Object-oriented analysis and design: a question of approach

H.C. Sharp, M.A. Newton
Faculty of Mathematics and Computing, The Open University, Walton Hall, Milton Keynes, MK7 6AA, UK

Abstract

Later this year, the Open University (OU) will offer the first presentation of a new course in object-oriented software technology. This course covers a wide range of issues from programming in Smalltalk to object-oriented analysis and design, to management of object-oriented projects. It is part of the Masters degree programme which is aimed at individuals who are involved in software development professionally. As with all OU courses, the course is to be taught using distance teaching techniques, so course materials are presented through a combination of media including written text, computer software and video. Each course is expected to have a lifetime of about six years – a long time in a field which is moving as rapidly as object-orientation.
This paper discusses our experiences of designing the object-oriented analysis and design element of the course: the approach chosen, the difficulties encountered, and the solutions devised.

1 Background

Choosing what you want to teach about object-oriented analysis and design (OOA&D) and what you want to use as a basis for this teaching is not easy. Many OOA&D methods have been developed, and whilst there are published papers which compare them [1, 2], these comparisons do not usually focus on educational criteria. Conferences which have concentrated more on education [3, 4] have included papers relating to teaching environments, programming languages, object orientation in the undergraduate curriculum, group working and the

need to use real world problems as case studies, but little has been reported about OOA&D teaching.
Key questions for us were What are the important issues of OOA&D? How should they be taught? What method, if any, should the teaching be based around? D'Souza [5] offers some interesting comments and suggestions regarding education and training for object orientation, but does not answer these questions. There have been some useful experience reports to draw on [6], but little which pulls together relevant issues.
In this paper we discuss some of the main issues we encountered in designing the OOA&D element of a postgraduate course which has been developed over the past two years.

2 M868: Object-oriented Software Technology

The course in which we are teaching object-oriented analysis and design, and from which this paper arises, is part of the Open University's programme: Computing for Commerce and Industry (CCI), which leads to an MSc. The programme is targeted at people who want to study part-time, usually because of work commitments, but in particular at those who have considerable computing experience and want to update their knowledge, obtain a theoretical basis for their skills or simply obtain a qualification relating to their work.
Although every student is assigned to a tutor, there are no mandatory tutorials, and in some cases the only contact with the student is via three written assignments completed throughout the six month presentation. Students are therefore working on their own with little technical support, and although courses in the (not too distant) future may use technology to support group working at a distance, the technology is not yet cheap enough, nor sufficiently understood in the distance education arena, for us to be able to rely on it.
The course is called "Object-oriented Software Technology" and it consists of about 100 to 120 hours of study. The development of the course is overseen by an external academic assessor and an external industrial assessor. Its scope includes various aspects of the object-oriented approach to software development, with the overall aim of providing an understanding of the use of object concepts both for programming and for analysis and design, together with an appreciation of the implications for working practices in adopting an object-oriented approach. The analysis and design element of the course accounts for approximately 3/8 of the teaching material.
The initial issue was whether to introduce programming before or after analysis and design. We decided to teach programming first, because it provides a basis for the object concepts required when doing analysis and design – we did not consider it appropriate to start teaching analysis by having to introduce object ideas in the abstract. We also expected our students to have some programming skills, though we

could not make assumptions about any particular language. There is a healthy debate currently being waged about whether knowledge of procedural languages helps or hinders the acquisition of object expertise [7], but there is no conclusive evidence either way. We believe, however, that teaching object-oriented programming first may help to counter any possible misconceptions. Smalltalk was chosen as the programming language because we thought that it could be introduced with minimum prerequisite knowledge, and that it provided a sounder basis for object concepts than, say, C++.

Similar general considerations influenced our approach to analysis and design. It was quite possible that students would have some previous experience of a conventional method for analysis and design, in particular from another course within the CCI programme, but we could not make knowledge of any approach, whether based on a structured or a data model method, a prerequisite for the course.

Thus there was an inherent conflict in what we wanted to do: identify the benefits of an object-oriented approach while not being able to do it in a detailed comparative way. Our resolution of this problem was to have a short description of the differences at a fairly abstract level, which includes a brief description of conventional development, to exclude comparative comments in the actual teaching of OOA&D, and to discuss the differences in more detail in a later part of the course which covers management aspects. This later part aims to show that it is possible to simply substitute OOA&D for a conventional method, resulting in some benefits in understanding and flexibility, but that the benefits which may arise from reuse require a significant change in the organisation of work and the responsibilities of individuals.

The working environment of our students leads to other influences on our courses, where the relevance and applicability of the material to commercial and industrial practice is an important factor for students and their managers; however, we have to interpret this requirement in as broad a way as possible, so as not to exclude students.

These considerations led to a set of requirements on the approach we should take in our teaching of analysis and design.

3 Criteria for choosing an OOA&D method

M868 is not intended to be a training course for a particular analysis and design method. Our aim is to equip students with enough knowledge and experience to be able to appreciate the basic concepts involved in this approach to system development, and to be able to make sense of any methods which they may encounter in their professional lives. Although we were trying to remain method-independent, i.e. not to concentrate on one commercially-available method, it became clear early on that this was not feasible.

There are areas in which methods agree. For example, most methods include techniques for identifying and documenting classes and their relationships, and for modelling the dynamic behaviour of the classes. However, the techniques vary considerably, and unlike conventional software development which has been practiced long enough for some 'common wisdom' to have emerged, we could find little common wisdom in object-oriented analysis and design. Indeed, even the areas of apparent agreement are, on close inspection, misleading. For example, there are significant differences in the treatment of relationships between classes, in terms of both how they are identified, and how they are represented.

We therefore had to choose one of the many analysis and design approaches to teach. In making this choice, the following issues were considered:

- The approach should be credible commercially. Teaching an academic method, or one which has not been used in industry is likely to dissuade companies from encouraging their employees to take the course, and put off the kinds of student who come into the CCI programme. Although this is not a training course, we still need to impart practical skills;
- The techniques should be as mature as possible so that problems will have been smoothed out. Since the course has to survive for about six years, the approach needed to be reasonably stable;
- It had to be possible to teach the approach at a distance and to give students practice in the skills required by the method. An approach often advocated in academia and industry as an effective way of educating teams into the object way of thinking is to start using a technique such as CRC cards [8, 9], or an 'Object Game'. These techniques involve team members role-playing, acting out scenarios with each of them pretending to be one of the objects identified for the system. This has also been found to be beneficial for team building. In the distance education context, this option is not viable, since we need to consider students working on their own;
- We wanted to provide students with a complete view of object-oriented development, one which would cover all stages from requirements definition through to implementation. The approach we chose therefore needed to provide a rounded view of the process; integrating techniques 'on the fly' in a course such as ours is not practical.
- A clear distinction between analysis and design was desirable. This was to facilitate the sequential teaching style which had to be adopted. Although we wanted to emphasise that designing object-oriented systems is an iterative process, we have to present the material in a sequential fashion; an approach which had a clear analysis/design boundary would help in this;
- There should be a comprehensive text book source and adequate published examples of how to use the approach. This was an important consideration for three reasons:

1 the course team needed a reference text in order to prepare the teaching material;
2 we decided to use case studies as the basis for the teaching material so as to give the students practical OOA&D experience (three case studies are included in the analysis and design section of the course), therefore a set of readily-accessible practical examples of the approach being applied was essential;
3 students may refer to the text if they require a supplement to our teaching.

Based on the above criteria, we chose to use OMT [10], although we would not wish to claim that this is the only approach which fulfils the above criteria. There were two principal reasons for this choice: Rumbaugh *et al*'s book is clear and contains a substantial number of worked exercises; industrial contacts confirmed its commercial credibility.

4 OMT: the chosen one

OMT (the Object Modeling Technique) was developed at General Electric Research and Development Center. It takes a modelling approach to analysis and design, and is based around three models which represent different views of the system: an object model, a dynamic model and a functional model.

1 An object model, which captures information about the classes and their relationships, is the first model to be developed, and in many cases, is the most important model. This model represents the stable structure of the system and is the one around which the system will be built;
2 A dynamic model has two components: a set of diagrams which represent a life history for each class, describing its states and the events to which it responds, and a diagram which shows how the classes interact. This model concentrates on the control aspects of the problem;
3 A functional model concentrates on the computational or processing aspects of a system. It records the details of the computations which the system must perform.

The object model is developed first, followed by the dynamic model and finally the functional model. Once all three models are available the method follows an iterative development in which the three models evolve, thus eliminating the need to change notation part way through development – a situation which often causes problems in conventional development. The approach divides OOA&D into three phases: Analysis, System Design and Object Design. A distinct activity which involves translating the resultant design into an implementation is also recognised. OMT compares against the criteria described above as follows:

- *Commercial credibility*. One of the indicators of the acceptance of any method within the commercial environment is the availability of CASE tools to support it. A number of these to support OMT have appeared on the market, and there is evidence that tool suppliers are finding that OMT is very popular [11]. In addition, discussions with industrial contacts confirmed that OMT is one of the approaches being adopted widely in industry.
- *Maturity*. OMT has been widely-available for many years, has been applied to many different systems, and is relatively mature;
- *Distance teaching*. OMT does not rely on team interaction. Classes are identified initially from a requirements document, and a set of guidelines is given for refining the list;
- *Coverage of development*. OMT includes guidance for all phases of development, from requirements analysis to implementation. Although there are problems with the approach, and there is less guidance for the later stages of development, OMT was one of the first widely-known methods to cover all phases of development.
- *Separation of analysis and design*. OMT has a distinct phase for analysis and two phases for design.
- *Reference material*. Rumbaugh *et al.*'s book [10] is widely available. It contains exercises at the end of each chapter, with selected answers, and three example applications. In addition, various articles have looked at applications of OMT.

5 Difficulties encountered and solutions adopted

- In common with any method, there are gaps in OMT, and although it is quite mature compared to other object-oriented analysis and design approaches, there were areas in which the guidance available was not adequate. The principal concern was how to integrate the different viewpoints. To overcome this, we showed how the different models related to each other as they evolved, and how they could all be used to progress the definition of operations. At the end of analysis, we drew up a table which indicated where elements in one model may be reflected in another model, and then took this unified view forwards into design.
  Another area of difficulty in OMT is the use of a functional model. The need for such a model is disputed by many in the object-oriented world, as being a throw-back to functional decomposition. This aspect of the model is being revised by the developers of OMT in recognition of this, but it is not being completely dropped. We decided to introduce this model, but to accord it less emphasis that the other two models.
- Since the first part of the course introduces object orientation using Smalltalk, students embark on the analysis and design section of the course with a particular view of what is an object. They must change their mindset on coming into analysis and design so as to be able to

use the concepts but reject the implementation bias. This problem has been approached by highlighting how ideas can be interpreted in a Smalltalk environment, but also emphasising when such interpretations are inappropriate. For example, in analysis and design, attributes should not be objects, whereas Smalltalk leads you to believe that everything is an object, including attributes.

- The notation used in OMT is both complex and at times vague, for example the meaning of aggregation. For teaching purposes, we need to be precise and explicit, especially because of the circumstances of our students, and the fact that the course has a life of six years or more, i.e. we cannot change the presentation next year! This has had two implications. Firstly, we have had to describe explicitly the meaning of the notation, or indeed anticipate any misunderstandings suggested by the terminology used. Secondly, we have only introduced enough of the notation to allow the development of the case studies. Although this may sound limited, all the fundamental concepts are covered, and in dynamic modelling, many of the advanced elements are introduced.
- In common with other OOA&D methods, OMT is evolving. In an attempt to maintain the accuracy of the course for as long as possible, we have included teaching, albeit brief, about CRCs and Use Cases [12].
- Although we sought a method with distinct analysis and design phases because of the sequential nature of the teaching style, analysis and design is not a rigid sequence; we have to allow for the prototyping and cyclic nature of development. At the same time, it would be inappropriate to present deliberately incorrect models to students as this may cause confusion. We have overcome this by emphasising that the models produced are the result of a number of cycles and discussions, although there are also occasions on which 'errors' have been found to illustrate the validation mechanisms in action.
- Analysis and design is usually a team activity. Imparting the flavour of object-oriented development in a team setting is achieved via the video element of the course, which includes an excerpt from an early design meeting.

Summary

During the development of the course on Object-oriented Software Technology at the Open University, we have had to consider a number of issues related to the distance teaching of object-oriented analysis and design. This paper describes the constraints and influences of the OU style of teaching, and the requirements we identified for what and how OOA&D should be taught as part of this course.

In particular, we explain why we had to choose a specific methodology as the basis for our teaching, give our criteria for choosing a methodology and show how OMT satisfies these criteria.
Finally we describe some of the difficulties in the use of OMT that we encountered in the development of our teaching material, and the solutions we adopted. However, we have not yet had feedback from students studying our course materials, and so look forward to evaluating the effectiveness of our approach.

References

1 Arnold, P. et al. Evaluation of five object-oriented development methods, *JOOP*, 1991, pp 101–121.
2 de Champeaux, D. & Faure, P. A comparative study of object-oriented analysis methods, *JOOP*, 1991, pp 21–33.
3 TaTTOO '94, Proceedings in *Object Technology Transfer*, Alfred Waller, 1994.
4 OOPSLA '92 Educator's Symposium, in addendum to *Proceedings of OOPSLA'92*, ACM Press, 1992
5 D'Souza, D. *Education and Training, JOOP*, 1992, **5**.
6 Bruegge, B., Blythe, J., Jackson, J. & Shufelt, J. Object-oriented system modeling with OMT, pp 359–376 *Proceedings of OOPSLA'92*, ACM Press, 1992.
7 Sharp, H.C. What do programmers need to become object-oriented? in *Object Technology Transfer*, Alfred Waller, 1994.
8 Beck, K. & Cunningham, W. A Laboratory for Teaching Object-Oriented Thinking, *SIGPLAN Notices*, 1989, **23** 11.
9 Wirfs-Brock, R.J., Wilkerson, B. & Weiner, L. *Designing Object-Oriented Software*, Prentice-Hall, Englewood Cliffs, NJ, 1990.
10 Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F. & Lorensen, W. *Object-oriented Modeling and Design*, Prentice-Hall, Englewood Cliffs, NJ, 1991.
11 Frost, S. *OMT – Taking it Further*, Select Software Tools Ltd., Cheltenham, 1994.
12 Jacobson, I., Christerson, M., Jonsson, P. & Overgaard, G. *Object-oriented Software Engineering: a use case driven approach*, Addison-Wesley, 1992.