raised by the particular problems encountered when teaching object technology to joint honours undergraduate students for the first time. The problems of teaching object technology in mainstream computing undergraduate courses are currently being debated [1, 2]. Given the recent general move to modularised degree schemes, it is thought that the lessons learned at Staffordshire may be of benefit to the wider object technology teaching community.

It has not been simple to introduce object technology into traditional single honours courses in computing and information systems. One of the prime requirements of such courses is to equip graduates with skills considered essential by the profession and in wide demand by industry.

Until very recently object technology has not been considered a commercially relevant skill and has tended to appear in courses as a specialised (often final year) option.
Given the amount of pressure on students to assimilate ever more complex changes in computing, even in core subjects, object technology has tended to suffer. Such problems are also experienced in joint and hybrid courses with considerable computing content. Paradoxically, it is often easier to change the curricula of such courses as a multi disciplinary approach has to be more flexible than a single honours degree.

The computing option of the scheme of applied sciences offered at the Stoke site of Staffordshire University fits into this category and in 1993/94 a second level object oriented programming option was offered for the first time. Only seven out of approximately forty students enrolled and although general observations were possible, quantitative results were largely meaningless. In 1994/95 thirty students enrolled for the course from a similarly sized cohort and it is the experience gained from teaching this cohort that forms the basis of this paper.

The rest of the paper is further divided into four sections. The first is an outline description of the course; the second describes the results and the student experience of the 1993/94 course. The third makes observations from an academic perspective and the last section summarises the paper and details our conclusions.

2 An outline description of the course

The computing option of the degree in applied science at Staffordshire forms a complete half degree in computing. In level one three subjects are studied and students choose two for further study at levels two and three at the beginning of level 2. This can be quite significant as some students choose computing as a third subject intending to drop it at the end of level one. For various reasons they then decide to carry on with computing as their second subject - often with considerably less enthusiasm than for their other subject.

The computing option is a general course covering subjects in software engineering, information systems and computer science. All students learn Pascal in level one and continue to study it in level two in core software development subjects. Additionally in level two, students are required to study two other language modules. The choice is assembly language, C, COBOL or Object Oriented Programming.

The course (CO210 OOP) was based on a 14 week semester, with 12 weeks devoted to teaching, two hours per week. The basic course was divided into three sections thus;

- basic OO concepts,
- OO programming,
- introduction to OO analysis and design.

Similar structures are described by Weiner & Pinson [2] and Wu [5, 6].

When the OOP course was devised it was decided to use Smalltalk as the implementation language in order to compel students to learn and use a pure
object paradigm and Digitalk's Smalltalk/V for windows was selected as the most appropriate tool for this purpose. There is good coverage of the dialect in JOOP and some of the newer Smalltalk text books which are specific for this dialect are excellent books for teaching and project work [3, 4]. The analysis and design content was introductory because of the timescale involved. The method chosen was Rumbaugh's OMT [7] because of its clarity for simple models and the existence of a good text.

3 The results and the student experience

The results (before resits, due in September 1994) are presented here in order to provide a context for the results of a survey of the students and are as follows. Out of 30 students 19 passed at the first attempt with 11 nominal fails. Due to course regulations that allowed students to fail one course given a suitably good profile otherwise, not all of the 11 fails would be required to do some form of referred work. This split is similar to the other language options offered. The detailed breakdown of grades is shown in Fig. 1 below.

![CO210 1993/94 Results](image)

Fig. 1 CO210 1993/94 Student results

The students were surveyed by questionnaire at the end of the course and the results are presented below. The survey was not intended to be comprehensive but was designed to search for reasons why some students were finding object technology difficult to absorb. The results then should be taken to be illustrative rather than absolute. The sample size was 21, representing 70% of the students that took the course (30).

Questions one and two concerned identity

Question 3

What was your other language option choice?

i) Assembly Language 4

ii) C 17

Comment:
C is quite clearly the choice of the majority of the students, reflecting a pragmatic attitude amongst most of the students that it the language most likely to secure employment.

**Question 4**

**Did you choose OOP because:**

i) it sounded interesting 7 (33%)  
ii) it seemed useful 12 (57%)  
iii) to avoid other language options 0 (0%)  
iv) because of the staff involved 0 (0%)  
v) other reason 2 (10%) (both stated its choice was because it is a new technology)

**Comment:**
This supports the notion that the students have a largely pragmatic desire to learn OO rather than for the academic interest of the subject. This could be significant as learning such a lot of material in such a short space of time requires considerable enthusiasm and some of the students lacked a sufficiently high level of commitment.

**Question 5**

**Have you understood OOP concepts ?**

i) very well 0 (0%)  
ii) well 12 (57%)  
iii) not very well 9 (43%)  
iv) not at all 0 (0%)

**Comment:**
The students' perception of their understanding of OO is largely in agreement with their results although they tended to underrate their achievement. In the survey 57% thought they had grasped the concepts well whilst 63% passed the course at the first attempt. In the survey 43% thought they had not grasped the concepts very well whilst only 37% of them failed or require referred work.

**Question 6**

**Did you know how to use windows before starting this OOP course ?**

i) Yes 20  
ii) No 1

**Comment:**
This particular cohort had not been exposed to Windows use or development as part of the taught course at level one. Because the students had not done any software development under Windows, time had to be spent explaining basic Windows concepts as well as how Smalltalk dealt with them and this reduced the time available to develop their Smalltalk programming skills.
Question 7

How do you find using the Smalltalk/V environment?

i) Easy 3 (14%)
ii) Strange but easy 11 (52%)
iii) Strange & hard 7 (33%)
iv) just hard 0 (0%)

Most of the students (85%) found the Smalltalk development environment 'strange'. This was not unexpected as their previous study had been traditional edit, compile, link and run type tools such as VAX Pascal or Turbo Pascal. Most (67%) of the students found the environment 'easy' to use whilst 33% found it 'hard' and this bears out the experience of supervising practical sessions. Once students had realised the power of the multiple windowing, incrementally compiled style of the development environment they found it easy to use. Those that found it difficult to use were (in the opinion of the authors) also liable to find conventional programming difficult.

Question 8

How do you compare using Smalltalk/V to other languages? (Open question - results summarised as)

i) Pascal
   Smalltalk harder: 9
   Smalltalk same difficulty 2
   Smalltalk easier 0
   Couldn't compare 2

ii) C
   Smalltalk harder: 0
   Smalltalk same difficulty 3
   Smalltalk easier 5
   Couldn't compare 1

iii) Assembly Language
   Smalltalk harder: 1
   Smalltalk same difficulty 0
   Smalltalk easier 3
   Couldn't compare 0

Comment:

This question was deliberately left open in order to obtain as many opinions as possible.

What was surprising was the similarity of most of the responses. So much so that it was possible to construct a classification as shown.

The results for Pascal are not very surprising given the amount of time spent learning Pascal in levels one and two and can be attributed to a comparison between the comfortable and familiar rather than the new and unknown.

The results for C are more interesting as this language is also encountered at level two for the first time. It can quite clearly be seen that those who
responded to the question thought that Smalltalk was easier to learn than C. At first glance this would suggest that Smalltalk is easier to learn than C, yet C is conceptually similar to Pascal which the students found much easier to learn than Smalltalk. Instead it is thought that the students find it hard to learn detailed C syntax. Smalltalk syntax is simple, it is the concepts that are new and require the greatest amount of effort. This was also supported by some student comments. Questions 9, 10, 11, 12 concerned course delivery.

Question 13

How have you found the course

i) useful and interesting 6 (31%)
ii) useful 9 (47%)
iii) not useful 3 (16%)
iv) not useful and uninteresting 0 (0%)
v) other 1 (6%)

Comments:

Of the 19 students that responded to this question, 31% (6) said that they had found the course useful and interesting and a further 47% (9) that they thought it was useful. This is another result that suggests a pragmatic approach to choice of programming language options with 78% of the students saying they found the course useful whilst only 16% (3) thought that the course was not useful to them.

Whilst there is no firm basis for comparison, it is interesting to compare the pass / fail division of 19 out of 30 passing at the first attempt with 16 out of 30 saying that they found the course useful.

4 Observations from an academic perspective

The results from the course are comparable to the other programming language options and in this respect must be seen as very positive considering that the students were completely new to object technology, windows software development, incrementally compiled development environments and OO A&D (Analysis & Design). However in absolute terms the amount of progress was disappointing. Whilst the better students made good progress, those that were struggling found it very hard to progress at all. The students grasp of A&D concepts was very weak, however this is also true of traditional A&D courses and is not considered to be a particular problem with object technology although other work at Staffordshire [1] suggests that OMT may not be the best method to teach initially. It is considered that the following factors accounted for these observations;

1. motivation
2. contact time
3. lack of background knowledge (Windows rather than OO)
It is difficult to pinpoint the lack of motivation of some students compared to others. An attempt to classify achievement according to the student's other subject proved inconclusive. This tends to strengthen the observations made during the course that those students who well in object technology also did well in other computing subjects, whilst in general, those that did not do well in OO were also weak in other subjects. This is thought to be caused by the students' preference for their other subject. In some cases, as described in section 2 students opt for computing because their other subject results at the end of level one are much worse than anticipated. Thus they study computing 'because they have to' rather than because they really want to. Whilst this attitude is disquieting it is quite legitimate and the move to modularised courses could exacerbate the problem.

The contact time allocated to the course was the same as for the other programming language options, yet the students have by that stage spent at least one year studying imperative programming and the paradigm is familiar even if the language is not. It was quite clear that a large number of the students needed more help in overcoming the basic learning curve of object technology but it was not possible to provide this.

This is probably the most important conclusion of the study and has been addressed in the modular schemes by introducing two modules, one in OO A&D (Analysis and Design) with an exposure to programming and one in OOP with an exposure to OO A&D. Both modules run for three hours contact time per week for 12 weeks thus providing the potential OO student with three times as much staff assisted support. The extra tuition and support available in the new scheme compared to the old one is considered to be the most effective method of overcoming the initial problems faced by the prospective OO student in a joint honours course.

5 Conclusions

It is concluded that most of the students that took the OO course did so with a pragmatic desire to increase employability in a modern computing discipline. The problems encountered by these students can be classified according to motivation and ability. The diversity of motivation towards computing and OO is set to become wider under modularity. In order to increase the number of students taking OO and increase the quality of the OO teaching it has been considered necessary to offer and encourage students to take two courses rather than one and within the courses to spend more time initially on OO concepts rather than OO language features as with traditional programming education.

The results from the 1993/94 academic year (before resits) suggest that insufficient time was available to cover the basic object paradigm and that this led to slow progress in the practical programming part of the course. A generally poor grasp of analysis and design could be attributed to a lack of course time and a general antipathy on the part of the students towards analysis and design courses however other work at Staffordshire [1] suggests
that OMT may not be the best method to teach initially. The CRC method of Wirfs-Brock [8], Fusion method of Coleman [9] and ObjectrOry method of Jacobson [10] are under active investigation. The course results and those of the survey, also suggest that whilst the more motivated students did quite well, the less motivated and less able found the course to be disproportionately difficult. This was thought to be due to the students' general motivation towards computing subjects in a multi disciplinary scheme where computing may not be their favourite subject.

The implications of these results for modular schemes suggest that more time is needed to introduce the object paradigm. The range of previous study is set to become even wider. Students on half, major and minor awards will have already spent at least one year with the imperative paradigm, usually in traditional languages and environments such as Pascal and need more help to overcome the initial object learning curve. Other students on hybrid course will have different profiles and still other students may decide to take OO as an elective choice. At Staffordshire this has been addressed by designing two OO modules. Provisional enrolment suggests that the extra time available will be beneficial as the backgrounds of the students include half degree courses in computing, information systems and hybrid Interactive Systems Design courses.

6 References


3) Shafer, D., Smalltalk programming for Windows, Prima Publishing , 1993


5) Wu, C.T., Three stage incremental approach in teaching OOP, JOOP, 1993, Vol. 6, No. 3


